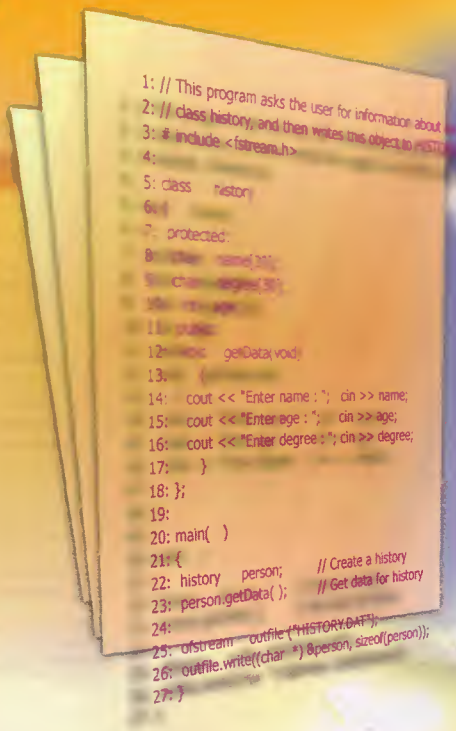
Microsoft Windows



C++ Complete

AUNG MYINT (M.E., AUSTRALIA)

CHAPTER 1
C++ BASICS

CHAPTER 2
FUNCTIONS

CHAPTER 3
PROGRAM FLOW CONTROL

Complete

CHAPTER 4
POINTERS AND ADDRESSES

Complete

Chapter 1

1979 ခုနှစ် နှစ်ဦးပိုင်းလောက်က အမေရိကန်ပြည်ထောင်စု ၊ New Jersy ပြည်နယ် ၊ AT & T Bell Laboratories မှာလုပ်ကိုင်နေတဲ့ Bjarne Stroustrup ဆိုသူကွန်ပျူတာပရိုဂရမ်မာတစ်ဦးက C language ကိုအခြေခံပြီး C++ ဆိုတဲ့ object-oriented programming language တစ်ခုကိုရေးသားခဲ့ပါတယ်။ အစက C++ မဟုတ်ပါဘူး။ C with Classes လို့အမည်ပေးခဲ့ပါတယ်။ 1983 ခုနှစ်ရောက်မှ C++ လို့နာမည်ပြောင်းလိုက်တာပါ။ C++ က C ကိုအဆင့်မြှင့်ထားတဲ့ version ပါပဲ။ C++ source code တွေကို C compiler တွေပေါ်မှာ run နိုင်အောင် Cfront ဆိုတဲ့ interpreter ကိုအသုံးပြုခဲ့တယ်လေ။ တစ်ချို့က C++ ကို C ရဲ့ superset လို့ပြောကြပါတယ်။ C language မှာ လုပ်လို့ရတာအကုန်လုံးလောက်ကို C++ မှာလည်း လုပ်လို့ရတာကိုး။ တစ်ခုတော့ရှိတယ် ၊ C++ language မှာအသစ်ထပ်ဖြည့်ထားတဲ့ keyword တွေက C language မှာ မပါဘူးဆိုတော့ အဲဒီ keyword တွေကို C program မှာ identifier တွေအနေနဲ့ အသုံးပြုလို့ရတယ်ပေါ့။ အဲဒီလို C program မျိုးကို C++ က compile လုပ်လို့မရပါဘူး။ C compiler ကပဲ compile လုပ်ပြီး run ပေးနိုင်မှာပါပဲ။ ဘာပဲဖြစ်ဖြစ် စာဖတ်သူအနေနဲ့ C language ကိုသိနေပြီးသားဆိုရင် C++ ကိုဆက်လက်လေ့လာတဲ့အခါ လွယ်ကူချောမောမှာပါ။

၁.၁ A Brief History of C and C++

C language ကို Dennis Ritchies ဆိုသူ ကွန်ပျူတာပရိုဂရမ်မာတစ်ဦးက AT & T Bell Lab မှာ လုပ်ကိုင်နေရင်း 1970 ခုနှစ်မှာ interpreted programming language BCPL ကို မှီးပြီးရေးသားခဲ့တာဖြစ်ပါတယ်။ ရည်ရွယ်ချက်က C နဲ့ low-level programming ကို Assembly language လိုပဲ လုပ်နိုင်ဖို့ပါပဲ။ ဆက်သွယ်နည်းက လူတိုင်းအသုံးပြုလို့ လွယ်ကူတဲ့ high-level language နဲ့ဆက်သွယ်တာမျိုးပေါ့။ Ritchie ဟာ သူလက်ရှိအသုံးပြုနေတဲ့ UNIX system PDP-11 စက်မှာပဲ C ကိုလက်တွေ့စမ်းကြည့်တော့ အောင်မြင်သွားပါတယ်။ ဒီအခါမှာ UNIX developer ဖြစ်တဲ့ Ken Thompson ကလည်း UNIX system ကို C နဲ့ ပြောင်းလဲရေးသားတယ်လေ။ C ဟာ UNIX system မှာ နှစ်အကြာကြီးတည်ရှိနေခဲ့ပါတယ်။

တစ်ချိန်မှာ AT & T ကနေ UNIX system ကို တက္ကသိုလ်တွေမှာအသုံးပြုဖို့အတွက် အလကားလောက်နီးနီးနဲ့လျှုခဲ့တော့ C language ဟာ တက္ကသိုလ် ကျောင်းသားတွေနဲ့ အနီးစပ်ဆုံး language တစ်ခုဖြစ်သွားခဲ့ပါတယ်။ ယင်းအကျိုးဆက်က တက္ကသိုလ်ကျောင်းသားတွေ ကျောင်းကထွက်ပြီးအလုပ်ဝင်သွားတဲ့အချိန်မှာတော့ high level language မှာကြီးစိုးနေတဲ့ COBOL နဲ့ low level language မှာကြီးစိုးနေတဲ့ Assembly language တို့ကို သူတို့တတ်သိလာတဲ့ C နဲ့ တိုက်ဆုတ်လိုက်ပြီပေါ့။ အဲဒီအချိန်က microcomputer တွေမှာ CP/M operating system ကလွှမ်းမိုးနေဆဲပါ။ high-level language အနေနဲ့အသုံးများနေတာက BASIC language ဖြစ်ပါတယ်။

1979 ခုနှစ်မှာ Brian Kernighan နဲ့ Dennis Ritchie တို့ပူးတွဲရေးသားတဲ့ The C Programming Language စာအုပ်ကို Prentice Hall စာအုပ်တိုက်ကနေထုတ်ဝေခဲ့ပါတယ်။ ဒါပေမယ့် အဲဒီစာအုပ်မှာပါဝင်တဲ့ C language specification တွေကို standard လို့ မသတ်မှတ်နိုင်သေးပါဘူး။ ခေတ်စကားနဲ့ပြောရရင် "Classic C" လို့ပဲခေါ်ရမှာပေါ့။ တစ်ချို့ကလည်းဒီစာအုပ်ကို စာရေးသူတွေရဲ့နာမည်အဖျားဆွတ်ထည့်ပြီး K&R C လို့ ခေါ်တာလည်းရှိရဲ့။ တိုက်တိုက်ဆိုင်ဆိုင် IBM ကလည်း main frame တွေထုတ်လုပ်နေရာမှ 1981 ခုနှစ်မှာ PC လောကထဲကိုဝင်လာတယ်လေ။ IBM ကိုတုတဲ့ IBM clone တွေကလည်းပေါမှပေါပေါ့။ Microsoft ကိုယ်တိုင်က သူ့ရဲ့ပထမဆုံးအောင်မြင်လာတဲ့ Windows 3.X operating system ကို C language နဲ့ပဲရေးလိုက်တဲ့အခါမှာလူအများက C ကို လွတ်လွတ်ကျွတ်ကျွတ် လက်ခံသွားတာမဆန်းတော့ပါဘူး။ ဒီအခြေအနေတွေဟာ C language အတွက် ရေကန်အသင့်ကြာအသင့်ဖြစ်သွားစေပါတယ်။

1983 ခုနှစ်မှာ ANSI က C language definition ကို အရင်သတ်မှတ်ပြီး ISO (International Standards Organization) ကိုဆက်လက်တင်ပြခဲ့ရာမှာ 1990 ခုနှစ်ကျမှ C standard document ကိုရိုက်နှိပ်ဖြန့်ချိနိုင်ခဲ့ပါတယ်။ အဲဒီစာအုပ်ဟာ Standard C ပါပဲ။ international standard C++ document ကို 1998 ခုနှစ်မှာအပြီးသတ်ထုတ်ဝေနိုင်ခဲ့ပါတယ်။ စာဖတ်သူအနေနဲ့ C ၊ C++ တို့ရဲ့တိုးတက်ဖြစ်ပေါ်မှု timeline ကို အလွယ်တကူ လေ့လာလို့ရအောင် ဇယား (၁.၁) မှာဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

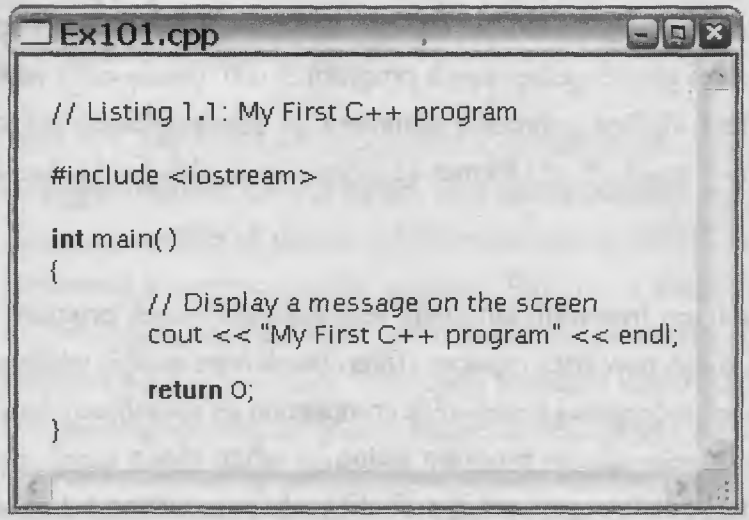
ဇယား (၁.၁) **C and C++ Timeline**

Year	Event
1970	UNIX system ကို minicomputer တွေမှာအသုံးပြုနိုင်အောင် Assembly language နဲ့ Ken Thompson ကရေးသားပါတယ်။
1972	C programming language ကို Dennis Ritchie က စတင်ဒီဇိုင်းလုပ်ပြီး Thompson က UNIX system ကို C နဲ့ အသစ်ပြင်ရေးပါတယ်။
1978	Kernighan and Ritchie တို့ နှစ်ဦးစပ်တူရေးသားတဲ့ C Programming Language စာအုပ်ကို Prentice Hall ကရိုက်နှိပ်ဖြန့်ချိပါတယ်။
1979	C language ကို အခြေခံပြီး Bjarne Stroustrup က C with Classes (1st version) ကို အဆင့်မြှင့်ရေးသားပါတယ်။
1980	CP/M 8080 microcomputer operating system တွေမှာ အသုံးပြုနိုင်တဲ့ C compiler တွေ စတင်ထွက်ရှိလာပါတယ်။
1983	C with Classes နာမည်ကို C++ လို့ အမည်ပြောင်းလဲပါတယ်။ C ကို standardize လုပ်ဖို့အတွက် ANSI ကနေစတင်စည်းဝေးပါတယ်။
1985	C++ ကိုပြင်ပမှာအသုံးပြုနိုင်အောင် Cfront interpreter ကို AT&T ကနေ ဖြန့်ချိပေးပါတယ်။ Stroustrup ရေးသားတဲ့ The C++ Programming Language စာအုပ်လည်းထွက်ရှိလာပါတယ်။ ဒီခုနှစ်မှာ MS-DOS platform မှာအသုံးပြုနိုင်တဲ့ C compiler တွေလည်းပေါ်လာပါပြီ။

1986	Cfront ကို MS-DOS မှာတင်ပေးပါတယ်။ MS-DOS C compiler တစ်ခုကို Microsoft ကနေဖြန့်ချိခဲ့ပါတယ်။
1987	Borland ကနေ ထုတ်လုပ်တဲ့ Turbo C (for MS-DOS) ဟာ တစ်ခြား version တွေထက်ပိုမိုကောင်းမွန်ပါတယ်။
1989	C++ standardization အတွက် ANSI က စတင်စည်းဝေးပြီဖြစ်ပါတယ်။
1990	C standard document ကို ANSI ကနေ ပုံနှိပ်ထုတ်ဝေခဲ့ပါတယ်။ Borland ကနေ C++ compiler (for MS-DOS) ကိုဖြန့်ချိပါတယ်။
1991	Microsoft ကနေ C compiler (for MS-DOS) ကို ဖြန့်ချိပါတယ်။
1998	ANSI နဲ့ ISO တို့ကနေ C++ standard definition document ကို အပြီးသတ် ပုံနှိပ်ထုတ်ဝေဖြန့်ချိခဲ့ပါတယ်။

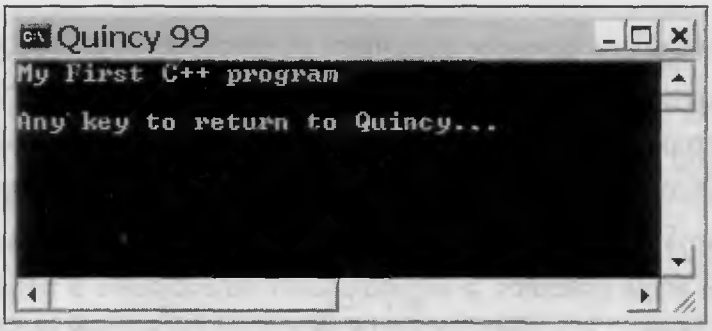
၁.၂ My First C++ Program

၁။ ပုံ (၁. ၁) မှာဖော်ပြထားတဲ့ Ex101.cpp program ဟာဆိုရင် ကျွန်တော်တို့စတင်ရေးလိုက်တဲ့ C++ program အသေးလေးပါပဲ။ program က သေးပေမယ့် run လို့ရပါတယ်။



ပုံ (၁.၁)

ဒီ program ကို Compile လုပ်ပြီး Run လိုက်တာနဲ့ ကွန်ပျူတာစကရင်မှာ My First C++ program ဆိုတဲ့စာကြောင်း ပေါ်လာတာကိုတွေ့ရပါလိမ့်မယ်။ ပုံ (၁.၂) ကိုကြည့်ပါ။



ပုံ (၁.၂)

၃။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

■ စစချင်းမှာတွေ့ရတဲ့ // Listing 1.1: My First C++ program ဆိုတဲ့ statement ရဲ့အစက double-slash characters // ကို single-line comment လို့ခေါ်ပါတယ်။ အဲဒီနောက်က စာတွေဟာ program အတွက်မှတ်ချက်တစ်ခု ထည့်ရေးထားတဲ့သဘောပါပဲ။ program ထဲမှာ

comment တွေကိုလိုအပ်တဲ့နေရာတွေမှာ ထည့်သွင်းပေးခြင်းအားဖြင့် ဒီ program ကို ဘယ်သူပဲ ဖတ်ဖတ် ရှင်းလင်းလွယ်ကူနေမှာပါ။ program ကို run တဲ့အခါမှာလည်း white space တွေလိုပဲ effect မရှိပါဘူး။ တစ်ကယ်လို့ comment မှာ စာအများကြီးရေးဖို့လိုရင် comment line တွေကို အခုလို /* */ format နဲ့ပိတ်ပြီးရေးပေးပါ။ ဒါဆိုရင်မှတ်ချက်တွေမှန်း compiler က သိပါတယ်။

■ C++ ဟာ free-form language အမျိုးအစားဖြစ်တဲ့အတွက် program ကို compile လုပ်တဲ့အခါမှာ new lines ၊ spaces ၊ tabs ၊ blank lines အစရှိတဲ့ white-space character တွေကို အပိုတွေအနေနဲ့ သတ်မှတ်ပြီး compilation မှာ အနှောင့်အယှက်မရှိ ချန်ထားခဲ့မှာ ဖြစ်ပါတယ်။ တစ်ကယ်တော့ program listing မှာ white space တွေကို အသုံးမပြုနိုင်ဘူးဆိုရင် source code တွေဟာ ဖတ်လို့ ခက်ခဲပါလိမ့်မယ်။ ဥပမာ Listing 1.1 ကို အခုလိုပြောင်းရေးရင် program ကို run လို့ရပေမယ့် program code တွေဟာ ဖတ်လို့မကောင်းတာသေချာပါတယ်။

```
#include <iostream>
int main( ) {          // Display a message on the screen
cout << "My First C++ program";    return 0; }
```

■ #include<iostream> statement ကို preprocessing directive လို့ခေါ်ပါတယ်။ current program မှာ <iostream> header ဖိုင်ထည့်ပေးဖို့ #include directive ကနေ compiler ကိုအမိန့်ပေးနေတာပါ။ console input/output အတွက်အသုံးပြုမယ့် classes ၊ functions နဲ့ global value တွေဟာ <iostream> standard library header ဖိုင်ထဲမှာ အကုန်ရှိနေပါတယ်။ ဒါကြောင့်မို့ cout ကို program မှာအသုံးပြုလို့ရတာပါ။ <iostream> မှာ angle bracket< > တွေကိုထည့်ပေးထားတဲ့အဓိပ္ပါယ်က ဒီ header ဖိုင်ကို compiler system ကနေ supply လုပ်ပေးပါလို့ ဆိုလိုပါတယ်။ တစ်ကယ်လို့ ကိုယ်ပိုင် header ဖိုင်တွေကို include လုပ်ပေးချင်တယ်ဆိုရင် bracket တွေအစား quote (" ") တွေကို အသုံးပြုရပါမယ်။ ဥပမာ #include "myheader.h" ဆိုတာမျိုးပေါ့။

■ int main( ) statement ဟာဆိုရင် main( ) function ကနေ integer(int) value တစ်ခု ကို return လုပ်ပေးပါလို့ ကြေငြာတာပါ။ main( ) function ရဲ့ အဝင်မှာ ဘာ argument မှ လက်မခံပါဘူး။

- left brace { သင်္ကေတဟာဆိုရင် main( ) function ထဲက statement block တွေကို စတင်ဖြေရှင်းတော့မယ်လို့ ကြေငြာတာပါ။

- cout << "My First C++ program"; ဆိုတဲ့ statement ဟာဆိုရင် cout object ကို အသုံးပြုပြီး My first C++ program ဆိုတဲ့ string constant တစ်ခုကို screen မှာပေါ်လာအောင် compiler ကို display လုပ်ခိုင်းတာပါ။ cout ဟာ BASIC language က PRINT statement နဲ့သဘောအတူတူပါပဲ။ မတူတာက PRINT ဟာ BASIC language ရဲ့အစိတ်အပိုင်းတစ်ခုဖြစ်ပေမယ့် cout ကျတော့ std:: prefix ကနေ compiler ကို (cout ဟာတစ်ခြား identifier တွေနဲ့ မတူကြောင်းကို) အသိပေးပြီး standard input/output library header ဖိုင်ကနေသေချာဆွဲထုတ်ဖို့ အမိန့်ပေးတာဖြစ်ပါတယ်။ cout object နဲ့ တွဲထားတဲ့ << stream insertion operator ကို သတိပြုကြည့်ပါ။ stream insertion operator << ဟာ data object တွေကို output stream ထဲမှာထည့်ပေးနိုင်တဲ့အတွက် output operator လို့ခေါ်လည်းရပါတယ်။ မြားရဲ့ direction ကနေ data object လားရာအရပ်ဝင်္ဂန်းကို ဖော်ပြနေပါတယ်။ cin ဆိုရင် input operator >> ကိုအသုံးပြုရမှာပါ။ C++ statement တစ်ခုရေးပြီးတိုင်း semicolon (;) နဲ့ပိတ်ပေးရပါမယ်။ ဒါမှ statement ဆုံးသွားတာကို compiler ကသိမှာပါ။ << endl က next line statement ကိုကူးခိုင်းတာဖြစ်ပါတယ်။ ဒါမှမဟုတ် << '\n' လို့ ရေးလည်းရပါတယ်။

- နောက်ဆုံးနားမှာရေးထားတဲ့ return 0; statement ကဆိုရင် main( ) function ကိုအလုပ်ရပ်ခိုင်းပြီး constant integer (0) value ကို operating system ဆီ return လုပ်ပေးတာပါပဲ။ right brace (}) သင်္ကေတဟာ statement block တွေကို ဖြေရှင်းလို့ပြီးသွားကြောင်းကြေငြာတာပါ။ ဒါဆိုရင် EX101.cpp program ဟာပြီးသွားပါပြီ။

၁.၃ Identifiers

C++ program တစ်ခုမှာ variables ၊ functions ၊ classes အစရှိတဲ့ element မျိုးစုံဟာပါဝင်နေပြီး element တိုင်းမှာ နာမည်တစ်ခုစီ ရှိနေကြတယ်လေ။ ဒီနာမည်တွေကို identifier လို့ ခေါ်တာပါပဲ။ လွယ်လွယ်ပြောမယ်ဆိုရင် identifiers ဆိုတာ variable name တွေပဲပေါ့။ ကျွန်တော်တို့ identifier တစ်ခုကိုသတ်မှတ်တဲ့အခါမှာ ဖော်ပြပါအချက်အလက်တွေကို လိုက်နာရပါမယ်။

- identifier တစ်ခုမှာ စာလုံးတွေ (letters) ၊ ဂဏန်းတွေ (digits) နဲ့ underscore (_) character တွေ ပါဝင်လို့ရပါတယ်။ ဒါပေမယ့် identifier ရဲ့ စဦးအက္ခရာဟာ letter သို့မဟုတ် underscore တစ်ခုပဲဖြစ်ရပါမယ်။ underscore (2) ခုတောင် ဖြစ်လို့မရပါဘူး။ ဘာပြုလို့လဲ ဆိုတော့ အဲဒီ identifier မျိုးတွေကို system အတွက်ပဲ သီးသန့်သတ်မှတ်ထားလို့ပါ။ C++ program တစ်ခုက လက်သင့်ခံတဲ့ identifier မျိုးတွေဟာ ဥပမာ salary ၊ Amount ၊ TAX ၊ GrossPay ၊ sum_it ၊ TaxRate အစရှိသည်တို့ဖြစ်ပါတယ်။ လက်သင့်မခံတဲ့ identifier မျိုးတွေကို အောက်မှာဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

4th_July	identifier ရဲ့ ရှေ့ဆုံးအက္ခရာဟာ ဂဏန်းဖြစ်လို့မရပါဘူး။
"Amount"	identifier မှာ special character (") တွေ ပါလို့မရပါဘူး။
Kount - 1	အနုတ်လက္ခဏာလည်း လက်မခံပါဘူး။
num 2	identifier မှာ space ကွက်လပ်ချန်လို့မရပါဘူး။

- letter တွေကို စာလုံးအကြီး (uppercase letter) သို့မဟုတ် စာလုံးအသေး (lowercase letter) ကြိုက်ရာနဲ့ သတ်မှတ်ပေးလို့ရပါတယ်။ ဒါပေမယ့် C++ ဟာ case-sensitive ဖြစ်တဲ့အတွက် MyFunc ၊ myFunc သို့မဟုတ် myfunc ဆိုတဲ့ identifier တွေဟာတစ်ခုနဲ့တစ်ခု မတူဘူးဆိုတာကိုသတိပြုပါ။ ဥပမာ area = pi * r * r ဆိုတဲ့ statement ကို **MyFunc = myFunc*myfunc * myfunc** လို့ပြောင်းရေးရင် သဘောအတူတူပါပဲ။

- identifier name ကို ကြိုက်သလောက် နာမည်ရှည်ရှည်ပေးလို့ ရပါတယ်။ ဒါပေမယ့် ရှေ့ဆုံးက အက္ခရာ (32) လုံးပဲ identifier နဲ့တိုက်ရိုက်အကျိုးသက်ရောက်မှာပါ။ C++ language အတွက် သီးသန့်သတ်မှတ်ထားတဲ့ C++ keyword တွေကို identifier တွေအဖြစ် အသုံးပြုလို့မရပါဘူး။

၁.၄ Keywords

C++ program တစ်ခုမှာ identifier name တွေအနေနဲ့ အသုံးပြုခွင့်မရှိတဲ့ standard reserved word တွေကို C++ keyword တွေလို့ခေါ်ပါတယ်။ နာမည်ကွက်တိမတူရင်တော့ သုံးလို့ရပါတယ်။ ဇယား (၁. ၂) မှာ C++ keyword တွေကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

ဇယား (၁. ၂) **Standard C++ Keywords**

Non-Identifiers			
asm	else	operator	throw
auto	enum	private	true
bool	explicit	protected	try
break	extern	public	typedef
case	false	register	typeid
catch	float	reinterpret_cast	typename
char	for	return	union
class	friend	short	unsigned
const	goto	signed	using
const_cast	if	sizeof	virtual
continue	inline	static	void
default	int	static_cast	volatile
delete	long	struct	wchar_t
do	mutable	switch	while
double	namespace	template	
dynamic_cast	new	this	

၁.၅ Variables

variable ဆိုတာ program ထဲက data value တွေကို ထားသိုဖို့အသုံးပြုတဲ့ storage နေရာတစ်ခုပါပဲ။

variable တိုင်းမှာ type name တစ်မျိုးစီရှိပါတယ်။ type ဆိုတာ variable ရဲ့ format နဲ့ data အမျိုးအစားကို သတ်မှတ်ပေးတာပါ။ variable တစ်ခုရဲ့တန်ဖိုးတွေကို program ထဲမှာ ကြိုက်သလိုပြောင်းလဲလို့ရပေမယ့် သူနဲ့ ပတ်သက်တဲ့ data type ကတော့မပြောင်းပါဘူး။ type declaration မှာ ကြေငြာထားတဲ့အတိုင်း အမြဲရှိနေမှာပါပဲ။ C++ မှာပါဝင်တဲ့ data type တွေက bool ၊ char ၊ wchar_t ၊ int ၊ float နဲ့ double တို့ဖြစ်ပါတယ်။ variable type တွေကိုထပ်ခွဲဦးမယ်ဆိုရင် unsigned ၊ long ၊ short ၊ long double တို့ဆိုတာရှိပါသေးတယ်။

The bool Type

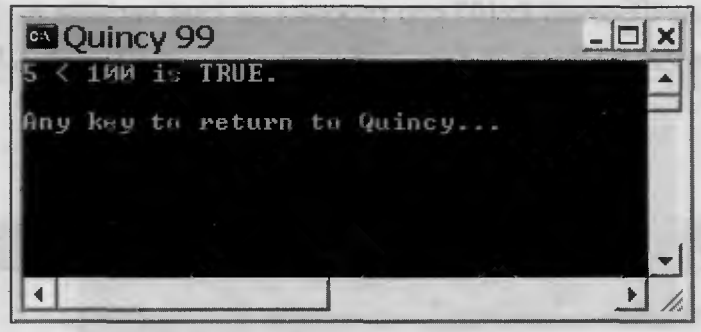
၁။ bool variable ဆိုတာ true သို့မဟုတ် false ဆိုတဲ့ value (2) မျိုးကိုအသုံးပြုတဲ့ boolean (logical) type အမျိုးအစားပါ။ arithmetic expression တစ်ခုမှာ bool variable ကို အသုံးပြုမယ်ဆိုရင် expression က true ဖြစ်တဲ့အခါမှာ integer value 1 ကို return လုပ်ပေးပါလိမ့်မယ်။ false ဆိုရင် zero ကို return လုပ်မှာပါ။ တစ်ကယ်လို့ integer တစ်ခုကို bool ဖြစ်အောင်ပြောင်းပေးမယ်ဆိုရင် integer 0 ဟာ false ဖြစ်ပြီး integer nonzero က true ဖြစ်မှာပါပဲ။ ပုံ (၁. ၃) မှာဖော်ပြထားတဲ့ Ex102.cpp program ဟာဆိုရင် bool variable အသုံးပြုနည်းကိုရေးထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex102.cpp

```
// Listing 1.2: Using a bool variable

#include <iostream>

int main( )
{
        bool  flag;   // bool variable

        flag = 5 < 100;    // Set a logical value to flag
        if (flag)
                cout << "5 < 100 is TRUE." << endl;

        return 0;
}
```

ပုံ (၁. ၃)

program ကို run လိုက်တာနဲ့ကွန်ပျူတာစကရင်မှာ 5 < 100 is TRUE. ဆိုတဲ့စာကြောင်း ပေါ်လာတာကိုတွေ့ရပါလိမ့်မယ်။ ပုံ (၁.၄) ကိုကြည့်ပါ။



ပုံ (၁.၄)

ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- bool flag; ဆိုတဲ့ statement ဟာ identifier flag ကို type bool လို့ ကြေငြာတာပါ။
- flag = 5 < 100; ဆိုတဲ့ statement ကို logical assignment statement လို့ခေါ်ပါတယ်။ 5 ဟာ 100 ထက်ငယ်တာမို့ flag တန်ဖိုးဟာ True ဖြစ်ပါတယ်။
- if (flag) ဆိုတာ if (True) လားလို့မေးတာပါ။ True ဖြစ်တဲ့အတွက် cout << "5 < 100 is TRUE." ဆိုတဲ့ statement ကို execute လုပ်ပြီး console (screen) မှာ 5 < 100 is TRUE. ဆိုတဲ့ string constant ကို display လုပ်ပြတာပါ။ တစ်ကယ်လို့ flag = 5 > 100 ဆိုရင် False ဖြစ်တဲ့အတွက် if (flag) နောက်က statement ကိုမဖြေရှင်းပဲ ကျော်သွားမှာပါ။
- နောက်ဆုံးမှာရေးထားတဲ့ return 0; statement ကိုတွေ့ရင် main( ) function ရဲ့အလုပ်ပြီးသွားပြီမို့ constant integer (0) value ကို operating system ဆီ return လုပ်ပေးပါလိမ့်မယ်။ ပြီးတော့ရင် right brace (}) သင်္ကေတကနေ program ပြီးဆုံးကြောင်းကို ကြေငြာပါတယ်။

The char Type

၁။ char variable ဆိုတာ 8-bit bytes ASCII character set ထဲက character value တွေကို ဆိုလိုပါတယ်။ ဥပမာ ch ဆိုတဲ့ identifier တစ်ခုကို type char နဲ့ declare လုပ်မယ်ဆိုရင် char ch; လို့ရေးရင် ရပါတယ်။ ပုံ (၁. ၅) မှာဖော်ပြထားတဲ့ Ex103.cpp program မှာ char variable အသုံးပြုနည်းကိုရေးထားပါ တယ်။ Ex103.cpp program ကို run လိုက်တာနဲ့ ကွန်ပျူတာစကရင်မှာ C++ ဆိုတဲ့စာကြောင်းပေါ်လာမှာပါပဲ။ ပုံ (၁. ၅) ကိုကြည့်ပါ။

Ex103.cpp

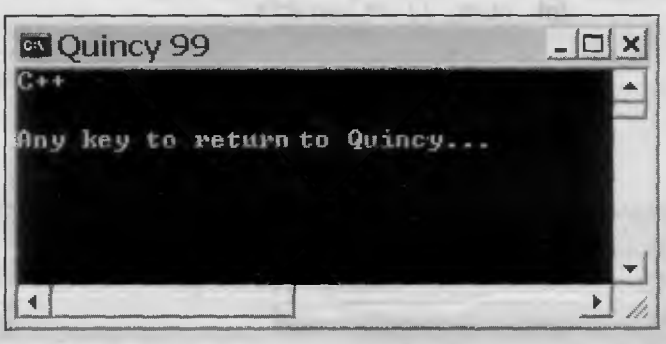
```
// Listing 1.3: Using a character variable

#include <iostream>
int main( )
{
        char  ch;        // char variable

        ch = 'C';        // assign  'C' to ch
        cout << ch;      // display 'C'

        ch = '+';        // assign  '+' to ch
        cout << ch << ch << endl;    // display '++'

        return 0;
}
```

ပုံ (၁. ၅)

၂။ ပုံ (၁. ၆) မှာ Ex103.cpp program ကို run ပြထားပါတယ်။ ကွန်ပျူတာစကရင်မှာ C++ ဆိုတဲ့စာ ပေါ်လာတာကို တွေ့ရပါလိမ့်မယ်။ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ identifier ch ကို type char လို့ကြေငြာပြီး ch ကို character 'C' နဲ့ assign လုပ်ပေးပါတယ်။ ပြီးတော့ရင် ch ရဲ့ value ဖြစ်တဲ့ C စာလုံးကို screen မှာ display လုပ်ပြပါ လိမ့်မယ်။

- နောက်တစ်ခါ ch ကို character constant '+' နဲ့ assign လုပ်ပေးပါတယ်။ ပြီးတော့ရင် cout << ch << ch; ဆိုတဲ့ statement ကနေ C စာလုံးနောက်မှာ '+' သင်္ကေတ (2) ခု ကိုကပ်ပြီး display လုပ်ပြပါလိမ့်မယ်။ ဒါကြောင့်မို့ကွန်ပျူတာမှာ C++ ဆိုတဲ့စာကြောင်းပေါ်လာတာပါပဲ။ ပုံ (၁.၆) ကိုကြည့်ပါ။



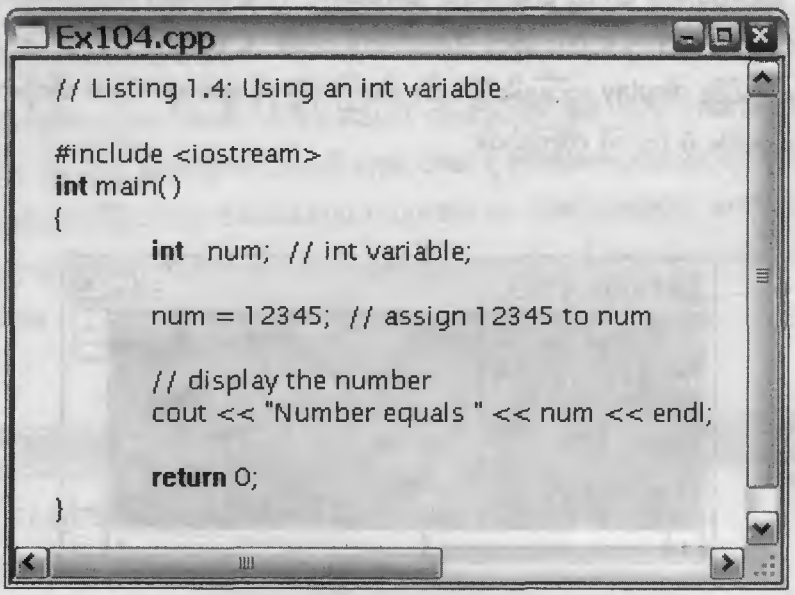
ပုံ (၁.၆)

The wchar_t Type

C++ compiler မှာ wide character class library ရှိနေမယ်ဆိုရင် wout object ကို အသုံးပြုပြီး 16-bits wide char data type ကနေ character တွေယူပြီး stream output လုပ်ပေးနိုင်ပါတယ်။

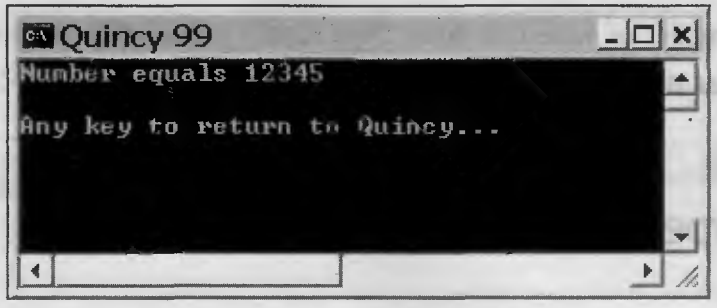
The int Type

၁။ int type specifier တွေမှာ plain signed integer ဥပမာ int counter; ဆိုတာမျိုးရှိသလို signed int၊ unsigned int၊ long int၊ short int စသည်ဖြင့် အမျိုးမျိုးရှိနေကြပါတယ်။ int ကို အသုံးပြုမယ်ဆိုရင် range က -32768 to 32767 ဖြစ်ပါတယ်။ short int နဲ့အတူတူပါပဲ။ unsigned int ဆိုရင် range: 0 to 65535 ဖြစ်ပါတယ်။ signed long int ရဲ့ range ဟာ -2147483648 to 2147483647 ဖြစ်ပြီး unsigned long int ရဲ့ range က 0 to 4294967295 ပါ။ ပုံ (၁.၇) မှာဖော်ပြထားတဲ့ Ex104.cpp program မှာ int variable သုံးနည်းကို ဖော်ပြထားပါတယ်။



ပုံ (၁. ၇)

၂။ Ex104.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာစကရင်မှာ Number equals 12345 လို့ ပေါ်လာပါလိမ့်မယ်။ ပုံ (၁. ၈) ကိုကြည့်ပါ။



ပုံ (၁. ၈)

၃။ ဒီ program ကိုလေ့လာကြည့်ရင်

- စစချင်းမှာ identifier num ကို type int လို့ကြေငြာပါတယ်။ နောက်ပြီး num ကို 12345 ဂဏန်းတန်ဖိုးနဲ့ assign လုပ်ပေးပြီး num ရဲ့ value ကို string constant တစ်ခုနဲ့တွဲပြီးတော့ display လုပ်ပြခိုင်းတာပါပဲ။

The Floating-Point Numbers

C++ မှာ ဒဿမဂဏန်းအမျိုးအစား (floating-point number) (3) မျိုးကိုအသုံးပြုနိုင်ပါတယ်။ အဲဒီ (3) မျိုးက float ၊ double နဲ့ long double တို့ပါပဲ။ float ရဲ့ range ဟာ 3.4E-38 to 3.4E1038 ၊ double ရဲ့ range က 1.7E-308 to 1.7E308 နဲ့ long double ရဲ့ range ဟာ 3.4E-4932 to 1.1E4932 တို့ ဖြစ်ပါတယ်။ ဒီသတ်မှတ်ချက်တွေက ပုံသေမဟုတ်ပါဘူး။ compiler အမျိုးအစားပေါ်မူတည်ပါတယ်။ ပုံ (၁.၉) မှာ ပြထားတဲ့ Ex105.cpp program ဟာ float အသုံးပြုနည်းကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex105.cpp

```
// Listing 1.5: Using a float variable

#include <iostream>
int main()
{
        float fnum; // float variable

        fnum = 1.234567; // assign a value to fnum

        // display the number
        cout << "fnum equals " << fnum << endl;

        return 0;
}
```

ပုံ (၁.၉)

program ကို run လိုက်ရင် ကွန်ပျူတာစကရင်မှာ fnum equals 1.23457 လို့ ပေါ်လာပါလိမ့်မယ်။ ပုံ (၁.၁၀) ကိုကြည့်ပါ။ default အနေနဲ့ ဒဿမ (5) လုံးစာပဲဖော်ပြပါတယ်။

Quincy 99

```
fnum equals 1.23457

Any key to return to Quincy...
```

ပုံ (၁.၁၀)

၁.၆ Constants

C++ program တစ်ခုမှာ တန်ဖိုးမပြောင်းလဲပဲ ကိန်းသေဖြစ်နေတဲ့ value မျိုးတွေကို constants လို့ ခေါ်ပါတယ်။ constant ဘယ်နှစ်မျိုးရှိလဲဆိုတော့ (5) မျိုးပါ။ (၁) Character constants (၂) Integer constants (၃) Floating-point constants နဲ့ (၅) Address constant တို့ဖြစ်ကြပါတယ်။

Character Constants

Character constants ဆိုတာ single code (' ') တွေနဲ့ပိတ်ထားတဲ့ char value တွေကိုဆိုလိုပါတယ်။ တစ်ကယ်တော့ character ဟာ letter ဖြစ်ဖြစ် digit ဖြစ်ဖြစ် ၊ blank ကိုတောင် character လို့ခေါ်လို့ရတယ်လေ။ character တွေမှာ numeric value တွေရှိကြပါတယ်။ ဒီတန်ဖိုးတွေကို ASCII (American Standard Code for Intermation Interchange) value တွေအနေနဲ့သတ်မှတ်မှာပါ။ ဥပမာ

```
ch1 = 'A'        // ASCII char constant
ch2 = '\x2f'     // char constant expressed as hex value
ch3 = '\013'     // char constant expressed as octal value
```

ဒီဥပမာမှာ backslash \ သင်္ကေတတွေကို escape sequences လို့ ခေါ်ပါတယ်။ ဒီသင်္ကေတပါလာရင် character constant မှာ ထူးခြားတဲ့ ရည်ညွှန်းချက်တစ်ခု ရှိနေပါပြီ။ \a လို့ပါလာရင် audible bell character ဖြစ်သလို \n ဆိုရင် output data အတွက် newline character ဖြစ်ပါတယ်။ ဇယား (၁.၃) မှာ constant escape sequence တွေကိုဖေါ်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

ဇယား (၁.၃) **Constant Escape Sequences**

Escape Sequence	Description
\'	Single quote
\"	Double quote
\\	Backslash
\0	Null (zero) character

\0nnn	Octal number (nnn)
\a	Audible bell character
\b	Backspace
\f	Formfeed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\x	Hexadecimal number (nnn)
\?	question mark

Integer Constants

Integer constants ဆိုတာ type name တွေဖြစ်တဲ့ long သို့မဟုတ် short ၊ signed သို့မဟုတ် unsigned fixed value တွေကို ဆိုလိုတာပါ။ integer constant မှာပဲ (3) မျိုး ထပ်ခွဲထားပါသေးတယ်။ (၁) Decimal constants (base 10) (၂) Octal constants (base 8) နဲ့ (၃) Hexadecimal constants (base 16) တို့ပါပဲ။ decimal constant တစ်ခုဟာဆိုရင် သုညကနေ (9) ထဲက ဂဏန်းတွေနဲ့စုပေါင်းဖြစ်နေတာပါ။ constant မှာ ဂဏန်း (2) လုံး သို့မဟုတ် (2) လုံးထက် ပိုပါနေမယ်ဆိုရင် ထိပ်ဆုံးဂဏန်းဟာ သုညဖြစ်လို့မရပါဘူး။ နောက်ပြီး constant value ဟာ သတ်မှတ်ထားတဲ့ lower bound နဲ့ upper bound ကြားက တန်ဖိုးတွေပဲ ဖြစ်ရပါမယ်။ C++ program က လက်သင့်ခံတဲ့ decimal constant တွေက ဥပမာ 0 ၊ 1234 ၊ 32700 တို့ပါပဲ။ အောက်ပါ constant တွေကတော့ decimal constant တွေ မဟုတ်ကြပါဘူး။

23,456	constant မှာ comma (,) ပါလို့မရပါဘူး။
54.68	ဒဿမလည်း ပါလို့မရပါဘူး။
32 591	space ကွက်လပ်ချန်လို့မရပါဘူး။
0987	စဦးအက္ခရာဟာ သုညဖြစ်လို့မရပါဘူး။
956705	constant value ဟာ upper bound limit ကိုကျော်နေပါတယ်၊ ဒါဆိုရင် decimal constant မဖြစ်ပါဘူး။

Octal constant တွေက သုညနဲ့ (7) ထဲကဂဏန်းတွေနဲ့ စုပေါင်းဖြစ်နေတာပါ။ octal constant မှန်းသိအောင်လို့ ထိပ်ဆုံးဂဏန်းကို သုညအမြဲထားပေးရပါမယ်။ ဥပမာ 0775 ဖြစ်ပါတယ်။

Hexadecimal constant တွေကျတော့ သုညကနေ (9) ထဲက ဂဏန်းတွေနဲ့အတူ (a) ကနေ (f) ထဲက letter တွေနဲ့ စုပေါင်းဖြစ်နေတာပါ။ hexadecimal မှန်းသိအောင်လို့ ထိပ်ဆုံးအက္ခရာ (2) ခုကို 0x သို့မဟုတ် 0X အမြဲလုပ်ပေးရပါမယ်။ ဥပမာ 0x54fa ဖြစ်ပါတယ်။

Floating-Point Constants

Floating-point constants ဆိုတာ base 10 ကို အခြေခံတဲ့ real number တွေကို ဆိုလိုပါတယ်။ ဒီ constant တွေကို ဒဿမဂဏန်း (floating point) အနေနဲ့ဖြစ်စေ သို့မဟုတ် ထပ်ကိန်း (exponent) ပုံစံနဲ့ဖြစ်ဖြစ် သတ်မှတ်လို့ရပါတယ်။ ဥပမာ 1234.56၊ 2E-8၊ 2.55E+38 တို့ ဖြစ်ပါတယ်။ အောက်ပါ constant တွေကိုတော့ C++ က floating-point constants တွေလို့ မသတ်မှတ်ပါဘူး။

1234	constant မှာ decimal point ပျောက်နေပါတယ်။ ဒီဥစ္စာမရပါဘူး။
15E+2.5	ထပ်ကိန်းဟာ ကိန်းပြည့်ဂဏန်းဖြစ်ရပါမယ်။
12e 5	space ကွက်လပ်ချန်လို့မရပါဘူး။

String Constants

String constants ဆိုတာ double quotes (" ") ထဲမှာ ရေးထားတဲ့အက္ခရာတွေကို ဆိုလိုတာပါ။ ဒီအက္ခရာတွေထဲမှာ blank ပါလည်းရပါတယ်။ အက္ခရာအရေအတွက် ဘယ်လောက်ပါရမယ်လို့ မကန့်သတ်ပါဘူး။ အောက်ဖော်ပြပါ string constant တွေကို C++ program က လက်ခံပါတယ်။

"Hello C++" **"MyanMar"** **"25/10/03"**

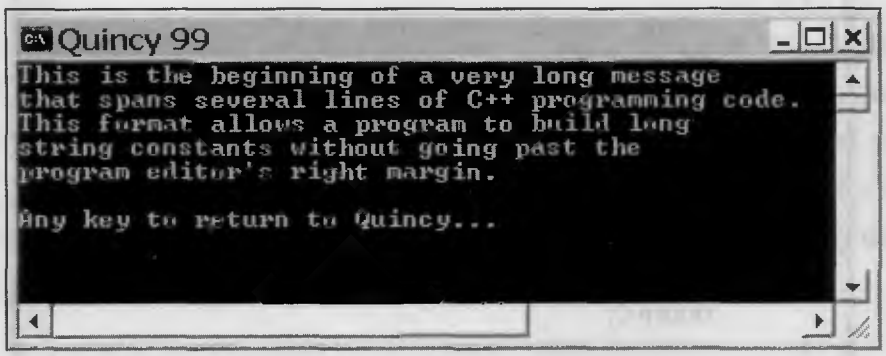
၁။ ပုံ (၁. ၁၁) မှာဖော်ပြထားတဲ့ Ex106.cpp program ဟာဆိုရင် string အရှည်ကြီးရေးတာကို string concatenation နည်းနဲ့ဆက်ပေးထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex106.cpp

```
// Listing 1.6: Concatenated string constants
#include <iostream>

int main()
{
        cout << "This is the beginning of a very long message\n"
                "that spans several lines of C++ programming code.\n"
                "This format allows a program to build long\n"
                "string constants without going past the\n"
                "program editor's right margin.\n";

        return 0;
}
```

ပုံ (၁.၁၁)

၂။ Ex106.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ အခုလိုပေါ်လာတာကိုတွေ့ရပါလိမ့်မယ်။ ပုံ (၁.၁၂) ကိုကြည့်ပါ။



ပုံ (၁.၁၂)

၃။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- string constant အရှည်ကြီးကို newline escape sequence \n အသုံးပြုပြီး ခွဲရေးခြင်းအားဖြင့် စာပိုဒ်အရှည်ကြီးတစ်ခုကို ဘောင်မကျော်အောင် display လုပ်ပြခိုင်းလို့ရပါတယ်။

Enumerated Constants

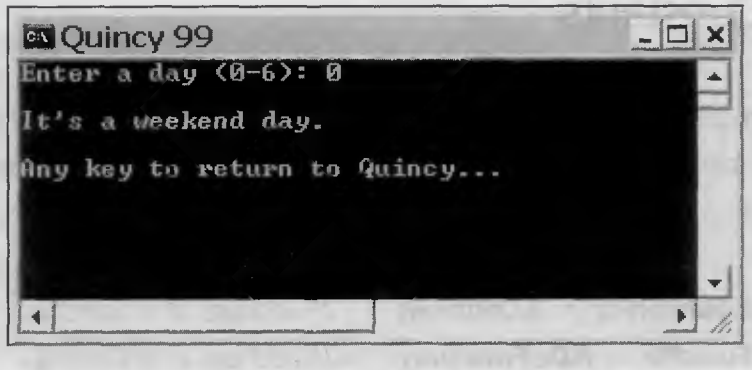
၁။ program တစ်ခုမှာ သတ်မှတ်ထားတဲ့ data set တစ်ခုကို type တစ်ခုအနေနဲ့ သတ်မှတ်ပေးချင်ရင် enumerated constant ကိုအသုံးပြုလို့ရပါတယ်။ ဥပမာ COLOR ကို enumeration တစ်ခုလို့ကြေငြာပေးပြီး (RED, BLUE, GREEN,WHITE, BLACK) ဆိုတဲ့အရောင် (5) မျိုးနဲ့သတ်မှတ်ပေးချင်ရင် enum COLOR {RED, BLUE, GREEN,WHITE, BLACK}; ဆိုတဲ့ statement ကိုရေးရင်ရပါတယ်။ COLOR ဆိုတာက enumeration type name ပါ။ RED အတွက် symbolic constant ကို 0 လို့သတ်မှတ်ပါတယ်။ ကျန်တာတွေကို 1, 2, 3, 4 လို့ အစီအစဉ်လိုက် သတ်မှတ်ပေးသွားမှာပါ။ ဒါမှမဟုတ် enum COLOR {RED=50, BLUE, GREEN=500,WHITE, BLACK}; လို့ကြေငြာပေးမယ်ဆိုရင် BLUE = 51 ၊ WHITE = 501 နဲ့ BLACK = 502 လို့ enumeration type က ပြောင်းလဲသတ်မှတ်ပေးသွားမှာပါ။ ပုံ (၁. ၁၃) မှာဖော်ပြထားတဲ့ Ex107.cpp program ကိုလေ့လာကြည့်ပါ။

Ex107.cpp

```
// Listing 1.7: Using an enumerated constant
#include <iostream>

int main()
{
        enum   days {Sun, Mon, Tue, Wed, Thu, Fri, Sat};
        int   choice;

        cout << "Enter a day (0-6): ";
        cin >> choice;
        if (choice == Sun || choice == Sat)
                cout << "\nIt's a weekend day.\n";
        else
                cout << "\nIt's a weekday.\n";
        return 0;
}
```

ပုံ (၁. ၁၃)

၂။ အခုနေ program ကို run လိုက်ပြီး Enter a day (0-6) : လို့ prompt တစ်ခုပေါ်လာတဲ့အခါမှာ 0 ကိုရိုက်ထည့်ပြီး ENTER နှိပ်လိုက်တာနဲ့ It's a weekend day. လို့ပေါ်လာပါလိမ့်မယ်။ ပုံ (၁. ၁၄) ကိုကြည့်ပါ



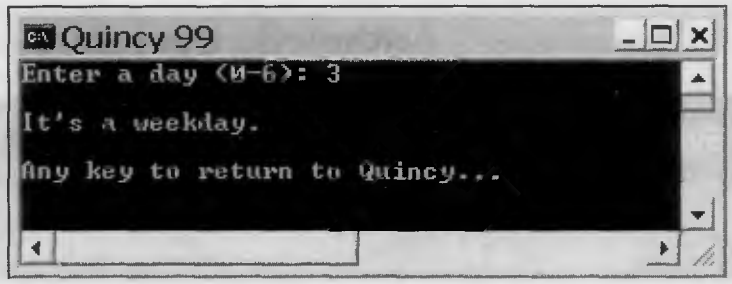
ပုံ (၁. ၁၄)

ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း identifier days ကို type enum လို့ကြေငြာပြီး {Sun, Mon,Tue,Wed,Thu, Fri, Sat } တို့နဲ့ assign လုပ်ပေးပါတယ်။ identifier choice ကို type int လို့ကြေငြာပါတယ်။

- cout << "Enter a day (0-6) : "; ဆိုတဲ့ statement ကနေ choice အတွက် data ထည့်ပေးဖို့ prompt လုပ်ပေးပါတယ်။ cin >> choice; ကနေ chice အတွက် data ကို 0 လို့ရိုက်ထည့်ပေးမယ်ဆိုပါစို့။

- ဒါဆိုရင် if (choice == Sun || choice == Sat) statement ကနေ choice ဟာ Sunday (=0) သို့မဟုတ် Saturday (=6) လားဆိုတာကိုစီစစ်ပါတယ်။ 0 ဖြစ်တဲ့အတွက် if ( ) အောက်တည့်တည့်က cout << "\nIt's a weekend day.\n"; ဆိုတဲ့ statement ကို execute လုပ်မှာဖြစ်ပါတယ်။ ဒီတော့ ကွန်ပျူတာ It's a weekend day. လို့ပေါ်လာပြီပေါ့။ တစ်ကယ်လို့ choice အတွက် data ကို 3 ထည့်ပေးရင် It's a weekday. လို့ display လုပ်ပြမှာပါ။ ပုံ (၁. ၁၅) ကိုကြည့်ပါ။



ပုံ (၁. ၁၅)

Address Constants

C++ program တွေမှာ pointer တွေကိုအသုံးပြုမယ်ဆိုရင် Address constant တွေကို အသုံးပြုရတော့မှာပါ။ variables ၊ function တွေမှာ memory address တွေ ရှိပါတယ်။ ဒီ address တွေကို C++ program မှာ အောက်ပါအတိုင်း reference လုပ်လို့ရပါတယ်။

```
CounterPtr = &Counter;      // address of a variable
FuncPtr = &DoFunction;      // address of a function
```

၁.၇ Expressions

expression ဆိုတာ တစ်သီးတစ်ခြားစီရှိနေတဲ့ constant တွေ ၊ variable တွေ ၊ array element တွေ ၊ function call တွေကို operator တွေနဲ့ဆက်စပ်ထားတာပါပဲ။ expression တစ်ခုကိုဖြေရှင်းလိုက်ရင် တန်ဖိုးတစ်ခုကို return လုပ်ပေးမှာပါ။ C++ expression တွေကို နမူနာဖော်ပြရမယ်ဆိုရင် count * 10 နဲ့ 1.8*(degC+32) ဆိုတာမျိုးပေါ့။

Arithmetic Operators

ကောင်းပြီ ၊ ဒါဆိုရင် operator တွေ ဘယ်နှစ်မျိုးရှိလဲ။ arithmetic operators ၊ unary operators၊ relational and conditional operators အစရှိသည်ဖြင့် အမျိုးအစားတွေ တစ်ပုံကြီးရှိတာပေါ့။ အဲဒါတွေသုံးရင် expression ဆိုတာဖြစ်လာတာပါပဲ။ C++ မှာပါဝင်တဲ့ arithmetic operator တွေကို ဇယား (၁. ၄) မှာဖော်ပြထားပါတယ်။ ဒီ operator တွေကိုအသုံးပြုပြီး တွက်တာချက်တာတွေကိုပြုလုပ်ရမှာပါ။

ဇယား (၁. ၄) **Arithmetic Operators**

Symbol	Description
+	Unary plus

-	Unary minus
*	Multiplication
/	Division
%	Modulus
+	Addition
-	Subtraction

၁.၈ Assignments

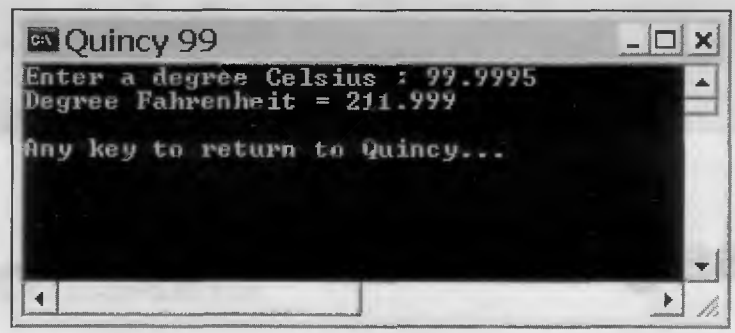
၁။ Assignment statement တစ်ခုဟာဆိုရင် expression တစ်ခုကိုဖြေရှင်းလိုက်လို့ ရရှိလာတဲ့ return value ကို variable တစ်ခုနဲ့ assign လုပ်ပေးမှာပါ။ ပုံ (၁. ၁၆) မှာဖော်ပြထားတဲ့ Ex108.cpp program ဟာ Celsius degree တစ်ခုကို 1.8*C+32; ဆိုတဲ့ expression ကို အသုံးပြုပြီး degree Fahrenheit ကိုပြောင်းလဲပေးတဲ့ program ဖြစ်ပါတယ်။

Ex108.cpp

```
// Listing 1.8: Assigning an expression

#include <iostream>
int main()
{
        float cel, fah;

        cout << "Enter a degree Celsius : ";
        cin >> cel;
        fah = 1.8*cel + 32;
        cout << "Degree Fahrenheit = " << fah << endl;

        return 0;
}
```

ပုံ (၁. ၁၆)

၂။ Ex108.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာစကရင်မှာ Enter a degree Celsius : လို့ prompt တစ်ခုပေါ်လာပါလိမ့်မယ်။ 99.9995 ကိုရိုက်ထည့်ပြီး ENTER နှိပ်လိုက်တာနဲ့ Degree Fahrenheit = 211.999 လို့ပေါ်လာပါပြီ။ ပုံ (၁. ၁၇) ကိုကြည့်ပါ။



ပုံ (၁. ၁၇)

၃။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း identifier cel နဲ့ fah တို့ကို type float လို့ကြေငြာပြီး Enter a degree Celsius: ဆိုတဲ့စာကြောင်း screen မှာပေါ်လာအောင် cout << "Enter a degree Celsius" ; ကနေလုပ်ပေးပါတယ်။

- cin >> cel; ဆိုတဲ့ statement ကိုရေးထားတာကြောင့် cel အတွက် data ကို user က keyboard ကနေရိုက်ထည့်ပေးရမှာပါ။ cel အတွက် dataကို 99.9995 ကိုရိုက်ထည့်ပေးပါမယ်။

- cel = 99.9995 လို့ program က သိသွားပြီဆိုတော့ fah =1.8*cel+32 = 1.8*99.9995 + 32 = 211.999 ဆိုတဲ့အဖြေကို ပုံသေနည်းသုံးပြီးတွက်ယူပါတယ်။

- အဖြေကိုထုတ်ပေးဖို့အတွက် cout >> "Degree Fahrenheit = " >> fah >> endl; ဆိုတဲ့ statement ကနေ ကွန်ပျူတာမှာ Degree Fahrenheit = 211.999 ဆိုတဲ့စာကြောင်းပေါ်လာအောင်လုပ်ပေးမှာပါ။ << endl expression ဟာ next statement line ကိုကူးခိုင်းဖို့အတွက် ထည့်ရေးထားတာပါပဲ။

Logical and Relational Operators

C++ language မှာ ပါဝင်တဲ့ relational and logical operator တွေကို ဇယား (၁. ၅) မှာဖော်ပြထားပါတယ်။ ဒီ operator တွေကိုအသုံးပြုပြီး expression (2) ခုကို နှိုင်းယှဉ်ခိုင်းမယ်။ နောက်ပြီး true (or false) value ကို return လုပ်ခိုင်းလို့ရပါတယ်။

ဇယား (၁. ၅) **Logical and Relational Operators**

Symbol	Description
&&	Logical AND
||	Logical OR
!	Unary NOT
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

relational and logical operator တွေအသုံးချနည်းကို ပုံ (၁. ၁၈) က C++ program မှာတင်ပြထားပါတယ်။ ဒီ program က တြိဂံတစ်ခုရဲ့ ဧရိယာရှာတဲ့ program ပါ။ တြိဂံတစ်ခုရဲ့ဧရိယာကိုမရှာခင်မှာ ပေးထားတဲ့ data တွေနဲ့ တြိဂံဖြစ်မဖြစ်ဆိုတာကို အရင်စစ်ဆေးရပါမယ်။ တြိဂံဖြစ်မှ ဧရိယာကိုရှာလို့ရမှာပါ။ ဒါဖြင့် ဘယ်လိုစစ်ရမလဲ။ တြိဂံတစ်ခုရဲ့အနားနှစ်ဖက်ပေါင်းခြင်းဟာ တတိယအနားထက် အမြဲကြီးရမယ်မဟုတ်လား။ အဲဒါမဖြစ်ရင် တြိဂံမဖြစ်ပါဘူး။ အခုစစ်ဆေးနည်းမှာ relational and logical operator တွေရဲ့အသုံးကျပုံကို Ex109.cpp program မှာတွေ့ပါလိမ့်မယ် ၊ လေ့လာကြည့်ပါ။

Ex109.cpp

```
// Listing 1.9: Using the relational and logical operators

#include <iostream>
int main()
{
        double   a,b,c,s,area;

        cout << "Enter three sides of triangle\n";
        cin >> a >> b >> c;
        if ((a+b>c) && (b+c>a) && (c+a>b))
        {
              s = (a+b+c)/2;
              area = sqrt(s*(s-a)*(s-b)*(s-c));
              cout << "\nArea of triangle = " << area << endl;
        }
        else
              cout << "\nTriangle can't be formed";

        return 0;
}
```

ပုံ (၁. ၁၈)

၂။ အခုနေ program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာစကရင်မှာ Enter three sides of triangle လို့ prompt တစ်ခုပေါ်လာပါလိမ့်မယ်။ 3 <space> 4 <space> 5 ကိုရိုက်ထည့်ပြီး ENTER နှိပ်လိုက်တာနဲ့ Area of triangle = 6 ဆိုတဲ့ အဖြေကိုတွက်ပေးပါလိမ့်မယ်။ ပုံ (၁. ၁၉) ကိုကြည့်ပါ။

Quincy 99

```
Enter three sides of triangle
3  4  5

Area of triangle = 6

Any key to return to Quincy...
```

ပုံ (၁. ၁၉)

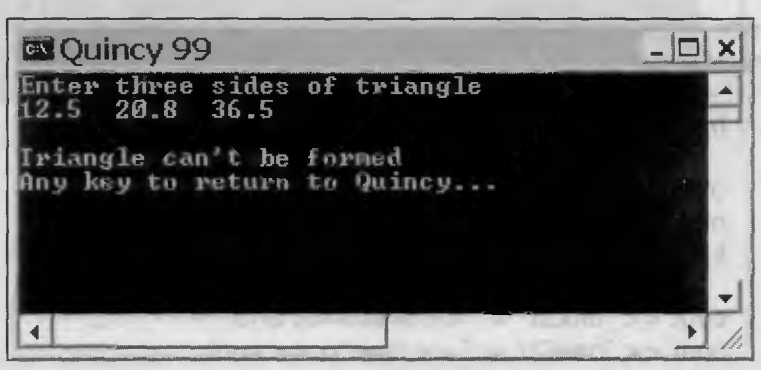
၃။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း program မှာအသုံးပြုမယ့် a ၊ b ၊ c ၊ s ၊ area ဆိုတဲ့ identifier တွေအတွက် data type ကို double လို့ကြေငြာပေးပါတယ်။

- Enter three sides of a triangle ဆိုတဲ့ prompt ကို display မှာပေါ်ခိုင်းပြီး a ၊ b ၊ c တန်ဖိုးတွေကို (space ခြားပြီး) keyboard ကနေရိုက်ထည့်ပေးရပါမယ်။ a ၊ b ၊ c တို့ရဲ့ တန်ဖိုးတွေကို a=3 ၊ b=4 နဲ့ c=5 လို့ ရိုက်ထည့်မယ်ဆိုပါစို့။

- ဒါဆိုရင် ပေးထားတဲ့ data တွေနဲ့တြိဂံဖြစ်မဖြစ်ဆိုတာကို relational operator(>) နဲ့ logical operator(&&) နှစ်ခုကိုအသုံးပြုပြီး စစ်ဆေးပါတယ်။ ဆိုလိုတာက (a = 3) + (b= 4) ဟာ (c = 5) ထက်ကြီးမယ်။ (b = 4) +(c = 5) က (a=3) ထက်ကြီးမယ်။ (c = 5) + (a = 3) က (b=4) ထက်ကြီးမယ်ဆိုရင် s နဲ့ area တို့ကိုဆက်တွက်ပြီး Area of triangle = 6 လို့ display မှာပေါ်ပေးပါလိမ့်မယ်။ sqrt( ) function ဟာဆိုရင် အပေါင်းလက္ခဏာကိန်းတစ်ခုရဲ့နှစ်ထပ်ကိန်းရင်းကိုရှာပေးနိုင်တဲ့ math function တစ်ခုဖြစ်ပါတယ်။

- တစ်ကယ်လို့ပေးထားတဲ့ data တွေနဲ့တြိဂံမဖြစ်ဘူးဆိုရင် Triangle can't be formed ဆိုတဲ့ သတိပေးချက်ကို ကွန်ပျူတာက ကြေငြာပေးပါလိမ့်မယ်။ ဥပမာ a = 3 ၊ b= 4 နဲ့ c = 9 ဆိုတဲ့ data တွေကိုထည့်ပြီး run ကြည့်ပါ။ အဖြေပေါ်လာပါလိမ့်မယ်။ ပုံ (၁. ၂၀) ကိုကြည့်ပါ။



ပုံ (၁. ၂၀)

Increment and Decrement Operators

C++ မှာ ပါဝင်တဲ့ Unary operator တွေကို ဇယား (၁. ၆) မှာဖော်ပြထားပါတယ်။ ကျွန်တော်တို့ ဒီ operator တွေကိုအသုံးပြုပြီး variable တွေရဲ့တန်ဖိုးတွေကို အတိုးအလျော့လုပ်ပေးလို့ရပါတယ်။

ဇယား (၁. ၆) **Unary Operators**

Symbol	Description
++	Increment operator
--	Decrement operator

၁။ ကျွန်တော်တို့ ပုံ (၁. ၂၁) မှာဖော်ပြထားတဲ့ Ex1010.cpp program ကို run လိုက်မယ်ဆိုရင် Unary operator တွေအကြောင်းကိုသိရပါလိမ့်မယ် ၊ လေ့လာကြည့်ပါ။

Ex1010.cpp

```
// Listing 1.10: Using increment and decrement operators

#include <iostream>
int main()
{
        int  kounter, oldCtr, newCtr;

        oldCtr = 1234;          // oldCtr = 1234
        newCtr = ++oldCtr;      // newCtr = 1235, oldCtr = 1235
        kounter = newCtr--;     // kounter = 1235, newCtr = 1234

        cout << "oldCtr  = " << oldCtr << endl;
        cout << "newCtr  = " << newCtr << endl;
        cout << "kounter = " << kounter << endl;
        return 0;
}
```

ပုံ (၁. ၂၁)

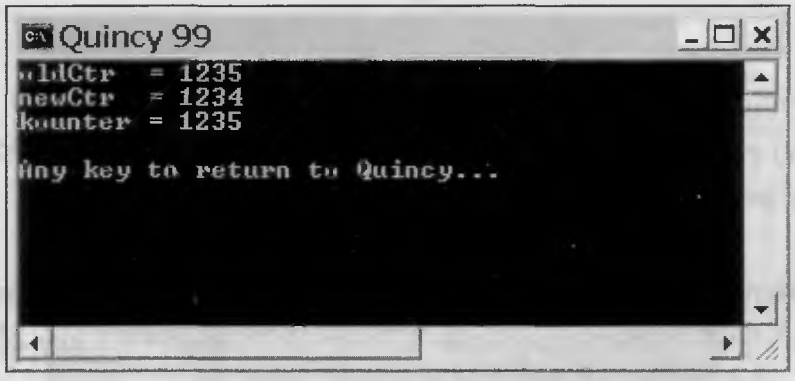
program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း program မှာအသုံးပြုမယ့် kounter ၊ oldCtr နဲ့ newCtr ဆိုတဲ့ identifier (3) ခု ကို type int လို့ကြေငြာပါတယ်။ နောက်တစ်ကြောင်းမှာ oldCtr ကို 1234 လို့ assign လုပ်ပေး ပါတယ်။

- oldCtr တန်ဖိုးကို increment အရင်လုပ်ပေးပြီး newCtr နဲ့ညီပေးပါတယ်။ ဒီတော့ newCtr = 1235 နဲ့ oldCtr = 1235 ဖြစ်သွားပါပြီ။

- ဒီတစ်ခါ kounter တန်ဖိုးကို newCtr = 1235 နဲ့အရင် assign လုပ်ပြီး newCtr တန်ဖိုးကို decrement လုပ်ပါတယ်။ ဒီတော့ newCtr = 1234 ပြန်ဖြစ်သွားပါပြီ။

- နောက်ဆုံး statement (3) ခုကနေ oldCtr ၊ newCtr နဲ့ kounter တန်ဖိုးတွေကို တစ်ခုချင်း display လုပ်ပြသွားပါလိမ့်မယ်။ အဖြေကို ပုံ (၁. ၂၂) မှာကြည့်ပါ။



ပုံ (၁. ၂၂)

Compound Assignment Operators

C++ မှာ ပါဝင်တဲ့ += ၊ -= ၊ /= ၊ %= ၊ <<= အစရှိတဲ့ Compound assignment operator တွေကို ဇယား (၁. ၇) မှာဖေါ်ပြထားပါတယ်။ ဒီ operatorတွေအသုံးပြုပြီး statement တွေကို short form ဖြစ်အောင်ပြောင်းရေးလို့ရပါတယ်။ ဥပမာ sum= sum + counts; ကို sum += counts; လို့ရေးလို့ရပါတယ်။

ဇယား (၁. ၇) **Compound Assignment Operators**

Symbol	Description
+=	Addition assignment
-=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment
<<=	Shift left assignment
>>=	Shift right assignment
&=	Bitwise AND assignment
|=	Bitwise OR assignment
^=	Bitwise exclusive OR assignment

၁။ ပုံ (၁. ၂၃) မှာဖော်ပြထားတဲ့ program ဟာဆိုရင် multiplication assignment operator ကိုအ
ပြုပြီး ပေးထားတဲ့ကိန်းတစ်ခုရဲ့ factorial ကိုရှာပေးတဲ့ program ပါပဲ ၊ လေ့လာကြည့်ပါ။

၂။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ Enter a number : လို့ prompt တစ်ခုပေါ်
လိမ့်မယ်။ ကျွန်တော်တို့က keyboard ကနေ ဂဏန်း 10 ကိုရိုက်ထည့်ပေးပြီး ENTER key နှိပ်လိုက်တာနဲ့ 1
တန်ဖိုးကို တွက်ပေးလိုက်ပါပြီ။ ပုံ (၁. ၂၄) ကိုကြည့်ပါ။

၃။ Ex1011.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း program မှာအသုံးပြုမယ့် identifiers i နဲ့ num တို့ကို type int လို့ကြေငြာ
fac ကို long int လို့ကြေငြာပါတယ်။ i က counter ပါ။ num က ကျွန်တော်တို့ factor
ရှာပေးချင်တဲ့ကိန်းပါ။ fac က num! ကို store လုပ်မယ့် identifier ဖြစ်ပါတယ်။

Ex1011.cpp

```
// Listing 1.11: Using a compound operator

#include <iostream>
int main()
{
        int     i, num;
        long  int  fac;

        cout << "\nEnter a number : ";
        cin >> num;
        fac = 1;
        for (i=2; i <= num; i++)
                fac *= i;
        cout << "\nFactorial of " << num << " is " << fac << endl;

        return 0;
}
```

ပုံ (၁. ၂၃)

Quincy 99

```
Enter a number : 10

Factorial of 10 is 3628800

Any key to return to Quincy...
```

ပုံ (၁. ၂၄)

■ cout << "\nEnter a number : "; ဆိုတဲ့ statement ကနေ prompt တစ်ခုပေါ်လာပေးရင် keyboard ကနေ 10 ကိုရိုက်ထည့်ပါမယ်။ ဒါဆိုရင် num = 10 လို့ cin >> num; ကနေ assign လုပ်ပေးလိုက်ပါပြီ။

■ for loop ထဲမှာ counter ကို 2 ကနေ 10 အထိပတ်ပေးပြီး num! ကိုအခုလိုတွက်ပေးမှာ

```
fac *= i ▶   fac = fac * i
             2! = 1 * 2 = 2              i = 2
             3! = 2 * 3 = 6              i = 3
             4! = 6 * 4 = 24             i = 4
             5! = 24 * 5 = 120           i = 5
             6! = 120 * 6 = 720          i = 6
             7! = 720 * 7 = 5040         i = 7
             8! = 5040 * 8 = 40320       i = 8
             9! = 40320 * 9 = 362880     i = 9
             10! = 362880 *10 = 3628800  i = 10
```

Conditional Operators

၁။ conditional operator တစ်ခုဟာဆိုရင် program ထဲက ညီမျှခြင်းရဲ့ညာဘက်က expression ကို test လုပ်ပြီး true ဖြစ်ခဲ့ရင် ? ရဲ့နောက်က expression တန်ဖိုးကို ညီမျှခြင်းရဲ့ဘယ်ဘက်က identifier နဲ့ assign လုပ်ပေးပါလိမ့်မယ်။ false ဆိုရင် : နောက်က expression တန်ဖိုးနဲ့ညီပေးမှာပါ။ ဒီဥစ္စာရဲ့ format ကို အခုလို အလွယ်မှတ်ထားလို့ရပါတယ်။

variable = <test> ? expr1 : expr2;

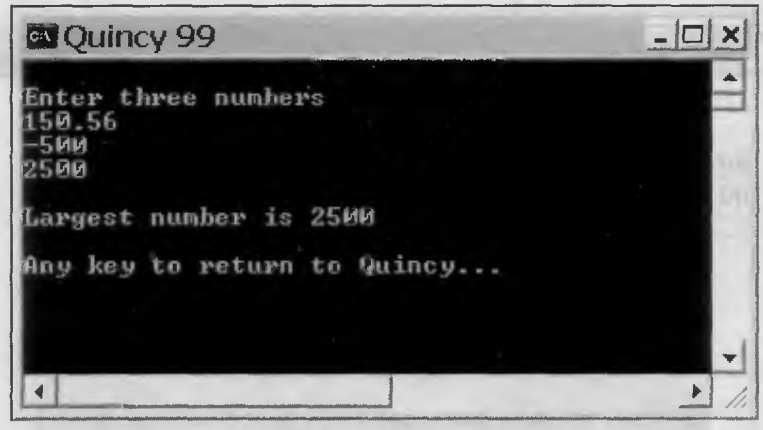
၂။ ပုံ (၁. ၂၅) မှာဖော်ပြထားတဲ့ program ဟာဆိုရင် conditional operator ကိုအသုံးပြုပြီး ပေးထားတဲ့ကိန်း (3) ခုထဲက အကြီးဆုံးကိန်းကိုရှာပေးတဲ့ program ပါပဲ ၊ လေ့လာကြည့်ပါ။

၃။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ Enter three numbers : လို့ prompt တစ်ခု ပေါ်လာမှာပါ။ keyboard ကနေ 150.56 ၊ -500 နဲ့ 2500 တို့ကိုရိုက်ထည့်ပေးပြီး ENTER key နှိပ်လိုက်တာနဲ့ Largest number is 2500 ကို ကွန်ပျူတာမှာ display လုပ်ပြပေးပါလိမ့်မယ်။ ပုံ (၁. ၂၆) ကိုကြည့်ပါ။

Ex1012.cpp

```
// Listing 1.12: Using the conditional operator

#include <iostream>
int main()
{
        float  num1, num2, num3, largst;

        cout << "\nEnter three numbers\n";
        cin >> num1 >> num2 >> num3;

        largst = num1 > num2 ? num1 : num2;
        largst = largst > num3 ? largst : num3;
        cout << "\nLargest number is " << largst << endl;
        return 0;
}
```

ပုံ (၁. ၂၅)



ပုံ (၁. ၂၆)

Ex1012.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း program မှာအသုံးပြုမယ့် identifiers (4) ခုကို type float လို့ကြေငြာပြီး သူတို့အတွက် data တွေကိုထည့်ပေးပါတယ်။

- largst = num1 > num2 ? num1 : num2 ; ဆိုတဲ့ statement ကနေ largst = 150.56 > -500 ? 150.56 : -500 = <true> ? 150.56 : -500 = 150.56 လို့ assign လုပ်ပေးမှာပါ။

- အခုတစ်ခါမှာတော့ largst = largst > num3 ? largst : num3 ; ဆိုတဲ့ statement ကနေ largst = 150.56 > 2500 ? 150.56 : 2500 = <false> ? 150.56 : 2500 = 2500 လို့ assign လုပ်ပေးမှာပါ။ ဒါကြောင့်မို့ ကွန်ပျူတာမှာ Largest number is 2500 ကို display လုပ်ပြတာပါပဲ။

Comma Operators

၁။ C နဲ့ C++ program တွေမှာ expression တွေကို comma (,) တွေနဲ့ခြားပေးလို့ရပါတယ်။ ဥပမာ var1 = (expr1, expr2, expr3); လို့ရေးပေးမယ်ဆိုရင် expression တစ်ခုချင်းရဲ့ တန်ဖိုးတွေကိုတွက်ပေးသွားပြီး ညာအစွန်ဆုံးက expression ရဲ့တန်ဖိုးကို var1 နဲ့ assign လုပ်ပေးသွားမှာဖြစ်ပါတယ်။ ပုံ (၁. ၂၇) မှာဖော်ပြထားတဲ့

Ex1013.cpp

```
// Listing 1.13: Using the comma operator

#include <iostream>
int main()
{
        float  a, b, c,num;

        a = 255.78;
        b = -100;
        c = 18;
        num = (a++, b+3, --c+3);
        cout << "\nValue returned is " << num << endl;

        cout << "a = " << a << endl;
        cout << "b = " << b << endl;
        cout << "c = " << c << endl;
        return 0;
}
```

ပုံ (၁. ၂၇)

program ဟာဆိုရင် comma operator ကိုအသုံးပြုပြီး ပေးထားတဲ့ကိန်း (3) ခုရဲ့တန်ဖိုးတွေကို အသစ်ပြင်ပေးတဲ့ program ပါပဲ ၊ လေ့လာကြည့်ပါ။

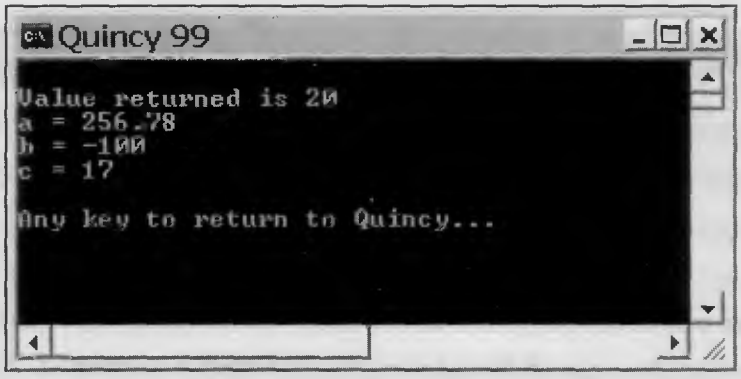
Ex1013.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ ပုံ (၁. ၂၈) မှာပြထားတဲ့အတိုင်းပေါ်လာမှာပါပဲ။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း program မှာအသုံးပြုမယ့် identifiers (4) ခုကို type float လို့ကြေငြာပြီး သူတို့အတွက် data တွေကို a = 255.78 ၊ b = -100 နဲ့ c = 18 လို့ assign လုပ်ပေးပါတယ်။

- num = (a++, b+3, --c+3); ဆိုတဲ့ statement ကနေ a = 255.78+ 1= 256.78၊ b = -100 ၊ c = --c = 18-1 = 17 နဲ့ --c + 3= 17+ 3= 20 တို့ကိုတွက်ပေးမှာပါ။ ဒီတော့ num = 20 ဖြစ်သွားပါပြီ။ b တန်ဖိုးကမပြောင်းပါဘူး။

- ဒါကြောင့်မို့ cout << "\nValue returned is " << num << endl; ကနေ Value returned is 20 လို့ display လုပ်ပြတာပါပဲ။

- ထိုနည်းတူ a = 256.78 ၊ b = 100 ၊ c = 17 တို့ကိုလည်း display လုပ်ပြမှာပါ။ ပုံ (၁. ၂၈) ကိုကြည့်ပါ။



ပုံ (၁. ၂၈)

Initializers

ကျွန်တော်တို့ ရှေ့ပိုင်းမှာတုန်းက program မှာပါတဲ့ variable တွေကို type declare လုပ်ပြီးမှ သူတို့ရဲ့ initial value တွေကို assign လုပ်ပေးခဲ့ပါတယ်။ စင်စစ် type declaration နဲ့အတူ data initialization ကို လည်းလုပ်လို့ရပါတယ်။ ပုံ (၁. ၂၃) မှာဖော်ပြခဲ့တဲ့ Ex1011.cpp program ကို ပုံ (၁. ၂၉) အတိုင်းရအောင်ပြင်ရေးပြီး run လိုက်မယ်ဆိုရင် ပုံ (၁. ၂၄) ကအဖြေမျိုးပဲရမှာပါ။ ဒါဆိုရင် C++ နဲ့ပတ်သက်တဲ့အခြေခံတွေကို စာဖတ်သူအနေနဲ့ စုံစုံလင်လင်လေး သိရှိသွားပြီလို့ထင်ပါတယ်။

Ex1014.cpp

```
// Listing 1.14: Using the initializers

#include <iostream>
int main()
{
        int         num = 10;
        long int    fac = 1;

        for (int i=2; i <= num; i++)
                fac *= i;
        cout << "\nFactorial of " << num << " is " << fac << endl;

        return 0;
}
```

ပုံ (၁. ၂၉)

Chapter 2

C++ function တွေဟာ C++ program တစ်ခုကိုတည်ဆောက်ရာမှာ အရေးပါပြီးအခြေခံတဲ့ subprogram တွေပါပဲ။ function တစ်ခုချင်းဟာ executable program code တွေကိုလက်ခံထားပါတယ်။ C++ program တစ်ခုမှာ အနည်းဆုံး function တစ်ခုတော့ ပါရပါမယ်။ အဲဒါဟာ program အတွက် အဓိကဝင်ပေါက်နဲ့ ထွက်ပေါက်တွေပါဝင်တဲ့ main( ) function ပါပဲ။ main( ) function ကနေတစ်ခြား lower-level function တွေကို cal ခေါ်ပြီး အဲဒီ function တွေထဲက code တွေကို execute လုပ်မှာပါ။ အဲဒီခွိုင်မှာ calling function ဟာအလုပ်ရပ်ထားပြီး call ခေါ်ခံရတဲ့ function က return ပြန်လာမယ့် value တစ်ခုကိုစောင့်နေတယ်လေ။ main( ) function ကနေ call ခေါ်သလိုပဲ lower-level function တစ်ခုကနေ နောက်တစ်ခုကို call ခေါ်တာလည်းရှိပါတယ်။ main( ) function ကိုတော့ call ပြန်ခေါ်လို့မရပါဘူး။ function တစ်ခုဟာ call ခေါ်ခံရတဲ့အခွိုင်မှာ အပြင်က argument တွေ pass ဝင်လာတာကိုလက်ခံနိုင်သလို ပြန်အတွက်မှာ value တစ်ခုကို return လုပ်ပေးနိုင်ပါတယ်။ function တွေကိုအသုံးပြုလို့ အကျိုးရှိရဲ့လားမေးရင် အကျိုးအမြားကြီးရှိတယ်လို့ ဖြေပါမယ်။ ဥပမာ main( ) function လိုမျိုး calling function တွေမှာ ထပ်ခါတစ်လဲ လုပ်ဆောင်ရမယ့်

အလုပ်တွေကို calling function ထဲမှာ ရှုပ်အောင်ရေးစရာမလိုတော့ပါဘူး။ new function တစ်ခု သပ်သပ်ရေးပြီး အဲဒီကိုသွားလိုက်ပြန်လိုက် လုပ်ခိုင်းလို့ရနေပြီမဟုတ်လား။ ဒါဆိုရင် calling function တွေမှာ code တွေရှင်းလင်းသွားသလို C++ programming technique ဟာလည်း အစွမ်းထက်လာမှာပါပဲ။

၂.၁ Create a Simple Function

၁။ function တစ်ခုမှာ အဓိကပါဝင်တဲ့အပိုင်း (2) ခုဟာ function header နဲ့ statement body တို့ဖြစ်ပါတယ်။ function header ကိုထပ်မံစီစစ်ကြည့်ရင် အပိုင်း (3) ခုပါဝင်နေတာကိုတွေ့ရမှာပါ။ (၁) return type (၂) function name နဲ့ (၃) parameter list တို့ဖြစ်ကြပါတယ်။ return value ဟာ C++ data type တွေထဲက တစ်ခုခုဖြစ်နိုင်သလို data type တွေကို point လုပ်နေတဲ့ pointer တစ်ခုလည်းဖြစ်နိုင်ပါတယ်။ ဒါမှမဟုတ် structure ၊ array တွေကို point လုပ်နေတဲ့ pointer မျိုးလည်းဖြစ်လို့ရပါတယ်။ function name ကတော့ identifier တွေကို နာမည်ပေးနည်းနဲ့အတူတူပါပဲ။ parameter list ဟာ blank ဖြစ်နိုင်သလို variable တွေအများကြီးရှိနေလို့လည်းရပါတယ်။ ဒီ variable တွေထဲကို calling function က argument တွေကိုကော်ပီကူးထည့်မှာပါ။ statement body ကို program တစ်ခုရေးတဲ့အတိုင်းရေးရပါမယ်။ local variable declaration တွေ ၊ executable code တွေပါဝင်ပါလိမ့်မယ်။ ကျွန်တော်တို့ function တစ်ခုကို call မခေါ်ခင် function ရဲ့ parameter type တွေနဲ့ return value အမျိုးအစားတွေဟာ ဘာတွေပါလို့ compiler ကို အရင်အသိပေးရပါမယ်။ အဲဒီလို function declaration ကို prototype လို့ခေါ်ပါတယ်။ ကောင်းပြီ ၊ လွယ်ကူတဲ့ function တစ်ခုကို define လုပ်ပြီး main( ) function ကနေ call ခေါ်ကြည့်ရအောင်။ ပုံ (၂.၁) က Ex201.cpp program ကိုလေ့လာကြည့်ပါ။

၂။ Ex201.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ ပုံ (၂.၂) မှာပြထားတဲ့အတိုင်းပေါ်လာမှာပါပဲ။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ main( ) function ရဲ့ထိပ်ပိုင်းမှာ func နာမည်နဲ့ functionprototype ကိုကြေငြာပါတယ်။ return type က void ဖြစ်ပြီး parameter list မှာ variable ဘာမှမရှိပါဘူး။ prototype declaration နောက်ဆုံးမှာ semicolon (;) နဲ့ပိတ်ပေးရပါမယ်။
- main( ) function ထဲမှာ အရင်ဆုံးလုပ်တာက STEP I: Inside the main( ) function ဆိုတဲ့စာကြောင်းကို display လုပ်တာပါပဲ။ ဒါဟာ ပုံ (၂.၂) က ပထမစာကြောင်းပေါ့။

Ex201.cpp

```
// Listing 2.1: Create a simple function

#include <iostream>
void   func( );    // declare the function prototype

int  main( )
{
        cout << "\nSTEP I: Inside the main( ) function\n";

        // call to the function
        func( );
        cout << "\nSTEP III: Back in the main( ) function again\n\n";

        return 0;
}

void  func( )
{
        // function body
        cout << "\nSTEP II: Now inside the func( ) function\n";
}
```

ﺽ (ﺝ٠ ٥)

Quincy 99

```
STEP I: Inside the main( ) function

STEP II: Now inside the func( ) function

STEP III: Back in the main( ) function again

Any key to return to Quincy...
```

ﺽ (ﺝ٠ ﺝ)

- func( ); ကနေ function call လိုက်တဲ့အတွက် void func( ) ထဲကိုဝင်လာပါပြီ။ calling function မှာ argument ထည့်ပေးမထားပါဘူး။ ဒီတော့ func( ) function ထဲကို တိုက်ရိုက်ရောက်သွားပြီး STEP II: Now inside the func( ) function ဆိုတဲ့စာကြောင်းကို display လုပ်ပြမှာပါ။ ဒါဟာ ပုံ (၂. ၂) က ဒုတိယစာကြောင်းပေါ့။ function body ထဲမှာလုပ်စရာကုန်သွားရင် main( ) function ကိုပြန်လာမှာပါ။

- အခုတစ်ခါ main( ) function ထဲက func( ); အောက်မှာရှိတဲ့ code statement ကိုဆက်ပြီးဖြေရှင်းပါလိမ့်မယ်။ ဒီတော့ STEP III: Back in the main( ) function ဆိုတဲ့စာကြောင်းကို display လုပ်ပြဦးမှာပါ။ ဒါဟာ ပုံ (၂. ၂) က တတိယစာကြောင်းပေါ့။ ဒါဆိုရင် Ex201.cpp program ဟာ ပြီးသွားပါပြီ။

- တစ်ကယ်လို့ func( ) function တစ်ခုလုံးကို main( ) function အပေါ်မှာကြိုရေးထားရင် function prototype declaration မပါပဲနဲ့ program run လို့ရပါတယ်။ ပုံ (၂. ၃) ကိုကြည့်ပါ

Ex201A.cpp

```
// Listing 2.1A: Create a simple function

#include <iostream>
void func()
{
        // function body
        cout << "\nSTEP II: Now inside the func() function\n";
}

int main()
{
        cout << "\nSTEP I: Inside the main() function\n";
        // call to the function
        func();
        cout << "\nSTEP III: Back in the main() function again\n\n";

        return 0;
}
```

ပုံ (၂. ၃)

Printing a Line of Asterisks

ဒီတစ်ခါတင်ပြမှာက starline( ) ဆိုတဲ့ function (void) တစ်ခုကို call ခေါ်ပြီး program output ကို asterisk(*) တွေပေါ်လာအောင်လုပ်ခိုင်းမှာပါ။ ပုံ (၂.၄) မှာရေးထားတဲ့ Ex202.cpp program က code တွေကိုလေ့လာကြည့်ပါ။

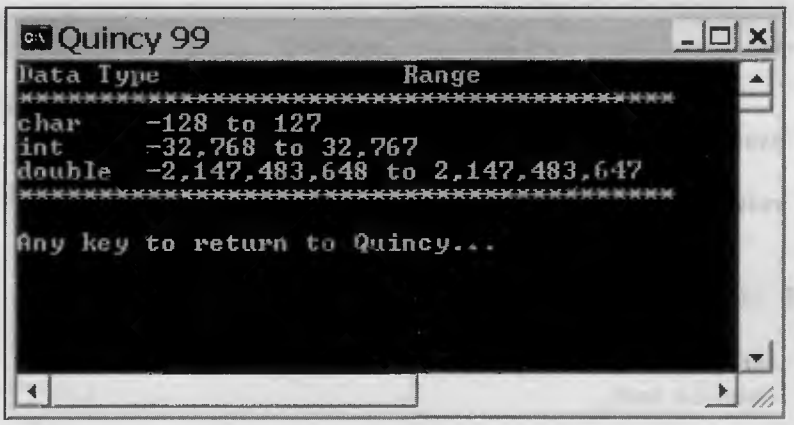
Ex202.cpp

```
// Listing 2.2: This program demonstrates a simple function
// that prints a line of 40 asterisks.

# include <iostream>
void    starline( );

int  main( )
{
     cout << "Data Type              Range" << endl;
     starline( );
     cout << "char      -128 to 127" << endl
          << "int       -32,768 to 32,767" << endl
          << "double  -2,147,483,648 to 2,147,483,647" << endl;
     starline( );

     return 0;
}

void  starline( )
{
     for (int  i=0; i<=40; i++)  cout << "*";
     cout << endl;
}
```

ပုံ (၂.၄)

Ex202.cpp program ကို run လိုက်မယ်ဆိုရင်ကွန်ပျူတာမှာ ပုံ (၂.၅) မှာပြထားတဲ့အတိုင်းပေါ်လာမှာပါပဲ။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း main( ) function ရဲ့အပေါ်မှာ starline နာမည်နဲ့ function prototype တစ်ခုကို ကြေငြာပါတယ်။ return type က void ပါ။ parameter list မှာ ဘာမှမရှိပါဘူး။ နောက်ပြီး Data Type <2 Tabs> Range ဆိုတဲ့ heading ကိုရေးခိုင်းပြီး starline( ) function ကို call ခေါ်ပါတယ်။
- starline( ) function ထဲမှာ for loop ကိုအသုံးပြုပြီး ကြယ်ပွင့်အခု (40) ကို print လုပ်ခိုင်းပါတယ်။ counter i ကို for loop ထဲမှာပဲ data type နဲ့တွဲပြီးကြေငြာထားပါတယ်။ asterisk (40) ခုကို print လုပ်ပြီးရင် main( ) ကိုပြန်လာပါပြီ။ အခုတစ်ခါ char <2 Tabs> -128 to 127 ကိုရိုက်ပြီး နောက်တစ်ကြောင်းကိုကူးပါတယ်။ endl နောက်မှာ semicolon(;) မရေးခဲ့အပေါ်ကအတိုင်းဆက်ရေးရင် cout ကို ဆက်ပြီးအသုံးပြုမယ်ဆိုတဲ့သဘောပါပဲ။ ဒီတော့ int <2 Tabs> -32,768 to 32,767 ကိုဆက်ရိုက်မှာပါ။ ဒီအတိုင်းပဲ တတိယအကြောင်းကိုလည်း print လုပ်သွားမှာပါ။
- ဒုတိယအကြိမ် starline( ) function ကို call ခေါ်ပြီးတော့ print လုပ်ခိုင်းမယ်ဆိုရင် output မှာ asterisk (40) ခုဟာ ထပ်ပေါ်လာမှာပါပဲ။ ပုံ (၂.၅) ကိုကြည့်ပါ။



ပုံ (၂.၅)

၂.၂ Pass the Variables by Value

၁။ calling program တစ်ခုကနေ function တစ်ခုကို call ခေါ်လိုက်တဲ့အခါမှာ argument တစ်ချို့ကို called function ထဲက variable တွေထဲကို pass လုပ်ပေးပါတယ်။ data pass လုပ်နည်းတွေအများကြီးရှိပါ

ဘယ်။ အဓိကနည်း (2) ခုက Passing by value နဲ့ Passing by reference နည်းတွေဖြစ်ပါတယ်။ ပုံ (၂.၆) မှာရေးထားတဲ့ Ex203.cpp program မှာ constant value တွေ pass လုပ်နည်းကို တင်ပြထားပါတယ်။

Ex203.cpp

```
// Listing 2.3: Calculate the area of a triangle

#include <iostream>
float   calArea(float, float);

int main( )
{
        float area = calArea(10,20);
        cout << "\nBase   = " << 10;
        cout << "\nHeight = " << 20;
        cout << "\nArea of the triangle = " << area << endl;
        return 0;
}

float   calArea(float  b, float  h)
{
        float area = 0.5*b*h;
        return  area;
}
```

ပုံ (၂.၆)

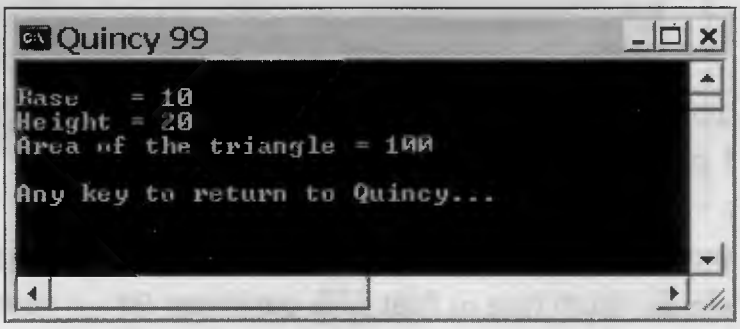
၂။ Ex203.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ ပုံ (၂.၇) မှာပြထားတဲ့အတိုင်းပေါ်လာမှာပါပဲ။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်း main( ) function ရဲ့အပေါ်မှာ calArea နာမည်နဲ့ prototype ကိုကြေငြာပါတယ်။ return type က float ဖြစ်ပြီး parameter list မှာ float data(2) ခုကို pass လုပ်ခိုင်းမှာပါ။
- main( ) function ထဲမှာ အရင်ဆုံးလုပ်တာက identifier area ကို float လို့ declare လုပ်ပြီး calArea( ) function ကို call ခေါ်ပါတယ်။ pass လုပ်ပေးမယ့် argument တွေက constant value တွေပါပဲ။ 10 က တြိဂံတစ်ခုရဲ့ base ဖြစ်ပြီး 20 က height ဆိုပါစို့။

- pass လုပ်ယူသွားတဲ့ constant argument (2) ခုကို function calArea( ) ထဲက b နဲ့ h ဆိုတဲ့ variable တွေထဲမှာ အသီးသီး copy ထည့်ပေးလိုက်ပါတယ်။ ဒီတော့ b = 10 နဲ့ h = 20 ဖြစ်သွားပါပြီ။ ဒီနေရာမှာ သတိထားရမှာက prototype declaration နဲ့ function header တို့ဟာတူညီရပါမယ်။ အခုလို data pass လုပ်တာကို passing by value လို့ခေါ်ပါတယ်။

- အခုဆိုရင် function body ထဲရောက်နေပါပြီ။ local variable ဖြစ်တဲ့ area ရဲ့ data type ကို area လို့အရင်ကြေငြာပြီး တြိဂံဧရိယာကို (base * height)/2 ဆိုတဲ့ပုံသေနည်းနဲ့တွက်ယူပါတယ်။ float area = 0.5 * b * h; ဆိုတဲ့ statement ဟာ အဲဒီအတွက် ရေးထားတာပါ။ area ကိုတွက်ပြီးသွားရင် return area; ကနေ function return value အဖြစ် area ကို return လုပ်ပေးပါလိမ့်မယ်။ ဒီတော့ area ရဲ့ data type နဲ့ return value data type တို့ဟာလည်း တူရမှာပေါ့။

- main( ) function ကိုပြန်ရောက်လာတဲ့အခါမှာ area= calArea(10,20)= return value ကို assign လုပ်ပေးမှာဖြစ်ပါတယ်။ main( ) function ထဲက local variable: area ထဲမှာ တြိဂံဧရိယာတန်ဖိုးက ရောက်နေပါပြီ။ နောက်ဆုံးအနေနဲ့ cout << expression ကိုအသုံးပြုပြီး base ၊ height နဲ့ area တို့ရဲ့အဖြေကိုကွန်ပျူတာမှာ display လုပ်ပေးလိုက်ပြီဖြစ်ပါတယ်။ ပုံ (၂.၇) ကိုကြည့်ပါ။



ပုံ (၂.၇)

၂။ Ex204.cpp ဟာဆိုရင် Ex203.cpp ကို အသစ်ပြင်ရေးထားတာပါ။ variable တွေကို pass လုပ်နည်းနဲ့ program ကိုပြင်ရေးထားပါတယ်။ ပုံ (၂.၈) ကိုကြည့်ပါ။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

အပေါ်ဆုံး statement ဟာ base ၊ height ၊ area တို့ကို global variable တွေလို့ declare လုပ်ပြီး prototype (3) ခုကို define လုပ်ပေးထားပါတယ်။ main( ) ထဲမှာပထမဆုံး getData( ) function ကို call ခေါ်ပါတယ်။ getData( ) function ထဲမှာ base နဲ့ height အတွက် တန်ဖိုးတွေကိုရိုက်ထည့်

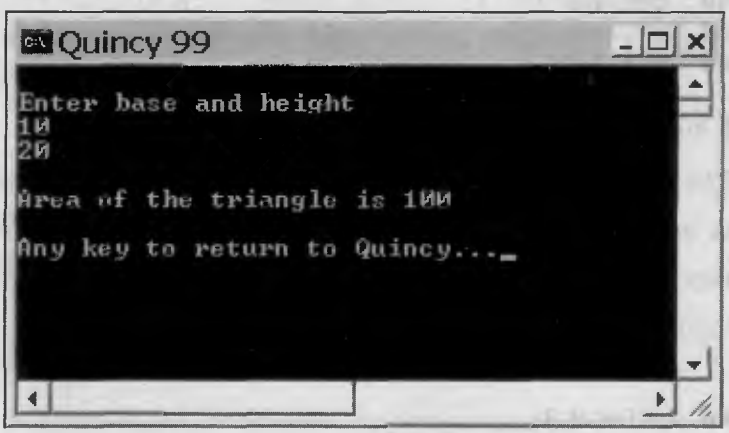
Ex204.cpp

```
// Listing 2.4: Passing variables by value
#include <iostream>

float   base, height, area;
void    getData( );
void    calArea( );
void    printArea( );

int main( )
{
        getData( );
        calArea( );
        printArea( );

        return 0;
}

void   getData( )
{
        cout << "\nEnter base and height\n";
        cin >> base >> height;
}

void   calArea( )
{
        area = 0.5*base*height;
}

void   printArea( )
{
        cout << "\nArea of the triangle is " << area << endl;
}
```

ပုံ (၂.၈)

ပြီးရင် main( ) ကိုပြန်ရောက်လာပြီး calArea( ) function ကို call ခေါ်မှာပါ။ calArea( ) function ထဲရောက်လာတဲ့အခါကျရင် တြိဂံဧရိယာကို ပုံသေနည်းနဲ့တွက်ယူပါမယ်။ base နဲ့ height တန်ဖိုးတွေက သိနေပြီမို့ function ထဲမှာကြေငြာစရာမလိုတော့ပါဘူး။ area တွက်ပြီးရင် main( ) ကိုပြန်ရောက်လာပြီး printArea( ) ကို call ဆက်ခေါ်ပါတယ်။ printArea( ) function က area တန်ဖိုးကို print လုပ်ပြပါလိမ့်မယ်။

၂။ Ex204.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ ပုံ (၂.၉) မှာပြထားတဲ့အတိုင်းပေါ်လာမှာပါ။ အဖြေက Ex203.cpp program နဲ့အတူတူပါပဲ။



ပုံ (၂.၉)

Creating a Self-defined Square Root Function

၁။ အခုတင်ပြမယ့် program ဟာဆိုရင် sqroot( ) ဆိုတဲ့ 10 ကို self-defined function တစ်ခုကို အသုံးပြုပြီး [illegible] နှစ်ထပ်ကိန်းရင်းကိုရှာပေးမှာပါ။ function အလုပ်လုပ်ပုံက calling function ကနေ constant (2) ကို called function ထဲ pass လုပ်ပေးလိုက်ပြီး အဲဒီမှာတွက်တာချက်တာလုပ်ပြီးတာနဲ့ အဖြေကိုလည်း print လုပ်ပေးခဲ့ပါတယ်။ အပြန်မှာ return value မယူပဲ main( ) ကိုပြန်လာတာပါ။ ဒီသဘောတရားအတိုင်း Ex205.cpp program ကိုရေးထားတာဖြစ်ပါတယ်။ ပုံ (၂.၁၀) မှာရေးထားတဲ့ Ex205.cpp program ရဲ့ code အတွဲကို လေ့လာကြည့်ပါ။

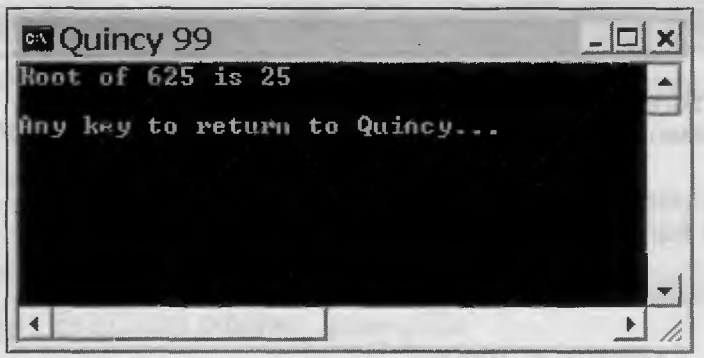
Ex205.cpp

```
// Listing 2.5: This program displays the square root of
// a number passed from a program to the function.

# include <iostream>
void  sqroot (float  , float );

int  main( )
{
        sqroot (625,10000);

        return 0;
}

void  sqroot (float   xn, float   xr)
{
        for (int  i=1; i<=50; ++i)
                xr = (xr+xn/xr)/2;
        cout << "Root of " << xn << " is " << xr << endl;
}
```

ပုံ (၂.၁၀)

၂။ Ex205.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်း main( ) function ရဲ့အပေါ်မှာ sqroot နာမည်နဲ့ prototype ကိုကြေငြာပါတယ်။ return type က void ဖြစ်ပြီး parameter list မှာ float data(2) ခုကို pass လုပ်ခိုင်းမှာပါ။ ကောင်းပြီ ၊ main( ) function မှာ sqroot (625, 10000) ကို call ခေါ်ပြီး 625 နဲ့ 10000 ကို sqroot( ) function ထဲ pass လုပ်လိုက်ပါတယ်။ passing by value နည်းပဲပေါ့။
- sqroot( ) function header မှာ xn = 625 နဲ့ xr = 10000 ဆိုပြီး constant တွေ pass လုပ်ဝင်သွားပါပြီ။ function body ထဲမှာ xn ရဲ့ square root တန်ဖိုးကိုရှာပါတယ်။ xr က ကျွန်တော်တို့ assume လုပ်တဲ့ root တန်ဖိုးပါ။ root ရှာတဲ့နည်းက xr = (xr+xn/xr)/2 ဆိုတဲ့ statement ကိုထပ်ခါတစ်လဲပြန်တွက်ခိုင်းတာပါပဲ။ for loop ကိုအကြိမ် (50) ပတ်ပြီးသွားရင်အဖြေရပါပြီ။ output result ကို ပုံ (၂.၁၁) မှာဖော်ပြထားပါတယ်။



ပုံ (၂.၁၁)

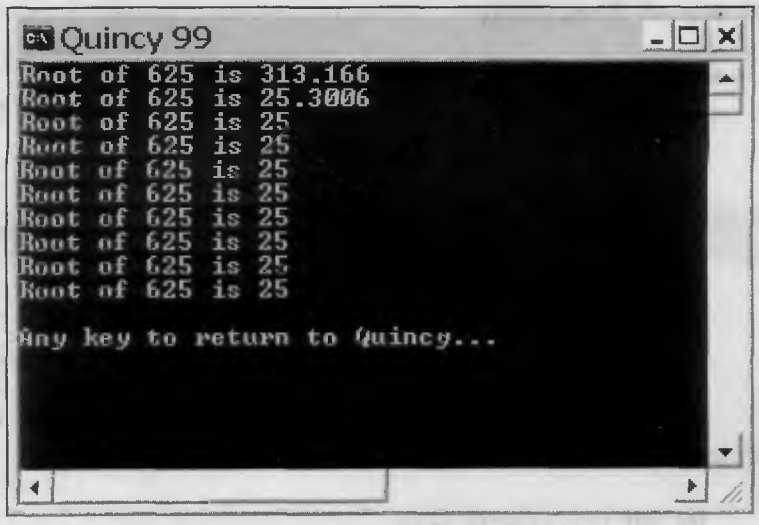
၃။ ဒီ program မှာ ကွန်ပျူတာကအဖြေကို ဖြည်းဖြည်းချင်း မှန်အောင်ခန့်မှန်းတွက်ချက်လာတာကို မြင်ချင်တယ် ဆိုရင် ပုံ (၂.၁၂) မှာပြထားတဲ့အတိုင်းပြင်ရေးပြီး run ကြည့်ပါ။

Ex205A.cpp

```
// Listing 2.5A: This program displays the square root of
// a number passed from a program to the function.

# include <iostream>
void  sqroot (float, float);

int  main( )
{
        sqroot (625,10000);
        return 0;
}

void  sqroot (float  xn, float  xr)
{
        for (int  i=1; i<=50; i++)
        {
                xr = (xr+xn/xr)/2;
                if (i % 5 == 0)
                        cout << "Root of " << xn << " is " << xr << endl;
        }
}
```

ပုံ (၂.၁၂)

၄။ ထူးခြားတာကိုတွေ့တယ်မဟုတ်လား။ for loop ကိုအကြိမ် (50) ပတ်တွက်ခိုင်းပြီး ခန့်မှန်းအဖြေတွေကို (5) ကြိမ်တွက်ပြီးတိုင်း တစ်ကြိမ်နှုန်းနဲ့ print လုပ်ခိုင်းထားတဲ့အတွက် ပုံ (၂.၁၃) ကလိုမျိုး display မှာပေါ်နေတာ ဖြစ်ပါတယ်။ စာဖတ်သူကိုယ်တိုင် program trace လုပ်ကြည့်ပါ။



ပုံ (၂.၁၃)

၅။ ဒီ program မှာ root ရှာချင်တဲ့ဂဏန်းနဲ့ assumed value တို့ကို keyboard ကနေ အရှင်ထည့်ပေးနိုင်မယ်ဆိုရင် ဒီ program ကပိုကောင်းလာမှာပါ။ xn နဲ့ xr တို့ရဲ့တန်ဖိုးတွေကို variable တွေအနေနဲ့ pass လုပ်ပေးကြည့်ရအောင်။ ပုံ (၂.၁၄) မှာရေးထားတဲ့ program code တွေကို ဘယ်လိုပြင်ရေးထားလဲဆိုတာကို သတိပြုကြည့်စေချင်ပါတယ်။

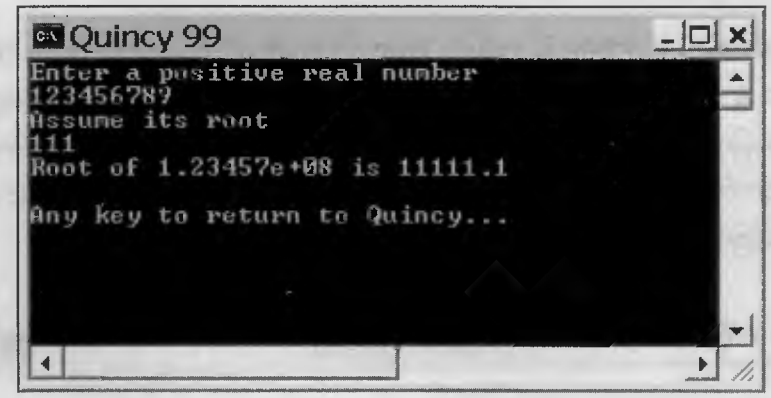
- sqroot( ) function ရဲ့ return type ကို float လို့ပြင်ရေးထားပါတယ်။ main( ) ထဲမှာ xn နဲ့ xr တန်ဖိုးတွေကို keyboard ကနေ ရိုက်ထည့်ပေးနိုင်အောင်ပြောင်းရေးထားပါတယ်။ အဖြေကို cout << နဲ့ sqroot(xn xr) တို့ကိုတွဲရေးပြီး print ထုတ်ခိုင်းထားပါတယ်။
- sqroot( ) function ထဲမှာတော့ root ရှာတဲ့နည်းသက်သက်ကိုပဲရေးထားပါတယ်။ တွက်ပြီးရင် အဖြေကို xr = sqroot(xn, xr) အနေနဲ့ return လုပ်ပေးမှာပါ။

၆။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၂.၁၅) မှာပြထားတဲ့အတိုင်း တွက်သွားလို့ရပါတယ်။

Ex205B.cpp

```
// Listing 2.5B: This program displays the square root of
// a number passed from a program to the function.

# include <iostream>
float  sqroot (float, float);

int  main( )
{
        float   xn, xr;

        cout << "Enter a positive real number\n";
        cin >> xn;
        cout << "Assume its root\n";
        cin >> xr;
        cout << "Root of " << xn << " is " << sqroot(xn,xr) << endl;
        return  0;
}

float  sqroot(float   xn, float   xr)
{
        for (int  i=1; i<=50; i++)
                xr = (xr+xn/xr)/2;
        return xr;
}
```

รูป (ง. ๑๗)



รูป (ง. ๑๗)

၂.၃ Pass Variables by Reference

၁။ C++ program တစ်ခုမှာ calling function ထဲက argument တစ်ချို့ကို called function ဆီကို pass လုပ်တဲ့အခါမှာ argument-to-argument ကော်ပီလုပ်ခိုင်းတာမျိုးမဟုတ်ပဲ calling argument မှာပြောင်းလဲ ဘာတွေကို address မှာ တိုက်ရိုက်ပြင်သွားအောင် argumen-to-address pass လုပ်နည်းကို Passing by Reference လို့ခေါ်ပါတယ်။ Ex203.cpp program ကို ဒီနည်းနဲ့ပြောင်းရေးကြည့်ရအောင်။ ပုံ (၂.၁၆) မှာရေးထားတဲ့ Ex206.cpp program ကိုလေ့လာကြည့်ပါ။

Ex206.cpp

```
// Listing 2.6: Passing variables by reference

#include <iostream>
void    calArea(float, float, float& );
void    printArea(float);

int main( )
{
        float   base, height, area;

        cout << "\nEnter base and height\n";
        cin >> base >> height;
        calArea(base, height, area);
        printArea(area);
        return 0;
}

void  calArea(float b, float h, float &x)
{
        x = 0.5*b*h;
}

void  printArea(float  a)
{
        cout << "\nArea of the triangle is " << a << endl;
}
```

ပုံ (၂.၁၆)

၂။ Ex206.cpp ကိုလေ့လာကြည့်မယ်ဆိုရင်

- အပေါ်ဆုံး statement (2) ခုဟာ function prototype declaration တွေပါ။ Ex203.cpp မှာကြေငြာနည်းနဲ့ မတူတော့ပါဘူး။ called function က return type တွေကို void တွေချည်း လုပ်ထား ပါတယ်။ ဒီတော့ တွက်လို့ချက်လို့ရတဲ့အဖြေတွေကို function return value အနေနဲ့ return မလုပ်တာတော့သေချာပါပြီ။ calArea( ) function ထဲကနောက်ဆုံး argument ကို သတိပြုကြည့်ပါ။ ဒီဥစ္စာရိုးရိုး variable မဟုတ်ပါဘူး။ calling function က argument ကို function မှာ address တစ်ခုနဲ့ pass လုပ်ပေးထားတယ်လေ။ ဒီတော့ address နေရာမှာ တစ်ခုခုပြင်လိုက်မယ်ဆိုရင် အဲဒီ address နဲ့ဆိုင်တဲ့ variable ရဲ့တန်ဖိုးဟာ program ရဲ့ဘယ်နေရာမှာပဲရှိရှိ ပြင်ပြီးသားဖြစ်သွားပါပြီ။
- main function ကိုပြန်ရောက်လာတဲ့အခါမှာ area တန်ဖိုးကို ကွန်ပျူတာကသိနေပြီဖြစ်တာမို့ assign လုပ်ပေးမှာဖြစ်ပါတယ်။ နောက်ဆုံးအနေနဲ့ printArea( ) function ကို call ခေါ်ပြီး function ထဲမှာ area တန်ဖိုးကို print လုပ်ပေးမှာပါပဲ။

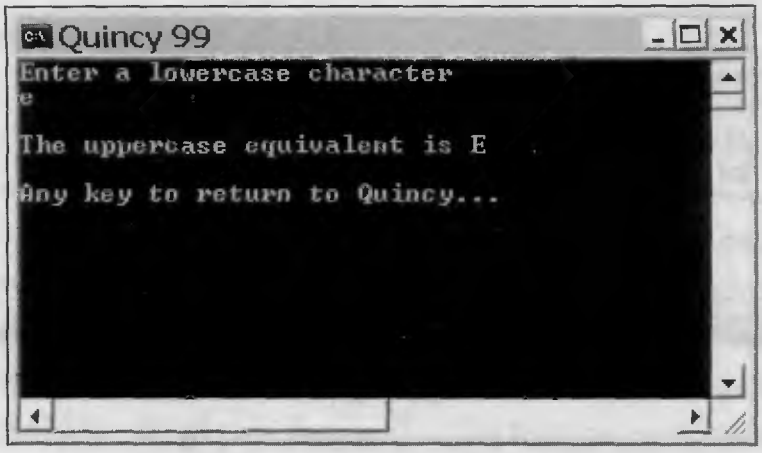
Lowercase to Uppercase Character Conversion Program

ပုံ (၂.၁၇) မှာပြထားတဲ့ Ex207.cpp program ကို လေ့လာကြည့်မယ်ဆိုလို့ရှိရင် function ကို call ခေါ်ကတည်းက data ယူသွားတာကို တွေ့ရပါတယ်။ data က lowercase letter 'e' ပါ။ lower_to_upper ဆိုတဲ့ function ထဲမှာ passing by reference နည်းနဲ့ uppercase ရဲ့ address မှာ letter 'E' ဖြစ်အောင်ပြင်ပေးလိုက်ပါတယ်။ ပြောင်းတဲ့နည်းကတော့ c2 = (c1>='a' && c1<= 'z') ? ('A'+c1-'a') : c1; statement ကို အသုံးပြုတာပါ။ ကျွန်တော်တို့က c1 = 'e' အနေနဲ့ pass လုပ်ပေးလိုက်တဲ့အခါမှာ (c1>='a' && c1<= 'z') expression နဲ့ test လုပ်လိုက်တဲ့အခါကျရင် TRUE ဖြစ်တဲ့အတွက် c2 ကို 'A' + c1 - 'a' နဲ့ assign လုပ်ပေးပါလိမ့်မယ်။ char 'A' ရဲ့ ASCII hex value က 41 ပါ။ c1 = 'e' အတွက်က 65 ပါ။ 'a' အတွက်က 61 ပါ။ ဒီတော့ c2 = 'A' + c1 - 'a' = 41 + 65 - 61 = 45 ကိုရပါတယ်။ ASCII hex value = 45 ဟာ 'E' ဖြစ်တဲ့အတွက် character variable (upper) ဟာ 'E' ဖြစ်သွားပါပြီ။ တစ်ကယ်လို့ c1 = 'E' လို့ထည့်ပေးမယ်ဆိုရင် c2 = (c1>='a' && c1<= 'z') ? ('A'+c1-'a') : c1; statement မှာ c2 = c1 လို့ assign လုပ်ပေးမှာပါပဲ။ ဒီတော့ c2 = c1 = 'E' ဖြစ်သွားပြီလေ။ ပုံ (၂.၁၈) မှာ program ကို run ပြထားပါတယ်။

Ex207.cpp

```
// Listing 2.7: This program reads a lowercase character,
// converts it to uppercase and then writes out
// the uppercase equivalent.

# include <iostream>

void  lower_to_upper (char  c1, char& c2)
{
        c2 = (c1>='a' && c1<= 'z') ? ('A'+c1-'a') : c1;
}

int  main( )
{
        char   lower, upper;

        cout << "Enter a lowercase character\n";
        cin >> lower;
        lower_to_upper(lower, upper);
        cout << "\nThe uppercase equivalent is " << upper << endl;
        return  0;
}
```

ပုံ (၂.၁၇)



ပုံ (၂.၁၈)

Swap Variables by Reference

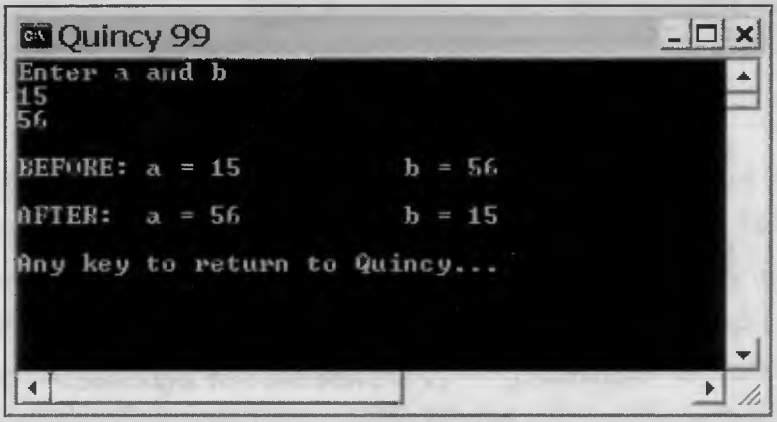
၁။ ပုံ (၂.၁၉) မှာပြထားတဲ့ Ex208.cpp program ဟာဆိုရင် passing by reference နည်းကိုအသုံးပြုပြီး variable a နဲ့ b တို့ရဲ့ value တွေကိုနေရာချင်းလဲပေးပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex208.cpp

```
// Listing 2.8: This program swap
// two arguments passed by reference.

# include <iostream>
void    swap(int&, int&);

int main()
{
        int    a, b;

        cout << "Enter a and b\n";
        cin >> a >> b;
        cout << "\nBEFORE: a = " << a << "\t\tb = " << b;

        swap(a,b);

        cout << "\n\nAFTER:  a = " << a << "\t\tb = " << b << endl;
        return 0;
}

void   swap(int & x, int &y)
{
        // swap x and y
        int  temp = x;
                x = y;
                y = temp;
}
```

ပုံ (၂.၁၉)

Ex208.cpp program ကို ပုံ (၂.၂၀) မှာ run ပြထားပါတယ်။



ပုံ (၂. ၂၀)

Default Function Arguments

function prototype တစ်ခုကို declare လုပ်တဲ့အခါမှာ parameter list မှာ ပါဝင်တဲ့ argument ထဲက ကြိုက်တဲ့အရေအတွက်ကို default value တွေအဖြစ်ထားပေးလို့ရပါတယ်။ function call ခေါ်လိုက်လို့ ထားတဲ့ argument တွေကိုတွေ့ရင် compiler ကနေ default value တွေကိုအစားထည့်ပေးမှာပါ။ ပုံ (၂. ၂၁) မှာဖော်ပြထားတဲ့ program မှာဆိုရင် function prototype ထဲက argument တွေအတွက် default value တွေကို declare လုပ်ပြီး အသုံးပြုသွားပုံကို တင်ပြထားပါတယ်။

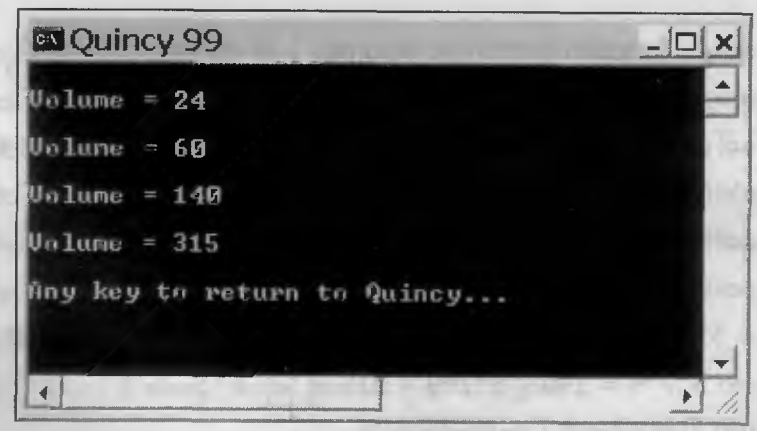
ဒီ program ကို လေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း function prototype volume( ) ကို declare လုပ်တဲ့အခါမှာ argument တွေအတွက် default value တွေကို 2၊ 3၊ 4 လို့ အသီးသီး သတ်မှတ်ပေးပါတယ်။ ပထမဆုံး call ခေါ်တဲ့ volume( ) function မှာ argument လုံးဝမပါတဲ့အတွက် default value (3) ခုလုံးကို အသုံးပြုပြီး volume ကို l*w*h=2*3*4=24 လို့ တွက်ပေးပါလိမ့်မယ်။
- ဒုတိယအကြိမ် call ခေါ်တဲ့အခါမှာ parameter list မှာ argument တစ်လုံးပဲ ပါတဲ့အတွက် ပထမဆုံး default value ကို new argument နဲ့ အစားပြောင်းပေးမှာပါ။ ဒီတော့ volume = 5*3*4= 60 ဖြစ်မှာပါ။ ဒီနည်းအတိုင်းကျန်တဲ့ function တွေကို call ခေါ်ပြီးရင် volume ဟာ 5*7*4 = 140 နဲ့ 5*7*9 = 315 တို့ ဖြစ်ပါလိမ့်မယ်။ ပုံ (၂. ၂၂) မှာ EX209.cpp ကို run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex209.cpp

```
// Listing 2.9: Using functions with default arguments

#include <iostream>
void  volume(float=2, float=3, float=4);

int main( )
{
        volume( );      // all three arguments default

        volume(5);      // provide the first argument

        // provide the first and second arguments
        volume(5,7);

        volume(5,7,9);  // provide all three arguments
        return 0;
}

void  volume(float  l, float  w, float  h)
{
        float vol = l*w*h;
        cout << "\nVolume = " << vol << endl;
}
```

ပုံ (၂.၂၀)



ပုံ (၂.၂၁)

၂.၅ Inline Functions

ကျွန်တော်တို့ function တစ်ခုကို define လုပ်လိုက်တာနဲ့ ပုံမှန်ဆိုရင် compiler ကနေ အဲဒီ function ကို execute လုပ်ဖို့အတွက် instruction အစုံတစ်ခုကို create လုပ်ပေးပြီး memory မှာ သိမ်းထားလိုက်ပါပြီ။ ဒီ function ကို call ခေါ်တိုင်း calling function ကနေ called function ရှိရာကို ခုန်ကျော်သွားပြီး တွက်တာချက်တာ လုပ်မှာပါ။ ပြီးရင် calling function ကိုပြန်လာမယ်ပေါ့။ function ကို (10) ကြိမ် call ခေါ်ရင် (10) ကြိမ်တိတိ ခေါက်တုန့်ခေါက်ပြန် သွားနေမှာပါပဲ။ call ခေါ်တဲ့ function က သိပ်ငယ်နေတယ်ဆိုရင် execution time ထက် traverse time ကပိုနေပါလိမ့်မယ်။ function ကလည်းသိပ်ငယ်ပြီး call ခေါ်တဲ့အကြိမ်ကနည်းမယ်ဆိုရင် inline function ကိုကြေငြာပြီး call ခေါ်တာဟာပိုမြန်မှာပါ။ inline function တစ်ခုကို create လုပ်ချင်ရင် keyword inline ကို function declaration ရှေ့ဆုံးမှာထည့်ပေးရပါမယ်။ inline function လို့ကြေငြာထားရင် compiler က တကယ့် function လို create လုပ်မပေးတော့ပဲ function call ခေါ်တဲ့အခါကျမှ inline function မှာရေး ထားတဲ့ code statement တွေကို calling function မှာ တိုက်ရိုက်ကူးထည့်ပေးပြီး အလုပ်ဖြစ်အောင်လုပ်ခိုင်း ပါတယ်။ ဒီတော့ inline function မှာ code statement တစ်ခုနှစ်ခုလောက်ပဲ ပါတာမျိုးမှ အဆင်ပြေမှာပေါ့။ program ကိုဒီတိုင်းကြည့်ရင် inline function တွေဟာ ပုံမှန် function တွေကို call ခေါ်သလိုပါပဲလေ။

၂.၆ Local and Global Variables

local variables ဆိုတာ main( ) function နဲ့ တစ်ခြား function တွေရဲ့ body ထဲမှာ declare လုပ် ထားတဲ့ variable name တွေပါပဲ။ သူတို့ဟာ မိမိရောက်ရှိနေတဲ့ function တစ်ခုချင်းနဲ့ပဲဆိုင်ပါတယ်။ function တစ်ခုထဲက variable တစ်ခုဟာ နောက် function ထဲက variable နဲ့နာမည်ချင်းတူနေတောင် program မှာ အနှောင့်အယှက်မရှိပါဘူး။ သူတို့တွေရဲ့ scope ဟာ မိမိရောက်ရှိနေတဲ့ function နဲ့ပဲဆိုင်တာကိုး။ နောက်တစ်ခုက function header တွေထဲက parameter list တွေကို အဲဒီ function အတွက် local variable တွေလို့သတ်မှတ် လို့ရပါတယ်။ ဒီ variable တွေကို function body ထဲမှာထပ်ပြီး declare လုပ်စရာမလိုပါဘူး။ global variable တွေကို main( ) function အပြင်ဘက် program ရဲ့ထိပ်ဆုံးနားမှာ declare လုပ်ရပါမယ်။ variable scope ကတော့ program တစ်ခုလုံးနဲ့ဆိုင်ပါတယ်။ ကောင်းပြီ ၊ ပုံ (၂. ၂၃) မှာဖော်ပြထားတဲ့ EX2010.cpp program ကိုလေ့လာကြည့်ရအောင်။

Ex2010.cpp

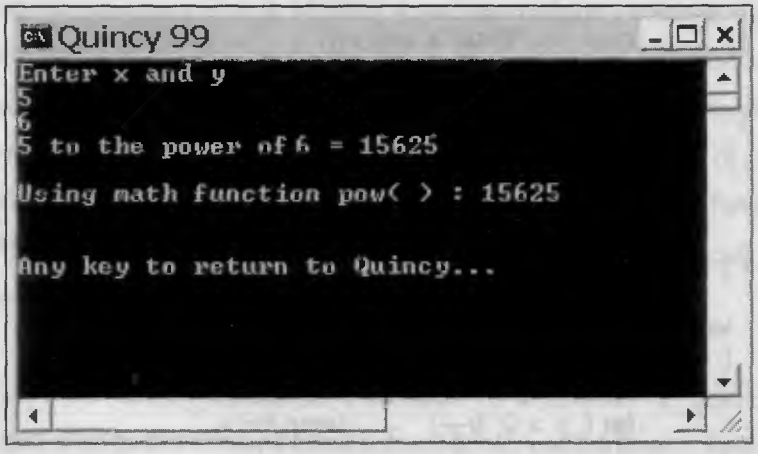
```
// Listing 2.10: This program creates a function power( )
// to compute the value of integer raised to the yth power.

#include <iostream>
#include <cmath>
long  int  power(int, int);

int main( )
{
        long int   x,y,temp;

        cout << "Enter x and y\n";
        cin >> x >> y;
        cout.precision(10);
        temp = power(x,y);   // function call
        cout << x << " to the power of "
             << y << " = " << temp;
        cout << endl << "\nUsing math function pow( ) : "
             << pow(x,y) << endl;
        cout << endl;
        return 0;
}

long  int  power(int  a, int  b)  // function header
{
        long int   temp = 1;
        for (; b > 0; b--)
                temp *= a;

        return  temp;
}
```

ပုံ (၂.၂၃)

၂။ Ex2010.cpp program ကို လေ့လာကြည့်မယ်ဆိုရင်

■ temp ဆိုတဲ့ variable name တစ်ခုဟာ main function မှာရော ၊ power( ) funct
မှာရော (2) မျိုးလုံးမှာ ပါနေပါတယ်။ ဒါပေမယ့်သူတို့ချင်းဟာ ဘာမှမဆိုင်ပါဘူး။ သူတို့ရောက်
function တစ်ခုချင်းစီနဲ့ပဲဆိုင်ပါတယ်။ main( ) function ထဲမှာ local variable (3)
long int လို့ကြေငြာပါတယ်။ ဒီထဲမှာ temp ဆိုတဲ့ variable လည်းပါတာကိုသတိပြုကြ

- x နဲ့ y အတွက် data တွေကိုထည့်ပေးပြီးရင် power( ) function ကို call ခေါ်ပါတယ်။ calling argument တွေက x နဲ့ y ဖြစ်ပါတယ်။ power( ) function ထဲ ရောက်လာတဲ့အခါ x နဲ့ y တို့ရဲ့တန်ဖိုးတွေကို function header က local variable တွေဖြစ်တဲ့ a နဲ့ b မှာအစားပြောင်းထည့်ပေးပါတယ်။
- power( ) function body ထဲက temp ဟာ main( ) function ထဲက temp နဲ့နာမည်ချင်းတူပေမယ့် ဘာမှမသက်ဆိုင်ပါဘူး။ function ထဲမှာတွက်ချက်ပြီးလို့ temp ကို return လုပ်ပေးတာဟာ power( ) function ရဲ့တန်ဖိုးကို return လုပ်ပေးတာနဲ့အတူတူပါပဲ။ main( ) ကိုပြန်ရောက်တဲ့အခါ local variable ဖြစ်တဲ့ temp ထဲကို power( ) function ရဲ့ value ကိုအခုလို temp ▶ power(x,y) ▶ tempပြောင်းထည့်ပေးပါတယ်။
- နောက်ဆုံးမှာ အဖြေ (2) ခုကိုနှိုင်းယှဉ်ပြထားပါတယ်။ ပထမတစ်ခုက ကျွန်တော်တို့ program ရေးပြီး တွက်လို့ရတဲ့အဖြေပါ။ နောက်အဖြေကတော့ math library function ဖြစ်တဲ့ pow( ) function ကိုအသုံးပြုပြီး တွက်လို့ရတာပါ။ ဒါလောက်ဆိုရင် local variable တွေရဲ့အဓိပ္ပါယ်ကို နားလည်သွားပြီလို့ထင်ပါတယ်။ ပုံ (၂. ၂၄) မှာ Ex2010.cpp ကို run ပြထားပါတယ်။



ပုံ (၂. ၂၄)

- Ex2010.cpp ကိုနောက်တစ်မျိုးရေးကြည့်ပါမယ်။ ပုံ (၂. ၂၅) မှာပြထားတဲ့ Ex2010A.cpp program မှာ global variable ကို အသုံးပြုပြီးပြင်ရေးထားပါတယ်။ Ex2010A.cpp ဟာပထမ program နဲ့တူသလိုထင်ရပေမယ့် ရေးပုံရေးနည်း အနည်းငယ်ကွာခြားသွားပါတယ်။ temp ကို global variable အနေနဲ့ကြေငြာထားတဲ့အတွက် main( ) ထဲက temp နဲ့ power( ) ထဲက temp တို့ဟာ တစ်ခုတည်းသော variable ဖြစ်နေပါပြီ။ function body ထဲမှာတွက်ချက်လို့ရတဲ့

temp value ကို return လုပ်စရာမလိုပဲနဲ့ main( ) ကိုပြန်ရောက်သွားအောင်လုပ်ပေးပါလိမ့်
မယ်။ ဒါကိုကြည့်ခြင်းအားဖြင့် global variable တွေဟာ function တွေကြား ကူးကလာပြန်ပြီး
data transfer လုပ်ပေးနိုင်တဲ့ အစွမ်းရှိတဲ့သဘောပေါ့။ စာဖတ်သူကိုယ်တိုင် ဒီ program (2
ခုကို နှိုင်းယှဉ်လေ့လာကြည့်စေချင်ပါတယ်။ program ကို run မယ်ဆိုရင် ပထမအဖြေမျိုးပဲရမှာပါ

Ex2010A.cpp

```
// Listing 2.10A: This program demonstrates
// the use of global variables.

# include <iostream>
void    power (int, int);
long int      temp;                  // define global variable

int  main( )
{
        long int   x, y;

        cout << "Enter x and y\n";
        cin >> x >> y;
        cout.precision(10);
        power (x,y);
        cout << x << " to the power of " << y << " = "
             << temp << endl;
        return  0;
}

void    power (int  a, int   b)
{
        temp =1;
        for (; b > 0; b--)        temp *= a;
}
```

ပုံ (၂.၂၅)

Overloaded Functions

C++ program တစ်ခုမှာ function အများကြီးကို function name တစ်ခုတည်းပေးပြီး create လုပ်နိုင်ရုပါတယ်။ အဲဒါကို function overloading လို့ခေါ်ပါတယ်။ function name တူပေမယ့် parameter list အမျိုးမတူတော့ဘူးပေါ့။ argument type ၊ argument number တွေလည်းကွဲပြားမှာပါ။ ဥပမာ C library မှာရှိတဲ့ absolute value ရှာတဲ့ function သုံးမျိုးလောက်ရှိပါတယ်။ integer အတွက်ဆိုရင် abs( ) ၊ long integer အတွက်ဆိုရင် labs( ) ၊ floating-point value အတွက်ဆိုရင် fabs( ) ဆိုပြီးခွဲထားပါတယ်။ ဒီဥစ္စာကြောင့်ရတာမခက်ဘူးလား ၊ အများမျှားနိုင်ပါတယ်။ ဒီအခက်အခဲကို C++ မှာ function overloading နည်း အသုံးပြုပြီးဖြေရှင်းနိုင်ပါတယ်။ function overloading ကို function polymorphism လို့လည်းခေါ်ပါတယ်။ poly ဆိုတာက many ၊ morph က form ပါ။ ဒါကြောင့်မို့ ပုံစံအမျိုးမျိုးရှိတဲ့ function တစ်ခုကို polymorphic function လို့ခေါ်တာပဲဖြစ်ပါတယ်။

ပုံ (၂.၆) မှာပြထားတဲ့ Ex2011.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်းမှာ PI = 3.141593 နဲ့ radius ၊ height ၊ volume တို့ကို global float variable တွေလို့ကြေငြာပါတယ်။ calVol( ) ကို overloaded function တွေလုပ်ပေးထားပါတယ်။ main( ) function ထဲဝင်လာလာချင်း radius နဲ့ height တို့အတွက် data တွေကို keyboard ကနေရိုက်ထည့်ရမှာပါ။ ပြီးတော့ရင် ပထမ calVol(radius) ကို call ခေါ်ပြီး void calVol(float r) function ထဲမှာ volume = 4*PI*r*r*r/3 ဆိုတဲ့ formula အသုံးပြုပြီး sphere volume ကိုတွက်ယူပါတယ်။ volume ကို global လုပ်ထားတာမို့ return လုပ်ပေးစရာမလိုပါဘူး။ main( ) မှာ volume တန်ဖိုးကို အလိုလိုသိနေပါပြီ။

- နောက်တစ်ခါ call ခေါ်တဲ့ calVol( ) function မှာ argument (2) ခုပါပါတယ်။ radius နဲ့ height ကို argument အနေနဲ့ pass လုပ်ပေးလိုက်ရင် calVol(float r, float h) ထဲဝင်သွားပြီး volume= PI*r*r*h ဆိုတဲ့ formula အသုံးပြုပြီး cylinder volume ကိုတွက်ယူပါတယ်။

- ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၂.၂၇) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ radius = 5.5 နဲ့ height = 7.5 ကိုထည့်ပေးကြည့်ပါ။ sphere နဲ့ cylinder volume တွေကိုတွက်ပေးလိုက်ပါပြီ။

Ex2011.cpp

```
// Listing 2.11: Using overloaded functions
#include <iostream>
const float   PI = 3.141593;
float radius, height, volume;
void   calVol(float);
void   calVol(float, float);
int main( )
{
        cout << "\nEnter radius : ";
        cin >> radius;
        cout << "Enter height : ";
        cin >> height;

        calVol(radius);
        cout << "\nVolume of sphere is " << volume << endl;

        calVol(radius, height);
        cout << "\nVolume of cylinder is " << volume << endl;
        return 0;
}
void   calVol(float r)
{
        volume = 4*PI*r*r*r/3;
}
void   calVol(float r, float  h)
{
        volume = PI*r*r*h;
}
```

ပုံ (၂.၂၆)

Quincy 99

```
Enter radius : 5.5
Enter height : 7.5

Volume of sphere is 696.91

Volume of cylinder is 712.749

Any key to return to Quincy...
```

ပုံ (၂.၂၇)

Overloading Function myDate( )

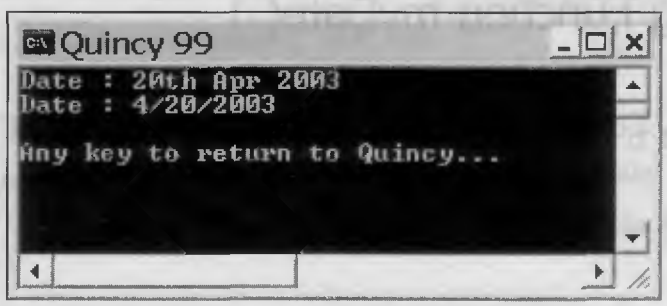
၁။ ပုံ (၂.၂၈) မှာပြထားတဲ့ Ex2012.cpp program ဟာဆိုရင် string type date နဲ့ integer type date ရက်စွဲ (2) မျိုးကို overloaded myDate( ) function တွေကိုအသုံးပြုပြီး display လုပ်ခိုင်းတာဖြစ်ပါတယ်။ call ခေါ်ပုံခေါ်နည်းကို လေ့လာကြည့်ပါ။

Ex2012.cpp

```
// Listing 2.12: This program shows function
// myDate( ) overloading

# include <iostream>
void    myDate (char *str);      // date as string
void    myDate(int, int, int);   // date as numbers

int main( )
{
        myDate("20th Apr 2003");
        myDate(4,20,2003);
        return 0;
}

void    myDate (char *str)
{
        cout << "Date : " << str << endl;
}

void  myDate (int  month, int  day, int  year)
{
        cout << "Date : " << month << "/" << day << "/"
             << year << endl;
}
```

ပုံ (၂.၂၈)

၂။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၂.၂၉) မှာပြထားတဲ့အတိုင်း Date: 20th Apr 2003 နဲ့ Date: 4/20/2003) ဆိုတဲ့ရက်စွဲ (2) မျိုးကို display လုပ်ပြနေပါပြီ။



ပုံ (၂. ၂၉)

Overloading with Default Arguments

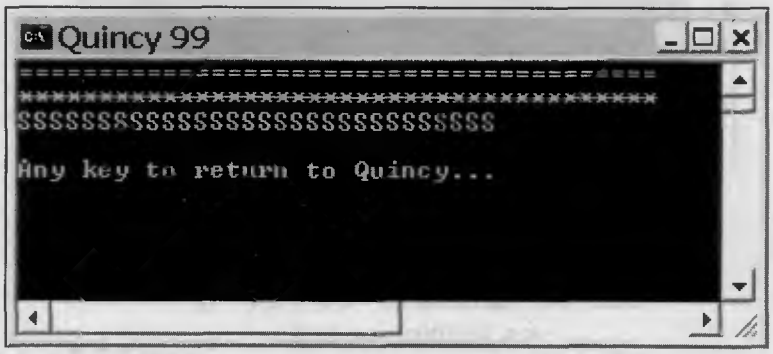
၁။ Ex2013.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် repeat_ch( ) function prototype ထဲမှာ default argument တွေရေးပြီး overloading လုပ်တဲ့အခါ function name တစ်ခု ၊ function definition တစ်ခုကိုပဲအသုံးပြုတာနဲ့ လုံလောက်တာကိုတွေ့ပါလိမ့်မယ်။ ပုံ (၂. ၃၀) ကိုကြည့်ပါ။

Ex2013.cpp

```
// Listing 2.13: This program demonstrates function
// overloading with default arguments.

# include <iostream>
void    repeat_ch(char = '=', int = 40);

int main( )
{
        repeat_ch( );
        repeat_ch('*');
        repeat_ch('S', 30);
        return 0;
}

void    repeat_ch(char  ch, int   n)
{
        for (int  i=0; i<n; i++)   cout << ch;
        cout << endl;
}
```

ပုံ (၂. ၃၀)

၅။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၂. ၃၁) မှာပြထားတဲ့အတိုင်း character line (3) မျိုးကို display လုပ်ပြတာ တွေ့ရမှာပါ။



ပုံ (၂. ၃၁)

၂.၈ Recursion

၁။ recursion ဆိုတာ function တစ်ခုကနေ same funtion ထဲကို ထပ်ခါတစ်လဲ call ပြန်ခေါ်ပြီးတော့ သတ်မှတ်ထားတဲ့ condition တစ်ခု မရောက်သေးမချင်း looping ပတ်နေတာပါပဲ။ recursive program တစ်ခုကို run တဲ့အခါမှာ function call တွေဟာချက်ချင်း execute လုပ်ပြီးအဖြေထုတ်မပေးပါဘူး။ အဖြေတွေကို (တိုင်တစ်တိုင်မှာ အပေါက်ပါတဲ့သံပြားအဝိုင်းတွေကို ထည့်စွပ်ထားပေးသလိုမျိုး) ပြောင်းပြန်ထပ်စီသွားပါတယ်။ သတ်မှတ်ထားတဲ့ condition တစ်ခုကိုရောက်လာလို့ recursion ရပ်သွားတော့မှ အဖြေတွေကို (တိုင်ကနေ သံပြားဝိုင်းတွေ တစ်ခုချင်းထုတ်ပေးသလို) ပြောင်းပြန်အစီအစဉ်နဲ့ ကွန်ပျူတာကထုတ်ပေးသွားမှာပါပဲ။ အဲဒါဟာ recursion နည်းပါ။ ကောင်းပြီ ၊ ဂဏန်းတစ်ခုရဲ့ factorial ကို recursive နည်းနဲ့ရှာကြည့်ရအောင်။ ပုံ (၂. ၃၂) မှာရေးထားတဲ့ code statement တွေကိုကြည့်ပါ။

၂။ ဒီပုစ္ဆာကို ဘယ်လိုရှင်းသွားတယ်ဆိုတာ ကျွန်တော် အစီအစဉ်ချပြီးတော့ရေးပြပါမယ်။

$$n ! = n \times (n - 1) !$$
$$(n - 1) ! = (n - 1) \times (n - 2) !$$
$$(n - 2) ! = (n - 2) \times (n - 3) !$$

Ex2014.cpp

```
// Listing 2.14: This program calculates the factorial of
// a positive integer using recursive method.

# include <iostream>
long double   facto(int);

int main()
{
        int  n;

        cout << "Enter a number : ";
        cin >> n;
        cout << "Factorial of " << n << " is "
                << facto(n) << endl;
        return 0;
}

long double   facto(int i)
{
        if (i <= 1)
                return 1;
        else
                return (i*facto(i-1));
}
```

ပုံ (၂.

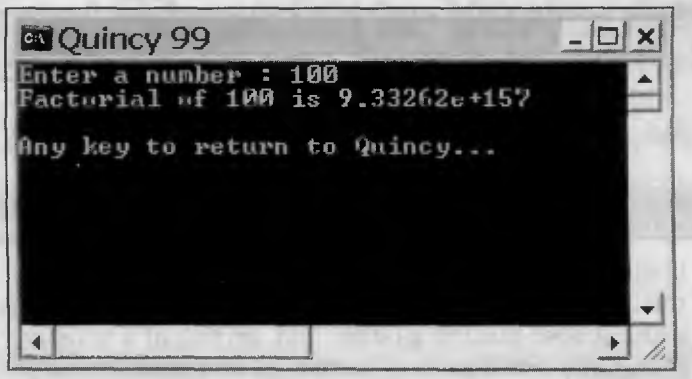
. . . . . .

4 ! = 4 x 3 !

3 ! = 3 x 2 !

2 ! = 2 x 1!

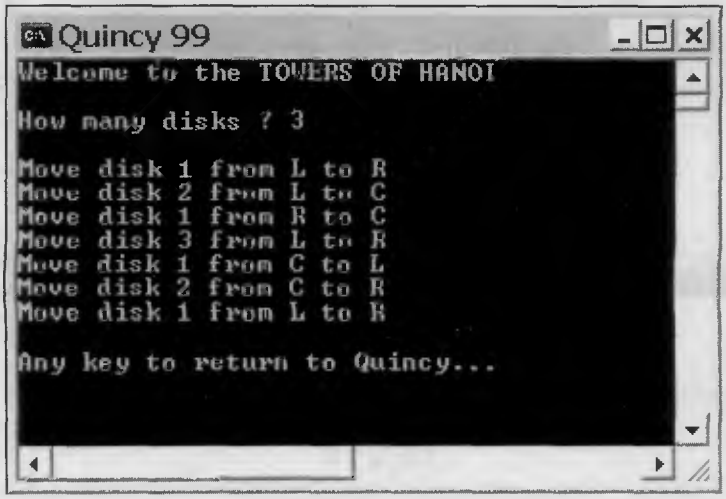
ပထမ function call မှာ n! ကို n x (n-1)! လို့သတ်မှတ်ပြီး execute မလုပ်သေးဘဲ တိုင်မှာစွပ်ထ
ပါတယ်။ ဒီဟာတိုင်ရဲ့အောက်ဆုံးမှာ ရှိနေမှာပါပဲ။ ပြီးတော့ ဒုတိယအကြိမ် function call ကို ခေါ်တဲ့အခါက
argument က (n-1) ဖြစ်သွားပါပြီ။ ဒီတော့ (n-1)! ကိုလည်း (n-1) x (n-2)! လို့သတ်မှတ်ပြီးတော့ exec
မလုပ်သေးပဲ တိုင်မှာပဲထပ်စွပ်ပြန်ပါတယ်။ ဒီလိုနည်းနဲ့ recursion ကိုဆက်လုပ်သွားတာဟာ argumen
အနုတ်တန်ဖိုးဖြစ်သွားတာနဲ့ process ဟာရပ်သွားပြီးတော့ actual value တွေကို reverse order နဲ့ကွန်ပျူတာ
တွက်ထုတ်ပေးပြီလေ။ ပုံ (၂.၃၃) မှာ program run ပြထားတာကိုကြည့်ပါ။



ပုံ (၂.၃၃)

The Tower of Hanoi

၁။ Ex2015.cpp program ကလည်း recursive program တစ်ခုဖြစ်ပါတယ်။ အလယ်မှာအပေါက်ပါပြီး အကြီးအသေးမတူတဲ့ အပိုင်းပြားတွေကို တိုင် (3) တိုင်မှာစွတ်ပေးတဲ့အခါ ဘယ်တိုင်မှာပဲစွတ်စွတ် အသေးအပြားက အကြီးအပြားပေါ်မှာ ရောက်နေအောင်စွတ်ပေးချင်ရင် တစ်ဆင့်ပြီးတစ်ဆင့် မမှားအောင်လုပ်သွားရမယ့်လမ်းကြောင်းကို ဒီ program ကတွက်ထုတ်ပေးမှာပါ။ ပထမတိုင်ကို LEFT၊ ဒုတိယတိုင်ကို CENTER နဲ့ တတိယတိုင်ကို RIGHT လို့ခေါ်ပါမယ်။ အခုနေ disk (3) ခုကိုတိုင်မှာစွတ်ပေးချင်ရင် ပုံ (၂.၃၄) မှာပြထားတဲ့အစီအစဉ်အတိုင်းလုပ်ရပါမယ်။



ပုံ (၂.၃၄)

၂။ ဒီ program ရဲ့ source code တွေကို ပုံ (၂.၃၅) မှာရေးပြထားပါတယ်။ စာဖတ်သူကိုယ်တိုင် trace လုပ်ကြည့်ဖို့ အလုပ်ချန်ခဲ့ပါမယ်။ စိတ်ရှည်ရှည်ထားပြီး program လမ်းကြောင်းကိုရှာကြည့်ပါ ၊ ရပါတယ်။

Ex2015.cpp

```
// Listing 2.15: This program is a game used  to transfer
// the disks from the leftmost pole to the rightmost pole,
// without ever placing a larger disk on top of a smaller
// disk. Only one disk may be moved at a time.

# include <iostream>
void    transfer (int, char, char, char);

int  main( )
{
        int    nDisk;

        cout << "Welcome to the TOWERS OF HANOI\n\n";
        cout << "How many disks ? ";
        cin >> nDisk;
        cout << endl;
        transfer (nDisk, 'L',' R', 'C');
        return  0;
}

void   transfer(int  n, char  from, char  to, char  temp)
{
        if (n > 0)
        {
                transfer (n-1 , from, temp, to);
                cout << "Move disk " << n << " from " << from;
                cout << " to " << to << endl;
                transfer (n-1 , temp,to, from);
        }

}
```

ပုံ (၂.၃၅)

၂.၉ Passing Structure Variables

function ထဲမှာ constant တွေ ၊ variable တွေကိုတစ်ခုချင်း pass လုပ်လို့ရသလို data type အုပ်စု တစ်ခုပါဝင်တဲ့ structure လိုမျိုးကိုလည်း pass လုပ်ပေးလို့ရပါတယ်။ ပုံ (၂.၁၆) မှာပြထားတဲ့ Ex2016.cpp program ဟာဆိုရင် structure တစ်ခုကို function ထဲရောက်အောင် pass လုပ်ပေးနိုင်ပါတယ်၊ လေ့လာကြည့်ပါ။

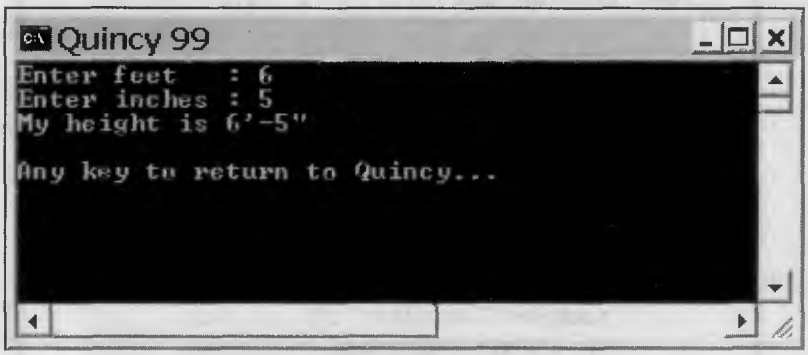
Ex2016.cpp

```
// Listing 2.16: This program demonstrates
// passing structure as argument.

# include <iostream>

struct  height
{
        int     feet;
        float   inches;
};

 void  convr(height  h)
{
        cout << h.feet << "\'-"
             << h.inches << "\"" << endl;
}

int main()
{
        height  myHeight;

        cout << "Enter feet   : ";
        cin >> myHeight.feet;
        cout << "Enter inches : ";
        cin >> myHeight.inches;
        cout << "My height is ";
        convr(myHeight);
        return 0;
}
```

ပုံ (၂.၁၆)

၂။ Ex2016.cpp program ကို trace လုပ်ကြည့်မယ်ဆိုရင်

- စစချင်း struct ဆိုတဲ့ keyword ကိုအသုံးပြုပြီး height ဆိုတဲ့ structure name ရှိတဲ့ type တစ်ခုကိုသတ်မှတ်ပေးပါတယ်။ အဲဒီထဲမှာတဲ့ structure member (2) ခုက feet နဲ့ inches ပါ။ main( ) ထဲမှာ myHeight ဆိုတဲ့ type structure height အသစ်တစ်ခုကို create လုပ်ပါတယ်။
- myHeight ရဲ့ feet အရှည်ကိုတောင်းတဲ့အတွက် keyboard ကနေ 6 ကိုရိုက်ထည့်ပေးမယ်ဆိုပါစို့။ inches အရှည်ကိုတောင်းတဲ့အအခါ 5 ကိုထည့်ပေးပါမယ်။ ဒီတော့ myHeight ရဲ့ member value တွေတစ်ခုချင်းကို ကွန်ပျူတာကသိသွားပါပြီ။ myHeight value တွေကို convr( ) function ထဲ pass လုပ်ပေးရအောင်။
- void convr(height h) function header မှာ myHight ကို h ထဲ ကော်ပီကူးထည့်ပေးလိုက်ပါတယ်။ h ကလည်း type structure height ဖြစ်လို့ ကူးထည့်ပေးလို့ရတာပါ။ ဒါဆိုရင် main( ) ထဲက structure တစ်ခု function ထဲကို pass ဝင်သွားပြီလေ။ structure value ကို function ထဲမှာပဲ print ထုတ်လိုက်ပါတယ်။ အဲဒါ ပြီးရင် main( ) function ကို ပြန်လာပြီး program လည်း ပြီးသွားပြီဖြစ်ပါတယ်။ ပုံ (၂.၃၇) ကိုကြည့်ပါ။



ပုံ (၂.၃၇)

Returning Structure Variables

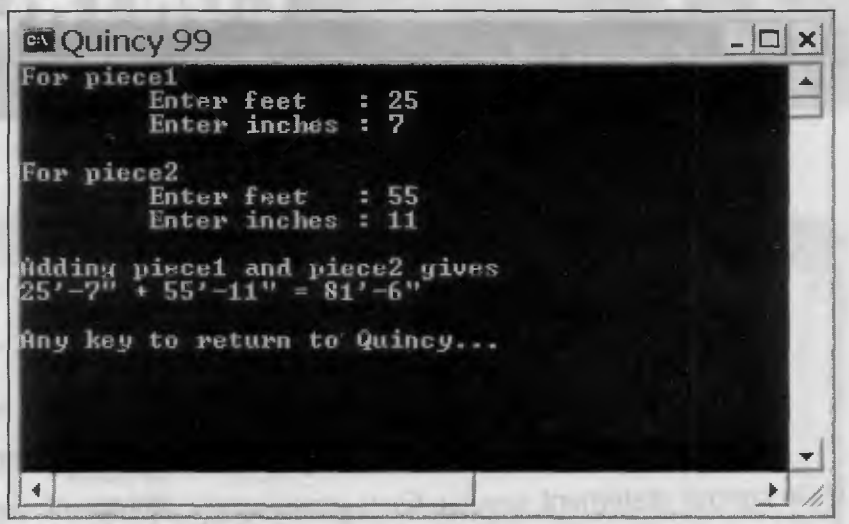
၁။ Ex2017.cpp program ဟာဆိုရင် structure variable တွေကိုအသုံးပြုပြီး length (2) ခုကိုပေါင်းပေးမှာပါ။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

```
// Listing 2.17: This program demonstrates returning a structure

#include <iostream>

struct  Length
{
        int     feet;
        float   inches;
};

void     convr(Length);
Length   add(Length, Length);

int  main( )
{
        Length   piece1,piece2,total;

        cout << "For piece1\n\tEnter feet   : ";
        cin >> piece1.feet;
        cout << "\tEnter inches : ";
        cin >> piece1.inches;

        cout << endl;
        cout << "For piece2\n\tEnter feet   : ";     cin >> piece2.feet;
        cout << "\tEnter inches : ";                 cin >> piece2.inches;

        total = add(piece1, piece2);
        cout << endl;
        cout << "Adding piece1 and piece2 gives" << endl;
        convr(piece1);
        cout << " + ";
        convr(piece2);
        cout << " = ";
        convr(total);
        cout << endl;
        return 0;
}
```

```
Length  add(Length  x, Length  y)
{
        Length  z;
        z.inches = x.inches + y.inches;
        z.feet = 0;
        if (x.inches >= 12.0)
        {
                z.inches -= 12.0;
                z.feet++;
        }
        z.feet += x.feet + y.feet;

        return  z;
}

void   convr(Length   x)
{
        cout << x.feet << "\'-" << x.inches << "\"";
}
```

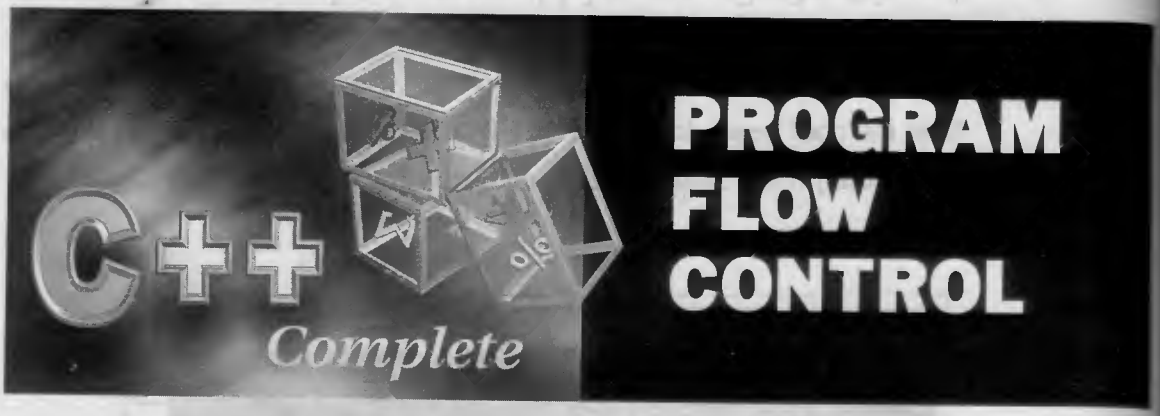
- main( ) ထဲမှာ piece1 ၊ piece2 ၊ total တို့ကို structure type Length ဖြစ်အောင် create လုပ်ပါတယ်။ Length ရဲ့ declaration ကို program ထိပ်ဆုံးနားမှာရေးထားပါတယ်။ piece1 နဲ့ piece2 တို့ရဲ့ member value တွေကိုရိုက်ထည့်ပေးပြီး add( ) function ကို call ခေါ်ပါတယ်။ argument တွေက piece1 နဲ့ piece2 ဆိုတဲ့ structure တွေပါပဲ။
- add( ) function ထဲကို ရောက်လာတဲ့အခါမှာ piece1 ကို x ၊ piece2 ကို y နဲ့ထပ် assign လုပ်ပေးတဲ့အတွက် x.inches ဟာ piece1.inches နဲ့အတူတူပါပဲ။ ဒီတော့ z.inches = x.inches + y.inches = 7+11= 18 ဖြစ်ပါလိမ့်မယ်။ z.feet ကို 0 လို့ initialize လုပ်ထားပါတယ်။ z.inches=18 ဟာ 12 ထက်ကြီးလား ၊ ညီလားလို့မေးပါတယ်။ ဒီ <test> ဟာ True ဖြစ်တဲ့အတွက် if block ထဲဝင်ပြီး z.inches -=12 သို့မဟုတ် z.inches = z.inches -12 = 18-12 = 6 လို့တွက်ယူပါတယ်။ 1 ပေ ပိုထွက်လာတဲ့အတွက် z.feet ထဲမှာပေါင်းထည့်ပေးရမှာပေါ့။ ဒါကြောင့်မို့ z.feet ++ လို့ရေးလိုက်တာပါ။ အခုဆိုရင် z.feet = z.feet + 1 = 0+1 = 1 ဖြစ်သွားပြီ။

■ နောက်တစ်ခါ z.feet = z.feet + x.feet + y.feet ဆိုတဲ့ expression ကနေ 1+25+55 = 81 ကိုတွက်ယူတာဖြစ်ပါတယ်။ အခုဆိုရင် structure z ထဲမှာ z.feet =81 နဲ့ z.inches=6 ကိုရနေပြီမဟုတ်လား။ z ကို return လုပ်ပေးလိုက်တဲ့အတွက် z = add( ) ဖြစ်သွားသလို main( ) ကိုရောက်တဲ့အခါမှာ total = add(piece1, piece2) ဆက်ဖြစ်သွားပါပြီ။ ဒီတော့ total member တွေမှာ total.feet = 81 နဲ့ total.inches = 6 အသီးသီးဖြစ်နေကြမှာပါ။ convr(total) ကနေ အဖြေရိုက်ထုတ်ပေးပါလိမ့်မယ်။ ပုံ (၂.၃၈) ကိုကြည့်ပါ။



ပုံ (၂.၃၈)

Chapter 3



C++ programming မှာ program တစ်ခုရဲ့ program flow ကို control လုပ်ပေးနိုင်တာ ဟာဆိုရင် C++ control statement တွေပါပဲ။ C++ မှာအသုံးများဆုံး statement တွေကတော့

break	**do**	**if**
case	**else**	**return**
continue	**for**	**switch**
default	**goto**	

တို့ဖြစ်ကြပါတယ်။ ဒီအခန်းမှာ အရင်ဆုံးတင်ပြမှာက selection test တွေလုပ်လို့ရတဲ့ if statement ပါပဲ။

၃.၁ The if Statement

၁။ if statement ဆိုတာ conditional statement တစ်ခုဖြစ်ပါတယ်။ if statement ကိုသုံးတဲ့အခါမှာ if နောက်ကကွင်းထဲမှာရှိတဲ့ <test expr> မှန်လားမှားလားဆိုတာကို အရင်စီစစ်ရပါတယ်။ အများအားဖြင့် <test

expr> တွေက relational expression မျိုးတွေကို အသုံးများပါတယ်။ if statement ရဲ့ပုံစံက အခုလိုပါ။

if (<test expr>) <statement1>;

ဒီပုံစံမှာ <test expr> ကိုဖြေရှင်းလိုက်လို့ nonzero ရရင် TRUE ဖြစ်ပါတယ်။ ဒါဆိုရင် program က <statement1> ကို execute လုပ်ပါလိမ့်မယ်။ ပြီးရင်နောက်တစ်ကြောင်းကို ဆင်းသွားမှာပါ။ တစ်ကယ်လို့ <test expr> ရဲ့အဖြေက zero (FALSE) ဆိုရင် <statement1> ကို execute မလုပ်တော့ပါဘူး။ next line ကို တန်းဆင်းသွားမှာပါပဲ။ if နောက်က statement တွေဟာ တစ်ခုထက်ပိုလာမယ်ဆိုလို့ရှိရင် brace{ } တွေနဲ့ ပိတ်ပေးဖို့မမေ့ပါနဲ့။ အဲဒီလိုပိတ်ပေးလိုက်ရင် if statement block ကို single statement တစ်ခုတည်းအနေနဲ့ execute လုပ်သွားမှာဖြစ်ပါတယ်။ if statement နဲ့ပတ်သက်တဲ့ Ex301.cpp program ကို ပုံ (၃. ၁) မှာ ဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex301.cpp

```
// Listing 3.1:This program illustrates the use of if statement.
# include <iostream>

int main()
{
        short int       num;
        cout << "Enter a number\n";
        cin >> num;

        if (num < 0)  cout << "The number is -ive.\n";
        if (num > 0)  cout << "The number is +ive.\n";
        if (num == 0) cout << "The number is zero.\n";
        return 0;
}
```

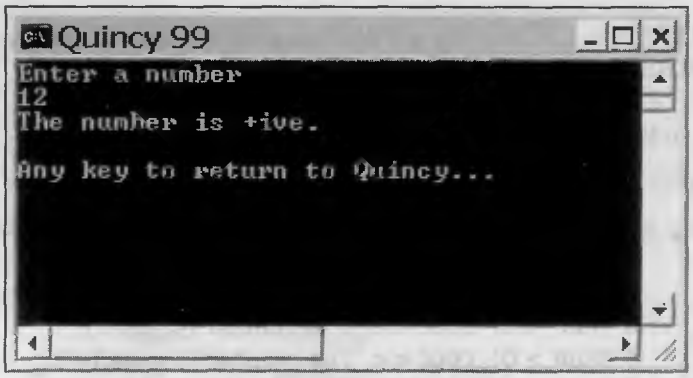
ပုံ (၃. ၁)

၂။ Ex301.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်းမှာ num ကို short integer variable လို့ကြေငြာထားပါတယ်။ num

ရဲ့တန်ဖိုးဟာ -32,768 ကနေ 32,767 အတွင်းမှာရှိနေရပါမယ်။ Enter a number ဆိုတဲ့ prompt ပေါ်လာတဲ့အခါမှာ num အတွက် data ကို 12 ရိုက်ထည့်ပြီး ENTER key နှိပ်မယ် ဆိုရင် num = 12 ဖြစ်သွားပါပြီ။

- ပထမဆုံး if statement နဲ့တွေ့ပါတယ်။ num က သုညထက်ငယ်လားတဲ့ ၊ ဘယ်ငယ်မလဲ။ zero (FALSE) ကိုထုတ်ပေးလိုက်တဲ့အတွက် ကွန်ပျူတာက နောက်တစ်ကြောင်းကိုဆင်းသွားပါပြီ။ ဒုတိယ if statement မှာ num က သုညထက်ကြီးလား မေးပါတယ်။ num = 12 က သုညထက်ကြီးတာမှန်ပါတယ်။ ဒီတော့ကွန်ပျူတာက <test expression> နောက်မှာရှိတဲ့The number is +ive. ဆိုတဲ့စာကြောင်းကို ကွန်ပျူတာမှာပေါ်လာအောင်လုပ်ပေးပါတယ်။ ပြီးတော့ရင် နောက်တစ်ကြောင်းကိုဆက်ဆင်းလာမှာပါ။
- နောက်ဆုံး if statement ကတော့ num ကို သုညနဲ့ညီလားလို့မေးပါတယ်။ မညီပါဘူး။ FALSE ဖြစ်တဲ့အတွက် ကွန်ပျူတာက next line ကို ဆက်ဆင်းသွားပါတယ်။ closing brace (}) နဲ့ တွေ့တဲ့အခါ program ပြီးသွားပါပြီ။ ပုံ (၃. ၂) မှာ program ကို run ပြထားပါတယ်။



ပုံ (၃. ၂)

Conditionally Excuting a Program Block

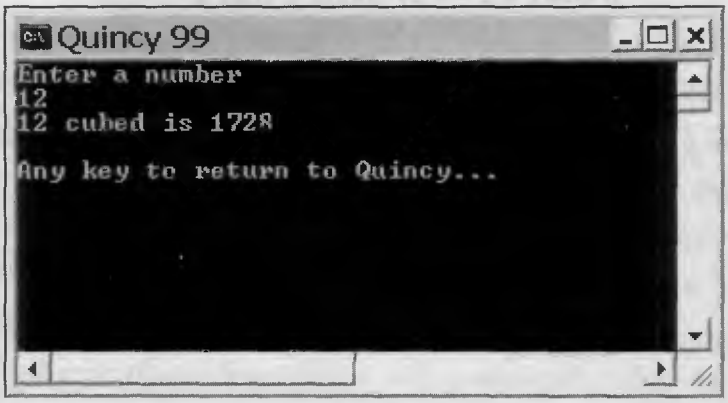
Ex301.cpp program ကိုပြုပြင်ပြီး နောက်တစ်မျိုးရေးလို့ရတာက num ဟာ positive value ဖြစ်မှ num ကိုသုံးထပ်ကိန်းတင်ပြီး အဖြေကို display လုပ်ပြပါလိမ့်မယ်။ မဟုတ်ရင် You chose not to compute the square message ကို display လုပ်ပြခိုင်းထားပါတယ်။

Ex302.cpp

```
// Listing 3.2:This program cubes a number if it is
// not a zero or negative value.

# include <iostream>
# include <cmath>

int main( )
{
        int   num;
        cout << "Enter a number\n";
        cin >> num;
        if (num)
        {
                cout << num << " cubed is " << pow(num,3) << endl;
                return 0;
        }
        cout << "You chose not to compute\n";
        return  0;
}
```

ပုံ (၃.၃)

၂။ ပုံ (၃.၃) မှာရေးထားတဲ့ Ex302.cpp program ကို run ရင် အခုလိုမြင်ရမှာပါ။ ပုံ (၃.၄) ကိုကြည့်ပါ။



ပုံ (၃.၄)

၃.၂ The goto Statement

၁။ goto ဆိုတာ program တစ်ခုရဲ့ ပုံမှန်သွားနေတဲ့လမ်းကြောင်းကနေ တစ်ခြားနေရာကို လွတ်လွတ်လပ်လပ် ခုန်ကျော်သွားချင်တယ်ဆိုရင် အသုံးပြုရမယ့် statement ပါပဲ။ goto ရဲ့ပုံစံက ဒီလိုပါ။

```
<label> : target statement ;
. . . . . .
goto  <label> ;
```

label ဆိုတာ ကွန်ပျူတာသွားချင်တဲ့နေရာဒေသကို ကွန်တော်တို့ အမှတ်သညာပြုထားတဲ့ local label

Ex303.cpp

```
// Listing 3.3: This program  computes the sum of
// odd numbers from Start through Last.

# include  <iostream>
int  main( )
{
        unsigned short int    start,last,odds,sum=0;

        cout << "Enter start and last\n";
        cin >> start >> last;
        if (start % 2 != 0)    odds = start;
        if (start % 2 == 0)   odds = start + 1;
        AGAIN:  if (odds <= last)
                        {
                sum += odds;
                odds += 2;
                goto AGAIN;
        }
        cout << "Sum of odd numbers " << start
             << " through " << last;
        cout << "\nequals " << sum << endl;

        return  0;
  }
```

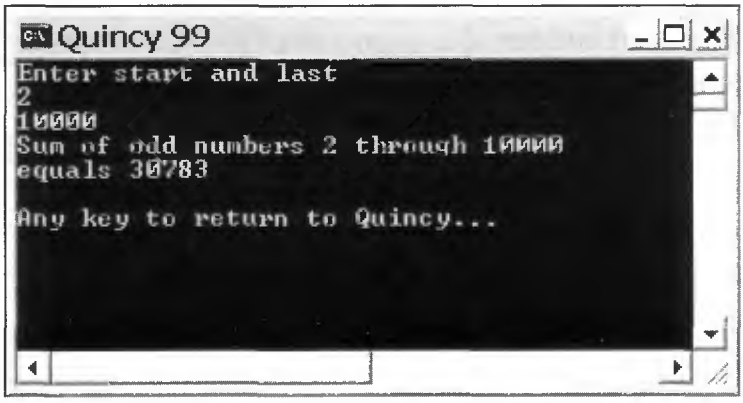
ပုံ (၃.၅)

name ပါ။ target statement မှာ label ကို colon(:) နဲ့တွဲပြီးရေးပေးရပါမယ်။ goto statement ကို အသုံးပြုပုံကို ပုံ (၃. ၅) က Ex303.cpp မှာဖေါ်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

ဒီ program ဟာ start ဆိုတဲ့ စဉ်းဂဏန်းတစ်ခုကနေ last ဆိုတဲ့ နောက်ဆုံးဂဏန်းတစ်ခုအတွင်းက မဂဏန်းတွေအားလုံးကို ပေါင်းပေးမှာပါ။ start နဲ့ last တို့ဟာ ဖြစ်ချင်ရာဂဏန်းတွေဖြစ်နိုင်ပါတယ်။ ဒီ program အလုပ်လုပ်သွားပုံကို လေ့လာကြည့်ရအောင်။

- program မှာအသုံးပြုမယ့် variable (4) ခုကို unsigned short integer တွေလို့ declare လုပ်ပေးပါတယ်။ ဒီတော့ variable တွေရဲ့တန်ဖိုးတွေဟာ 0 ကနေ 65,535 အတွင်းမှာပဲရှိရပါမယ်။ ကျွန်တော်တို့တွက်ချင်တာက မဂဏန်းတွေအားလုံးရဲ့ပေါင်းလဒ်ကို sum ထဲမှာ အဖြေတွက်ထုတ်မှာပါ။ ဘာမှတွက်တာချက်တာမစသေးခင် sum = 0 လို့ initialize လုပ်ထားပါတယ်။
- Enter start and last ဆိုတဲ့ prompt ကိုပေါ်ခိုင်းပါတယ်။ cin >> start >> last ဆိုတဲ့ statement ကြောင့် start နဲ့ last တို့အတွက် data တွေကို keyboard ကနေရိုက်ထည့်ပေးရပါမယ်။ 2 <ကွက်လပ်> 9 သို့မဟုတ် 2 <ENTERr> 9 ကိုရိုက်ထည့်ပြီး ENTER ပုတ်လိုက်မယ်ဆိုရင် start= 2 ၊ last= 9 ဖြစ်သွားပါပြီ။
- အရေးကြီးဆုံးက စဉ်းဂဏန်း start ဟာစုံလား ၊ မလားဆိုတာကို အရင်စီစစ်ရပါမယ်။ စစ်နည်းက start ကို 2 နဲ့စားကြည့်မယ် ၊ မပြတ်ဘူးဆိုရင် start ကမဂဏန်း။ ပြတ်တယ်ဆိုရင် စုံဖြစ်မှာပါ။ start % 2 (start ကို 2 နဲ့စားရင် ကျန်တဲ့အကြွင်း) က သုညနဲ့ညီတာပေါ့။ ဒီတော့ start % 2 != 0 ဆိုတဲ့ <test expression> ဟာ FALSE ဖြစ်သွားပါပြီ။ ဒီ expression ရဲ့နောက်က odds = start ဆိုတဲ့ statement ကိုကွန်ပျူတာက မဖြေရှင်းတော့ပါဘူး။ နောက်တစ်ကြောင်းကိုဆက်ဆင်းသွားမှာပါ။
- ဒုတိယ if ကျတော့ <test expression> က TRUE ဖြစ်သွားပါတယ်။ ဒါကြောင့် odds = start+1 = 2+1 = 3 လို့တွက်ယူလိုက်ပါတယ်။ ဆိုလိုတဲ့အဓိပ္ပါယ်က စဉ်းဂဏန်းဟာ 2 ဖြစ်ပေမယ့် 3 ကိုပဲ စဉ်းဂဏန်းလို့ ကွန်ပျူတာကပြောင်းယူလိုက်တာပါ။ odds နဲ့ last ကိုယှဉ်လိုက်လို့ငယ်တယ် သို့မဟုတ် ညီတယ်ဆိုရင် if နောက်က block statement ကို execute လုပ်ပါလိမ့်မယ်။ အခုလည်းပဲ odds = 3 က last = 9 ထက်ငယ်တာမို့ <test expr> က TRUE ပေါ့။ if block ထဲကိုဝင်လာပါပြီ။ sum += odds ဆိုတာကို sum = 0+3 = 3 လို့ ကွန်ပျူတာကတွက်လိုက်ပါတယ်။ အဓိပ္ပါယ်က sum ထဲကို ပထမဆုံးမဂဏန်းတစ်လုံး ထည့်လိုက်တာပါ။ 3 ပြီးရင် ဒုတိယမဂဏန်းဟာ 5 မဟုတ်လား။ အဲဒီဥစ္စာရဖို့အတွက် odds += 2 ကိုအသုံးပြုပြီး odds = odds+2 = 3+2 = 5 ဆိုတဲ့ ဒုတိယမဂဏန်းကိုတွက်ယူပါတယ်။

- goto statement ဟာ AGAIN ဆိုတဲ့ label name ရှိတဲ့နေရာကို ပြန်သွားခိုင်းနေတာပါ။ အခုဆိုရင်ကွန်ပျူတာဟာ block statement အတွင်းမှာ ထပ်ခါတစ်လဲပတ်နေမှာဖြစ်ပါတယ်။ တစ်ချိန်မှာ odds =11 ဖြစ်သွားပါလိမ့်မယ်။ ဒါဆိုရင် odds =11 က last = 9 ထက်ကြီးသွား တာကြောင့် <test expression> ဟာ TRUE ဖြစ်သွားပါပြီ။ if block ထဲကိုထပ်မဝင်တော့ပဲ အောက်ကိုဆင်းလာပြီလေ။
- နောက်ဆုံးမှာ Sum of odd numbers 2 through 9 equals 24 ဆိုတဲ့ output စာကြောင်း ကို (2) ကြောင်းခွဲပြီး အဖြေထုတ်ပေးပါလိမ့်မယ်။ ဒါဆိုရင် Ex303.cpp program ကပြီးသွား ပါပြီ။ ပုံ (၃. ၆) မှာပြထားတဲ့အတိုင်း start =2 နဲ့ last =10000 လို့ data သွင်းခဲ့ရင် အ ကို အခုလိုမြင်ရမှာပါ။



ပုံ (၃. ၆)

Reversing an Integer Number

၁။ Ex304.cpp program ရဲ့ရည်ရွယ်ချက်က ကိန်းပြည့်ဂဏန်းတစ်လုံးကို input data အနေနဲ့ ထည့ လိုက်မယ်ဆိုရင် အဲဒီဂဏန်းကို ပြောင်းပြန်အစီအစဉ်နဲ့ဖြစ်အောင် ကွန်ပျူတာကလုပ်ပေးမှာပါ။ ပုံ (၃. ၇) မှာ statement တွေကိုရေးပြထားပါတယ်။

၂။ Ex304.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ unsigned long int num, rev, q, r လို့ရေးထားတဲ့အတွက် ဒီ variab

Ex304.cpp

```
// Listing 3.4: This program reads an integer number
// and prints out the same number with digits reversed.

#include <iostream>

int main()
{
        unsigned long int       num, rev, q, r;

        cout << "Enter a number\n";
        cin >> num;
        q = num /10;
        rev = num % 10;
        START:  if ( q != 0)
                {
                        num = q;
                        q = num /10;
                        r = num % 10;
                        rev = rev * 10 + r;
                        goto START;
                }
        cout << "The number reversed is\n" << rev << endl;
        return 0;
}
```

ပုံ (၃.၇)

တွေရဲ့တန်ဖိုးကို 0 ကနေ 4,264,967,295 အတွင်းမှာ ကြိုက်တဲ့ဂဏန်းနဲ့ assign လုပ်လို့ရပါပြီ။ Enter a number ဆိုတဲ့ prompt ပေါ်လာပြီး cin >> num ကြောင့် num အတွက် data ကိုတောင်းနေပါပြီ။ keyboard ကနေ 123456789 ကိုရိုက်ထည့်ပေးလိုက်မယ်ဆိုရင် num = 123456789 ဖြစ်သွားမှာပါ။

- q တန်ဖိုးကို q = num/10 =123456789/10 = 12345678 လို့တွက်ယူပါတယ်။ rev = num % 10 = 123456789 % 10 = 9 ပေါ့။ if (q != 0) ဆိုတဲ့ <test expr> ကို ကွန်ပျူတာက TRUE လို့အဖြေထုတ်ပါတယ်။ q က သုညမှမဟုတ်တာကိုး။ ဒီတော့ if block ထဲဝင်လာပြီး ကွန်ပျူတာကအခုလိုဖြေရှင်းပါတယ်။

num = q = 12345678

q = num / 10 = 12345678 / 10 = 1234567

r = num % 10 = 12345678 % 10 = 8

rev = rev*10 + r = 9*10 + 8 = 98

- goto START ကြောင့် ကွန်ပျူတာက START label ရှိတဲ့နေရာကို ပြန်သွားပါတယ်။ q က သုညမဟုတ်တဲ့အတွက် if block ထဲကို ကွန်ပျူတာက နောက်တစ်ကြိမ်ပြန်ဝင်လာပြီလေ။ if block ထဲမှာ အခုလိုဖြေရှင်းပါတယ်။

num = q = 1234567

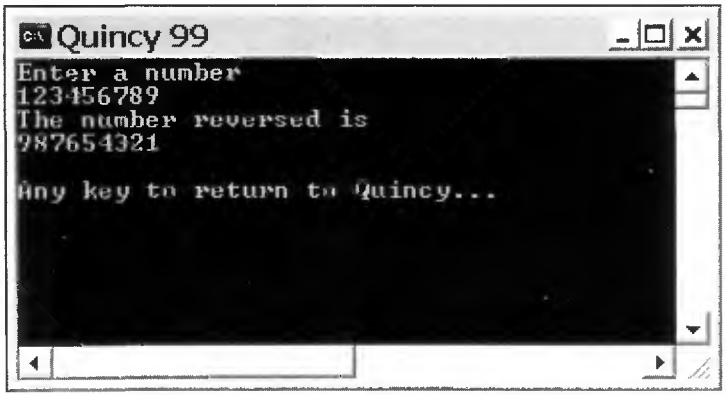
q = num / 10 = 1234567 / 10 = 123456

r = num % 10 = 1234567 % 10 = 7

rev = rev*10 + r = 98*10 + 7 = 987

goto START ကိုပြန်သွားပါ။

- တစ်ချိန်မှာ q = 0 ဖြစ်သွားပြီး if block ထဲကထွက်လာမှာပါ။ နောက်ဆုံးမှာ The number reversed is : 987654321 လို့ တစ်ကြောင်းစီခွဲပြီး အဖြေရိုက်ထုတ်ပေးပါလိမ့်မယ်။ ဒါဆိုရင် program ပြီးသွားပါပြီ။ အဖြေကို ပုံ (၃. ၈) မှာ program run ပြထားပါတယ်။



ပုံ (၃. ၈)

A Payroll Program

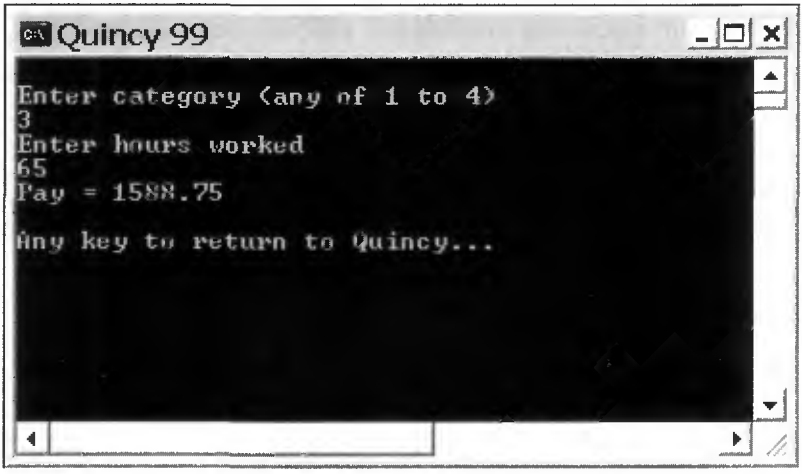
၁။ Ex305.cpp program ရဲ့ရည်ရွယ်ချက်က အလုပ်သမားတစ်ယောက်ရဲ့လုပ်ချိန်နဲ့ လုပ်အားခနှုန်းကိုသိရင် လစာကိုတွက်ပေးတဲ့ program ဖြစ်ပါတယ်။ အလုပ်ချိန်နာရီ (40) ကျော်ရင်အချိန်ပိုနာရီကို ပုံမှန်လုပ်အားခနှုန်းရဲ့ တစ်ဆခွဲနဲ့တွက်ပေးမှာပါ။ ပုံ (၃.၉) မှာ code statement တွေကိုရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex305.cpp

```
// Listing 3.5: This program calculates the payroll for
// different workers giving overtime if the hours
// worked are greater than 40 hours.
#include <iostream>

int main()
{
        int     category;
        float   wage, hours, pay;

        START : cout << "\nEnter category (any of 1 to 4)\n";
        cin >> category;
        if (category < 1 || category > 4)
                goto START;
        if (category == 1)  wage = 12.5;
        if (category == 2)  wage = 15.5;
        if (category == 3)  wage = 20.5;
        if (category == 4)  wage = 25.5;
        cout << "Enter hours worked\n";
        cin >> hours;
        if (hours <= 40)
                pay = wage*hours;
        if (hours > 40)
                pay = wage*40 + 1.5*wage*(hours-40);
        cout << "Pay = " << pay << endl ;
        return 0;
}
```

ပုံ (၃.၉)

၂။ ဒီ program ကို ကွန်တော်ရှင်းမပြတော့ပါဘူး။ run ထားတာကို ပုံ (၃. ၁၀) မှာဖော်ပြထားပါတယ်။ စာဖတ်သူကိုယ်တိုင် data ပြောင်းထည့်ပြီး run ကြည့်ပါ။ program လမ်းကြောင်း trace လုပ်ကြည့်ပါ၊ မခက်ပါဘူး။



ပုံ (၃. ၁၀)

၃.၃ The if-else Statement

၁။ program တစ်ခုရေးတဲ့အခါမှာ ရွေးစရာလမ်းကြောင်း (2) ခုရှိလာရင် if-else statement ကိုသုံးလို့ရပါတယ်။ if နောက်က <test expr> ဟာမှန်တယ်ဆိုရင် <statement1> ကိုဖြေရှင်းပြီး if-else အပြင်ကိုထွက်သွားမှာဖြစ်ပါတယ်။ <test expr> မှားနေရင် <statement2> ကို execute လုပ်မှာပါ။ statement တွေဟာ တစ်ကြောင်းထက်ပိုလာရင် brace { } တွေနဲ့ပိတ်ပေးရပါမယ်။ if-else statement ရဲ့ပုံစံက အခုလိုပါ။

```
if ( <test expr>)
        <statement1>;
else
        <statement2>;
```

ကောင်းပြီ၊ ပုံ (၃. ၁၁) မှာဖော်ပြထားတဲ့ Ex306.cpp program ကနေ if-else statement အသုံးပြု

နည်းကိုလေ့လာကြည့်ရအောင်။ ဒီ program ဟာ lowercase letter တစ်ခုကို uppercase ဖြစ်အောင်ပြောင်းပေးတဲ့ program ဖြစ်ပါတယ်။

Ex306.cpp

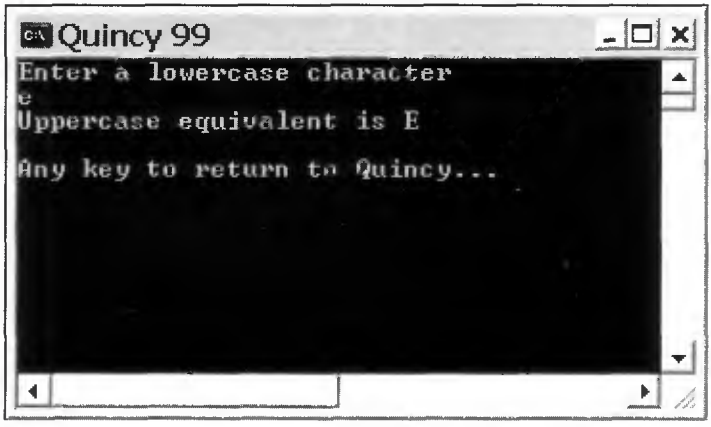
```
// Listing 3.6: This program reads a lowercase character,
// converts it to uppercase and then writes out
// the uppercase eqivalent.
#include <iostream>

int main()
{
        char    lower, upper;

        cout << "Enter a lowercase character\n";
        cin >> lower;
        if (lower >= 'a' && lower <= 'z')
                upper = 'A' + lower - 'a';
        else
                upper = lower;
        cout << "Uppercase equivalent is " << upper << endl;
        return 0;
}
```

ပုံ (၃. ၁၁)

၂။ Ex306.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်းမှာ lower နဲ့ upper ဆိုတဲ့ variable (2) ခုကို character တွေလို့ကြေငြာပါတယ်။ Enter a lowercase character ဆိုတဲ့ prompt တစ်ခု ကွန်ပျူတာမှာ ပေါ်လာတဲ့အခါ lower အတွက် data ကို e လို့ရိုက်ထည့်ပါမယ်။ ဒါဆိုရင် lower = 'e' ဖြစ်သွားပါပြီ။
- if (lower >= 'a' && lower <='z') ဆိုတဲ့ relational expression မေးခွန်းမှာ lower က 'a' ထက်ကြီးပြီး 'z' ထက်ငယ်ပါတယ်။ ဒီတော့အဖြေက TRUE ပေါ့။ upper = 'A'+lower-'a' ဖြစ်သွားပါပြီ။ upper ရဲ့ ASCII တန်ဖိုးကို upper = 65+101-97 =69 လို့တွက်လို့ရပါတယ်။ ASCII တန်ဖိုး 69 က uppercase letter E ကိုဆိုလိုတာပါ။ ပြီးရင် else အောက်ကလိုင်းကို ကျော်ဆင်းသွားပါပြီ။

■ cout << "Uppercase equivalent is :\n" << upper ဆိုတဲ့ statement ကြောင့် အဖြေ E ထွက်လာတာပေါ့။ ပုံ (၃.၁၂) မှာ program run ပြထားတာကိုကြည့်ပါ။ တစ်ကယ်လို့ keyboard ကနေ E ကို ရိုက်ထည့်ပေးမယ်ဆိုရင် ကွန်ပျူတာ ဘယ်လိုအလုပ်လုပ်မလဲ။ အဲဒါကို စာဖတ်သူကိုယ်တိုင် လက်တွေ့အဖြေရှာကြည့်ပါ။



ပုံ (၃.၁၂)

A Temperature Conversion Program

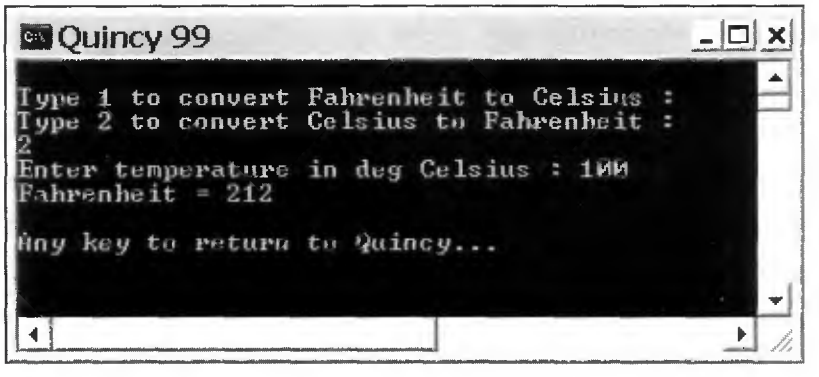
၁။ Ex307.cpp program ဟာ degree Fahrenheit ကနေ Celsius ကိုပြောင်းပေးနိုင်သလို degree Celsius ကနေ Fahrenheit ကိုလည်းပြောင်းပေးနိုင်တဲ့ program တစ်ခုပါပဲ။ flag အတွက် data ကို 1 ရိုက်ထည့်ပေးမယ်ဆိုရင် degree Celsius ကနေ Fahrenheit ကိုလည်းပြောင်းပေးပါလိမ့်မယ်။ 2 ရိုက်ထည့်ပေးရင်တော့ degree Fahrenheit ကနေ Celsius ကိုလည်းပြောင်းပေးမှာပါ။ ဒီအတွက် source code တွေကို ပုံ (၃.၁၃) မှာရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

၂။ ပုံ (၃.၁၄) မှာ program run ပြထားပါတယ်။ flag အတွက် data ကို 1 ရိုက်ထည့်ပေးပြီး degree Celsius = 100 ကိုရိုက်ထည့်ပေးတာနဲ့ Fahrenheit = 212 လို့ display လုပ်ပြနေပါပြီ။

Ex307.cpp

```
// Listing 3.7: A program that gives the user the option of
// converting Fahrenheit to Celsius or Celsius to Fahrenheit.
#include <iostream>

int main()
{
        int         flag;
        double      tempr;

        cout << "\nType 1 to convert Fahrenheit to Celsius : "
             << "\nType 2 to convert Celsius to Fahrenheit : "
             << endl;
        cin >> flag;

        if (flag == 1)   {
                cout << "Enter temperature in deg Fahrenheit : ";
                cin >> tempr;
                cout << "Celsius = " << (tempr-32)/1.8 << endl;
        }
        else    {
                cout << "Enter temperature in deg Celsius : ";
                cin >> tempr;
                cout << "Fahrenheit = " << 1.8*tempr+32 << endl;
        }
        return 0;
}
```

รูป (๒.๐๒)



รูป (๒.๐๑)

၃.၄ Nested if Statement

၁။ nested if-else statement ဟာ if statement အောက်မှာပဲအကျုံးဝင်ပါတယ်။ ရှေ့ပိုင်းမှာတုန်းက ကျွန်တော်တို့လေ့လာခဲ့တဲ့ if-else statement တွေမှာ ခွဲးစရာလမ်း (2) ခုပဲရှိတယ်လို့ပြောခဲ့ပါပြီ။ if နောက်က <test expr> ဟာ TRUE ဆိုရင်သူ့နောက်က <statement> ကိုဖြေရှင်းမယ် ၊ FALSE ဆိုရင် else နောက်က အလုပ်ပဲလုပ်မှာပါ။ ဒါကြောင့်မို့သွားစရာလမ်း (2) ခုပဲရှိတာလို့ပြောတာပါပဲ။ တစ်ကယ်လို့ သွားစရာလမ်းတွေ အများကြီးကို create လုပ်ချင်ရင် nested if-else statement တွေကိုအသုံးပြုရင်ရပါတယ်။ မှားမှာစိုးရင် brace တွေကို ပိုပိုလိုလို ထည့်ပေးပါ။ ပုံ (၃. ၁၅) မှာဖော်ပြထားတဲ့ Ex308.cpp program ဟာဆိုရင် complex nested if statement တွေ အသုံးပြုပုံပြုနည်းကို နမူနာရေးပြထားတာဖြစ်ပါတယ်။

၂။ Ex308.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

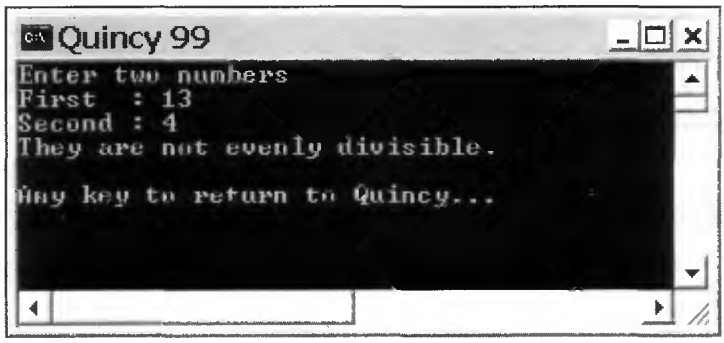
- program စစချင်းမှာ num1 နဲ့ num2 တို့ကို integer variable တွေလို့ကြေငြာပါတယ်။ Enter two numbers လို့ prompt ပေါ်လာတဲ့အခါကျရင် num1 တန်ဖိုးကို 13 လို့ရိုက်ထည့်ပေးမယ်ဆိုပါစို့။ ဒါဆိုရင် num1 = 13 ဖြစ်သွားပါပြီ။ num2 တန်ဖိုးကိုတောင်းတဲ့အခါမှာ 4 ကိုရိုက်ထည့်ပေးပါမယ်။
- program က အပြင်ဘက်အကျဆုံး if statement မှာ num1= 13 က num2= 4 ထက်ကြီးလား ၊ ညီလားလို့မေးပါတယ်။ <test expr> ဟာ TRUE ဖြစ်တဲ့အတွက် သူ့အောက်က if statement ကိုဆက်ပြီး execute လုပ်ပါတယ်။ အလယ်ပိုင်းကျတဲ့ if statement မှာလည်း num1 = 13 က num2 = 4 နဲ့ ညီလားလို့ထပ်မေးပါတယ်။ ဒီ <test expr> က FALSE ဖြစ်နေပါတယ်။ ဒီတော့ အတွင်းဘက်အကျဆုံး if-else ကိုကျော်ပြီး ဒုတိယ if-else ရဲ့နောက်က <statement> ကို execute လုပ်ပါပြီ။ They are not evenly divisible. ဆိုတဲ့စာကြောင်းကွန်ပျူတာမှာ display လုပ်ပြပါလိမ့်မယ်။
- အခုဆိုရင် အပြင်ဘက်အကျဆုံး if-else ရဲ့အလုပ်ပြီးသွားပါပြီ။ ပြီးရင် closing brace (}) ဆီကိုရောက်သွားတဲ့အတွက် program ပြီးသွားပါပြီ။

၃။ ပုံ (၃. ၁၆) မှာ Ex308.cpp program ကို run ပြထားပါတယ်။ num1 = 13 ၊ num2 = 4 ဖြစ်ရင် They are not evenly divisible. ဆိုတဲ့ message ကို display လုပ်ပြမှာပါ။ num1 = 16 ၊ num2 = 4 ဖြစ်ရင် ဘယ်လို message ကို display လုပ်ပြမလဲ ၊ ခန့်မှန်းကြည့်ပါ။ ပြီးရင် program run ကြည့်ပြီး အဖြေတိုက်လို့ရပါတယ်။

Ex308.cpp

```
// Listing 3.8: This program demonstrates a complex,
// nested if statement
#include <iostream>

int main()
{
        int  num1, num2;

        cout << "Enter two numbers\nFirst  : ";
        cin >> num1;
        cout << "Second : ";
        cin >> num2;
        if (num1 >= num2)
        {
                if (num1 % num2 == 0)
                {
                        if (num1 == num2)
                                cout << "They are the same.\n";
                        else
                                cout << "They are evenly divisible.\n";
                }
                else
                        cout << "They are not evenly divisible.\n";
        }
        else
                cout << "The second one is larger.\n";
        return 0;
}
```

ပုံ (၃.၁၄)



ပုံ (၃.၁၅)

၄။ Ex308.cpp program မှာ brace တွေကိုဖြုတ်ပြီး ပုံ (၃. ၁၇) မှာပြထားတဲ့အတိုင်း ပြင်ရေးမယ်ဆိုရင်လည်း run လို့ရပါတယ်။ အဖြေလည်းအတူတူရမှာပါ။ စာဖတ်သူကိုယ်တိုင်လက်တွေ့ run ကြည့်ပါ။

Ex308A.cpp

```
#include <iostream>

int main()
{
        int  num1, num2;

        cout << "Enter two numbers\nFirst  : ";
        cin >> num1;
        cout << "Second : ";
        cin >> num2;

        if (num1 >= num2)
                if (num1 % num2 == 0)
                        if (num1 == num2)
                                cout << "They are the same.\n";
                        else
                                cout << "They are evenly divisible.\n";
                else
                        cout << "They are not evenly divisible.\n";
        else
                cout << "The second one is larger.\n";
        return 0;
}
```

ပုံ (၃. ၁၇)

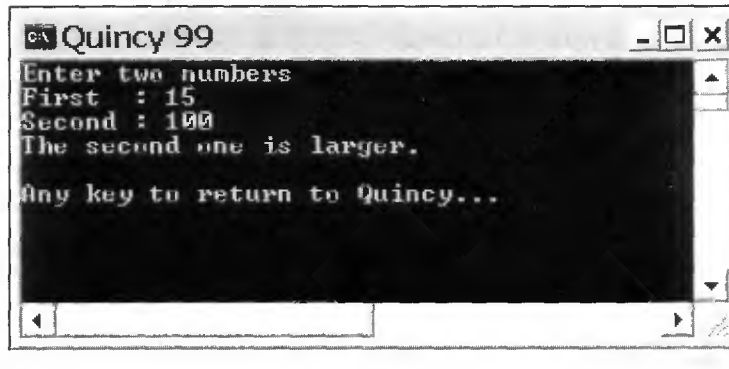
၅။ ဒါမှမဟုတ် program logic ကိုအခုလိုပြင်ရေးပြီး run ရင်လည်း အဖြေအတူတူရမှာပါ။ ဒီ program က compact ပိုဖြစ်တယ်လို့ထင်ပါတယ်။ ပုံ (၃. ၁၈) ကိုကြည့်ပါ။

၆။ Ex308B.cpp program ကို run ကြည့်ရင် ပုံ (၃. ၁၉) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ First = 15 နဲ့ Second = 100 လို့ data ထည့်ပေးရင် The second one is larger. ဆိုတဲ့အဖြေကို display လုပ်ပြပါလိမ့်မယ်။

Ex308B.cpp

```
#include <iostream>

int main()
{
        int  num1, num2;

        cout << "Enter two numbers\nFirst  : ";
        cin >> num1;
        cout << "Second : ";
        cin >> num2;

        if (num1 < num2)
                cout << "The second one is larger.\n";
        else if (num1 % num2 != 0)
                cout << "They are not evenly divisible.\n";
        else if (num1 != num2)
                cout << "They are evenly divisible.\n";
        else
                cout << "They are the same.\n";

        return 0;
}
```

ပုံ (၃.၁၈)



ပုံ (၃.၁၉)

Leap Year Test

၁။ အခုတင်ပြမယ့် Ex309.cpp program ဟာဆိုရင် အင်္ဂလိပ် ဘယ်ခုနှစ်ဟာ ဝါထပ်လား ၊ မထပ်ဘူးလား ဆိုတာကို စီစစ်ပေးမှာပါ။ စစ်နည်းက ခုနှစ်ကို 4 နဲ့အရင်စားကြည့်ပါ။ မပြတ်ဘူးဆိုရင် အဲဒီနှစ်ကဝါမထပ်ပါဘူး။ 4 နဲ့စားလို့ပြတ်ရင် ဝါထပ်ချင်ထပ်နိုင်ပါတယ်။ ထပ်စစ်ရပါဦးမယ်။ 100 နဲ့ထပ်စားကြည့်ပါ။ စားလို့မပြတ်ရင် အဲဒီနှစ်ဟာ သေချာပေါက် ဝါထပ်ပါတယ်။ ပြတ်တယ်ဆိုရင် အဲဒီနှစ်ဟာဝါထပ်လား၊ မထပ်လားမသေချာပြန်တော့ဘူး။ 400 နဲ့ ထပ်စားကြည့်ပါဦး။ ပြတ်တယ်ဆိုရင်ဝါထပ်ပါတယ်။ မပြတ်ရင်ဝါမထပ်ပါဘူး။ အဲဒီနည်းနဲ့ program ကိုရေးထားတာ ဖြစ်ပါတယ်။ ပုံ (၃. ၂၀) က code statement တွေကိုလေ့လာကြည့်ပါ။

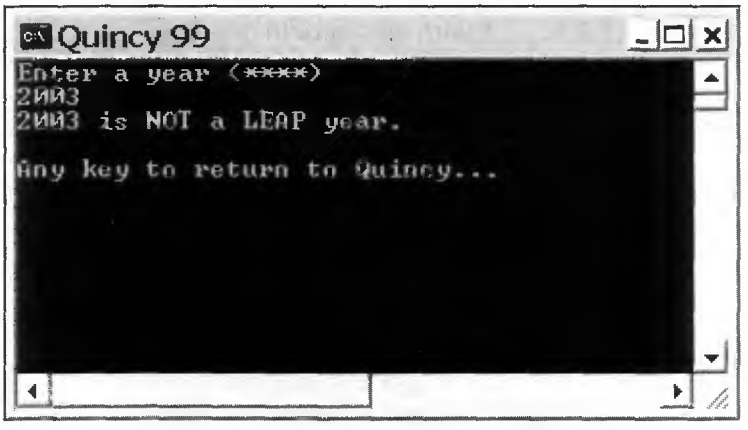
Ex309.cpp

```
// Listing 3.9: This program reads in a number representing
// a year and then determines whether it is a leap year or not.

#include <iostream>
const  int   TRUE=1;
const  int   FALSE=0;

int main()
{
        int      year, leap;

        cout << "Enter a year (****)\n";
        cin >> year;

        if (year % 4 != 0)              leap = FALSE;
        else if (year % 100 != 0)       leap = TRUE;
        else if (year % 400 == 0)       leap = TRUE;
        else leap = FALSE;

        if (leap)          cout << year << " is a LEAP year.\n";
        else               cout << year << " is NOT a LEAP year.\n";

        return 0;
}
```

ပုံ (၃. ၂၀)

Ex309.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- 1998 ခုနှစ်ဟာ ဝါမထပ်ပါဘူး။ ဘာပြုလို့လဲဆိုတော့ 1998 ကို 4 နဲ့စားရင်မပြတ်လို့ပါပဲ။ 2000 ခုနှစ်က ဝါထပ်ပါတယ်။ ဘာပြုလို့လဲဆိုတော့ 2000 ကို 4 နဲ့စားတာပြတ်တယ်။ 100 နဲ့ထပ်စားတာပြတ်သေးတယ်။ ဒီတော့ 400 နဲ့ထပ်စားတယ်။ ပြတ်သေးတာပဲ။ ဒါဆိုရင် 2000 ခုနှစ်ဟာ သေချာဝါထပ်တာပေါ့။
- 5000 ခုနှစ်ကျတော့ ဝါမထပ်တော့ဘူး။ ဘာပြုလို့လဲဆိုတော့ ပထမ 4 နဲ့စားတာပြတ်တော့ 100 နဲ့ထပ်စားတော့ပြတ်တယ်။ ဒါပေမယ့် 400 နဲ့စားတော့မပြတ်တော့ဘူး။ ဒါကြောင့်မို့ 5000 ခုနှစ်ဟာ ဝါမထပ်ဘူးလို့ပြောတာပါ။ ဒီစစ်နည်းတွေမှာ nested if-else statement တွေကို အသုံးပြုထားပါတယ်။ Ex309.cpp program ကို run ကြည့်ရင် ပုံ (၃. ၂၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ year= 2003 လို့ထည့်ပေးရင် 2003 is NOT a LEAP year. ဆိုတဲ့အဖြေကို display လုပ်ပြပါလိမ့်မယ်။



ပုံ (၃. ၂၁)

၃.၅ The while Statement

၁။ C++ program မှာ loop တစ်ခုကို ထပ်ခါတစ်လဲပတ်ရတော့မယ်ဆိုလို့ရှိရင် while statement ကို အသုံးပြုလို့ရပါတယ်။ while statement ရဲ့ပုံစံက အခုလိုပါ။

while (<test expr>) <statement1> ;

ဒီပုံစံဟာဆိုရင် ကွန်ပျူတာမှာ စစချင်း while ရဲ့နောက်ကကွင်းထဲက <test expr> ကို test လုပ်ပါတယ်။ သူ့ရဲ့တန်ဖိုးက nonzero (TRUE) ဖြစ်ခဲ့ရင် ကွန်ပျူတာက <statement1> ကို execute လုပ်မှာပါ။ တစ်ကယ်လို့ <test expr> ရဲ့တန်ဖိုးက zero (FALSE) ဖြစ်သွားမယ်ဆိုရင် while loop ကို execute မလုပ်တော့ပါဘူး။ statement တွေကအများကြီးဆိုရင် brace တွေနဲ့ပိတ်ပေးဖို့မမေ့ပါနဲ့။

၂။ အခုတင်ပြမယ့် *Ex3010.cpp program* ဟာဆိုရင် *1* ကနေ *9* အထိဂဏန်း *(9)* လုံးကို စာတစ်ကြောင်းတည်းမှာပေါ်လာအောင် while statement အသုံးပြုပြီးရေးထားတာဖြစ်ပါတယ်။ ဂဏန်းတစ်လုံးနဲ့တစ်လုံးကြားမှာ space တစ်ခုခြားပေးထားပါတယ်။ ပုံ (၃. ၂၂) မှာရေးပြထားတဲ့ code statement တွေကိုလေ့လာကြည့်ပါ။

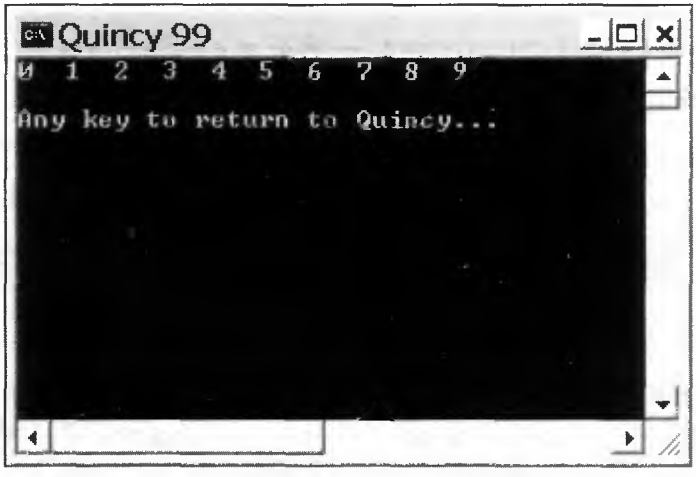
Ex3010.cpp

```
// Listing 3.10:  This program displays the consecutive
// digits 0,1,2,3, ... ,9 with all digits in one line
// separated by one space between them.

#include <iostream>

int main()
{
        int     digit = 0;

        while (digit <= 9)
        {
                // Print a digit followed by a space
                cout << digit << " ";
                ++ digit;
        }

        cout << endl;
        return 0;
}
```

ပုံ (၃. ၂၂)

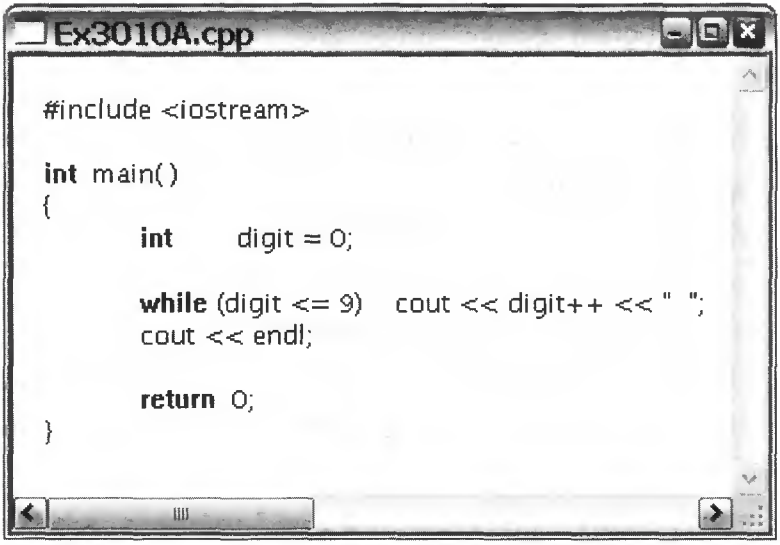
Ex3010.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်းမှာ digit ကို integer variable လို့ကြေငြာပြီး သုညဖြစ်အောင် initialize လုပ်ထားပါတယ်။ while (digit <= 9) ရဲ့အဓိပ္ပါယ်က digit ဟာ 9 ထက်ငယ်မယ် ၊ ညီမယ် ဆိုရင် brace { } တွေထဲက multiple statement တွေကို execute လုပ်မှာပါ။ digit=0 ဖြစ်တဲ့အတွက် ကွန်ပျူတာက while loop ထဲဝင်လာပါပြီ။
- cout << digit << " "; ဆိုတဲ့ statement ကြောင့်ကွန်ပျူတာမှာ 0 <ကွက်လပ်> လို့ print လုပ်တာကိုမြင်ရပါလိမ့်မယ်။ ++digit လို့ရေးထားတဲ့အတွက် digit တန်ဖိုးကို ကွန်ပျူတာက 1 တိုးပေးလိုက်ပါပြီ။ ပြီးရင် closing brace (}) ကိုတွေ့တဲ့အခါမှာ while loop အစကို ပြန်တက်သွားမှာပါ။
- အခုတစ်ခါ digit = 1 ဖြစ်နေပါတယ်။ digit = 1 က 9 ထက်ငယ်ပါတယ်။ ဒီတော့ while loop ထဲ ဒုတိယအကြိမ်ပြန်ဝင်ပြီး 1 <ကွက်လပ်> ကိုဆက်ပြီး print လုပ်ပါတယ်။ ပြီးရင် digit ကို တစ်တိုးပါတယ်။ ဒီနည်းအတိုင်း digit တန်ဖိုး 9 မကျော်မချင်း while loop ကိုပတ်နေမှာပါပဲ။ တစ်ချိန်မှာ digit = 10 ဖြစ်သွားတဲ့အခါ loop ထဲက အပြီးထွက်လာပြီးတော့ program လည်း ရပ်သွားပါလိမ့်မယ်။ program run ထားတာကို ပုံ (၃. ၂၃) မှာဖော်ပြထားပါတယ်။



ပုံ (၃. ၂၃)

၄။ ဒီ program ကို ပိုပြီးကျစ်ကျစ်လစ်လစ်ဖြစ်အောင် အခုလိုပြောင်းရေးလို့ရပါတယ်။ (++) increment operator သုံးသွားပုံကို ပုံ (၃. ၂၄) က Ex3010A.cpp program ကိုလေ့လာကြည့်ပါ။



ပုံ (၃. ၂၄)

Evaluating PI Series

၁။ အခုတင်ပြမယ့် Ex3011.cpp program ဟာဆိုရင် π ရဲ့တန်ဖိုးကို π/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - ..... series ကိုအသုံးပြုပြီး term တွေ အများကြီးပေါင်းခိုင်းမှာဖြစ်ပါတယ်။ ပုံ (၃. ၂၅) မှာရေးထားတဲ့ program ကိုလေ့လာကြည့်ပါ။ ဒီ series က slow converging ဖြစ်ပေမယ့် C++ compiler ဟာမြန်တဲ့အတွက် term (3) သန်းအထိ ပေါင်းခိုင်းတာကို အလွယ်တကူတန်းပေါင်းပေးနိုင်ပါတယ်။ ဒီ program ဘယ်လိုအလုပ်လုပ်သွားလဲဆိုတာ စာဖတ်သူကိုယ်တိုင် trace လုပ်ကြည့်ပါ။ while statement ကိုပိုနားလည်သွားပါလိမ့်မယ်။

၂။ Ex3011.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် စစချင်းမှာ pi = 0 လို့ initialize လုပ်ပြီး n = 3000000 ကိုရိုက်ထည့်ပေးပါတယ်။ term တွေကိုပေါင်းတဲ့အခါမှာ (2) စုံချင်းပေါင်းသွားပေးပါတယ်။ ပထမအစုံအတွက် counter (i) ကို 1 လို့သတ်မှတ်ပေးရင် ဒုတိယအစုံအတွက် counter (i) ကို i + 4 = 5 လို့တွက်ပေးရပါမယ်။ နောက်အစုံတွေကျရင်လည်း (4) စီတိုးသွားမှာပါပဲ။ ပုံ (၃. ၂၆) မှာ program run ပြထားပါတယ်။ pi = 3.141592 ရပါတယ်။ အဖြေမှန်က π = 3.141592 ဖြစ်ပါတယ်။

Ex3011.cpp

```
// Listing 3.11:  This program evaluates the value
// of PI using the series
//       PI/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - ...

#include <iostream>
int  main()
{
        double    n, i=1, pi=0;

        cout << "Enter n : ";
        cin >> n;
        while (i <= n)
        {
                pi += 1/i - 1/(i+2);
                i += 4;
        }
        cout.precision(10);
        cout << "PI = " << 4*pi << endl;
        return  0;
}
```

ဥပ (၃.၂၅)

Quincy 99

```
Enter n : 3000000
PI = 3.141591987

Any key to return to Quincy...
```

ဥပ (၃.၂၆)

Counting Characters and Words

၁။ အခုတင်ပြမယ့် Ex3012.cpp program ဟာဆိုရင် keyboard ကနေ ကျွန်တော်တို့ရိုက်ထည့်လိုက်တဲ့စာကြောင်းထဲမှာ character ဘယ်နှစ်လုံး ၊ word ဘယ်နှစ်လုံးပါလဲဆိုတာ ရေတွက်ပေးမယ့် program ပါပဲ။ ဒီ program မှာ getche( ) function ကိုအသုံးပြုလို့ရအောင် <conio.h> header ဖိုင်ကို ထည့်ပေးထားပါတယ်။ program ဘယ်လိုအလုပ်လုပ်သွားလဲဆိုတာကို စာဖတ်သူကိုယ်တိုင် ပုံ (၃.၂၇) မှာရေးထားတဲ့ code တွေကို trace လုပ်ကြည့်ပါ။ while statement ကိုလည်း ပိုနားလည်သွားပါလိမ့်မယ်။

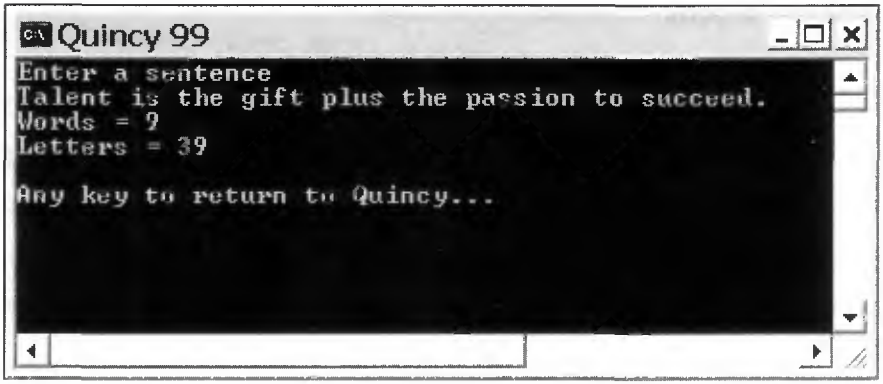
Ex3012.cpp

```
// Listing 3.12: This program counts characters
// and words that you typed in

#include <iostream>
#include <conio.h>  // For getche( ) function

int main( )
{
        int     chcount= 0, wdcount=1;
        char    ch='a';

        cout << "Enter a sentence\n";
        while (ch != '\r')
        {
                // While ch != ENTER
                ch = getche( );
                if (ch == ' ')  wdcount ++;
                else  chcount ++;
        }
        cout << "\nWords = " << wdcount << endl;
        cout << "Letters = " << (chcount - 1) << endl;
        return  0;
}
```

ပုံ (၃.၂၇)

Ex3012.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ <conio.h> ဆိုတဲ့ header ဖိုင်ဟာ getche( ) ဆိုတဲ့ library function ကို အသုံးပြုလို့ရအောင်ကူညီမှာပါ။ chcount ဟာ character အရေအတွက်ကို store လုပ်မယ့် variable တစ်ခုဖြစ်ပါတယ်။ chcount = 0 လို့ initialize လုပ်ထားပါတယ်။ wdcount က word အရေအတွက်ကို store လုပ်မယ့် variable ပါပဲ။ wdcount ကိုတော့ 1 လို့ initialize လုပ်ထားပါတယ်။ ch ဟာ ကျွန်တော်တို့ keyboard ကနေရိုက်ထည့်မယ့် character တစ်ခု ဖြစ်ပါတယ်။ ch ကို 'a' လို့ initialize လုပ်ထားပါတယ်။
- while loop နဲ့တွေ့တဲ့အခါမှာ ch က '\r' သို့မဟုတ် Enter key ကို ပုတ်တာမဟုတ်ဘူးဆိုရင်ဆိုတဲ့ <test expr> နဲ့စစ်ပါတယ်။ ch ဟာ '\r' မဟုတ်ပါဘူး ၊ 'a' ပါ။ ဒီတော့ while loop ထဲ ဝင်ပြီပေါ့။ ch = getche( ) ဆိုတဲ့အဓိပ္ပါယ်က keyboard ကနေ ကျွန်တော်တို့ရိုက်ထည့်တဲ့ character တစ်ခုကို ch နဲ့ညီပေးပါလို့ assign လုပ်တာပါ။ ကောင်းပြီ ၊ T စာလုံးကိုရိုက်ထည့်ကြည့်ပါ။ ch ဟာ blank space နဲ့တူတယ်ဆိုရင် wdcount ကို 1 တိုးပါလိမ့်မယ်။ အခုတော့ ch = 'T' ဖြစ်နေတာကြောင့် နောက်တစ်ကြောင်းကိုဆင်းသွားပါပြီ။
- character count ဖြစ်တဲ့ chcount အရေအတွက်ကို 1 တိုးပေးပါတယ်။ ဒီတော့ chcount = 0 + 1 = 1 ဖြစ်သွားပါပြီ။ closing brace (}) နဲ့တွေ့တဲ့အတွက် while loop အစကိုပြန်တက်သွားပါတယ်။ လောလောဆယ်မှာ ch = 'T' ဖြစ်တဲ့အတွက် while loop ထဲဆက်ဝင်လာတယ်လေ။
- ch = 'a' လို့ရိုက်ထည့်ပေးပါတယ်။ ဒီတစ်ခါလည်း ch က blank space မဟုတ်တဲ့အတွက် chcount ကို 1 ထပ်တိုးပေးဦးမှာပါ။ chcount = 1 + 1 = 2 ဖြစ်သွားပါပြီ။ ဒီနည်းအတိုင်း Talent အထိရိုက်ပြီးတဲ့အခါမှာ chcount = 6 ဖြစ်နေပါလိမ့်မယ်။ wdcount ကတော့ 1 ပါပဲ၊ မပြောင်းသေးပါဘူး။
- ဒီတစ်ခါ blank space နဲ့ assign လုပ်ပေးလိုက်မယ်ဆိုပါစို့။ wdcount ကို 1 တိုးပါလိမ့်မယ်။ chcount ကိုတော့မတိုးဘူးနော်။ ဒီလိုနဲ့ sentence တစ်ခုလုံးကိုရိုက်ပြီးသွားတဲ့အခါမှာ Enter ကိုနှိပ်လိုက်တာနဲ့ အဖြေတွေတစ်ခုပြီးတစ်ခု ကွန်ပျူတာစကရင်မှာပေါ်လာပါပြီ။ blank space (8) ခုကို တွေ့ခဲ့တဲ့အတွက် wdcount = 1 + 8 = 9 ဖြစ်သွားပါတယ်။ character တွေကျတော့ Enter နှိပ်တဲ့နောက်ဆုံးအကြိမ်မှာလည်း chcount ကို 1 ခုထပ်တိုးတယ်လေ။ ဒီတော့ အဖြေမှန်ကိုလိုချင်ရင် wdcount ထဲက 1 ခုပြန်နှုတ်ပေးရမှာပေါ့။ ပုံ (၃. ၂၈) မှာ program run ပြထားပါတယ်။



ပုံ (၃. ၂၈)

၃.၆ The do-while Statement

၁။ C++ program မှာ loop တစ်ခုကို while statement ကိုသုံးပြီးပတ်တဲ့အခါမှာ ရှေ့ဆက်ပြီးပတ်သင့်မပတ်သင့်ကို loop စစချင်းမှာပဲ စစ်ဆေးပါတယ်။ တစ်ခါတစ်ရံ ဒီလို test မျိုးကို loop ရဲ့နောက်ဆုံးကျမှ စစ်ဆေးတာမျိုးလည်းရှိပါတယ်။ အဲဒါဆိုရင် do-while statement ကိုအသုံးပြုရမှာပါ။ do-while ရဲ့ပုံစံက ဒီလိုပါ။

```
do {
        <statement1>;
        <statement2>;
        . . . . . .
}  while ( <test expression> );
```

စစချင်းမှာ loop ထဲကိုအတားအဆီးမရှိ do ကိုကျော်ပြီး loop body ထဲ ဝင်လို့ရပါတယ်။ loop body ထဲက statement တွေအားလုံးကို execute လုပ်ပြီးသွားရင် while နဲ့တွေ့မှာပါ။ ဒီခါကျရင် while နောက်က <test expr> ကိုစစ်လိုက်လို့ TRUE ဆိုရင် do-while loop ကို ဒုတိယအကြိမ်ပတ်ခွင့်ပြုမှာပါ။ FALSE ဆိုရင် ပတ်ခွင့်ပြုမှာမဟုတ်ပါဘူး။ while အောက်က next line တွေဆီကိုတန်းဆင်းရမှာပါ။ ဒါဟာ do-while control statement ရဲ့သဘောတရားပါပဲ။ ကောင်းပြီ ၊ do-while statement ကို လက်တွေ့အသုံးပြုကြည့်ရအောင်။

၂။ အခုတင်ပြမယ့် Ex3013.cpp program ရဲ့ရည်ရွယ်ချက်က ဂဏန်းတစ်လုံးပြောလိုက်တာနဲ့ ကွန်ပျူတာက အဲဒီဂဏန်းရဲ့ factorial ကိုရှာပေးမှာပါ။ ပုံ (၃. ၂၉) မှာရေးထားတဲ့ code statement တွေကို trace လုပ်ကြည့်ပါ။

Ex3013.cpp

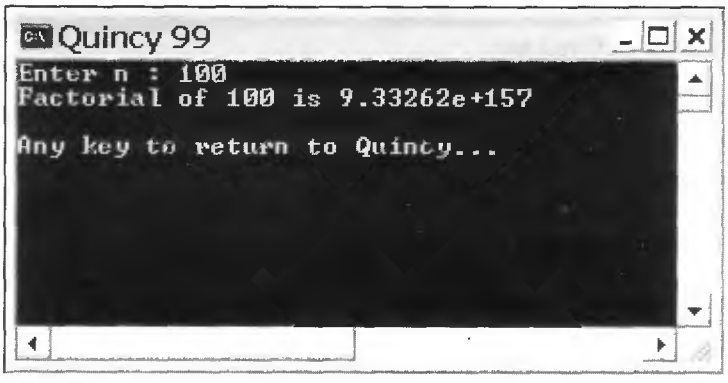
```
// Listing 3.13: This program computes the factorial of
// a positive integer n. The factorial of n represented
// by n! is defined as
//        n! = n(n-1)(n-2)(n-3)(n-4) .....(3)(2)(1)

#include <iostream>
int main( )
{
        long double   i= 2, n, nfac= 1;

        cout << "Enter n : ";
        cin >> n;
        do
                nfac *= i++;
        while (i <= n);
        cout << "Factorial of " << n << " is " << nfac << endl;

        return 0;
}
```

ပုံ (၃. ၂၉)

၂။ Ex3013.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် စစချင်းမှာ

- variable တွေကို long double လို့ declare လုပ်ပြီး i = 2 နဲ့ nfac = 1 လို့ initialize လုပ်ပါတယ်။ i က counter ပါ။ n! ရဲ့အဖြေကို နောက်ဆုံးမှာ nfac ထဲရောက်အောင်တွက်မှာပါ။ n ကတော့ factorial ရှာမယ့်ဂဏန်းပါပဲ။ long double ဆိုရင် 3.4 e-4932 ကနေ1.1e+4932 အထိ ကြိုက်ရာဂဏန်းတစ်ခုကို အသုံးပြုလို့ရပါတယ်။ Enter n : လို့ကွန်ပျူတာက prompt လုပ်တဲ့အခါကျရင် 5 ကိုရိုက်ထည့်မယ်ဆိုပါစို့။ ဒါဆိုရင် n = 5 ဖြစ်သွားပါပြီ။

- စစချင်း do-while loop ထဲဝင်လာပြီးတော့ nfac = nfac*i = 1*2 = 2 လို့ကွန်ပျူတာက တွက်ပါတယ်။ ပြီးတော့ရင် i ကိုတစ်တိုးပါတယ်။ do-while loop ကို i > 5 မဖြစ်မချင်းပတ်နေမှာပါ။ i = 6 ဖြစ်လာတဲ့အခါကျရင် do-while loop ကို ဆက်မပတ်တော့ပါဘူး။ next line ကို ဆင်းလာပြီးတော့ Factorial of 5 is 120 လို့အဖြေထုတ်ပေးပါလိမ့်မယ်။ n= 100 အတွက်လည်း run ကြည့်ပါဦး။ ပုံ (၃. ၃၀) ကိုကြည့်ပါ။ ဒါဟာ do-while statement ကိုအသုံးပြုနည်းပါပဲ။



ပုံ (၃. ၃၀)

Prime Factors of a Number

၁။ အခုတင်ပြမယ့် Ex3014.cpp program ရဲ့ရည်ရွယ်ချက်က ဂဏန်းတစ်လုံးပေးထားရင် အဲဒီဂဏန်းရဲ့ ဆခွဲကိန်းဂဏန်းတွေကို ရှာပေးတဲ့ program ဖြစ်ပါတယ်။ ပုံ (၃. ၃၁) ကိုကြည့်ပါ။

၂။ Ex3013.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်း Enter a number လို့ prompt ပေါ်လာတဲ့အချိန်မှာ 105 လို့ရိုက်ထည့်ပေးလိုက်ရင် n = 105 ဖြစ်သွားပါပြီ။ divisor = 2 ၊ adder = 1 လို့ initialize လုပ်ထားပါတယ်။ do-while loop ကိုစပတ်ပါပြီ။ if statement နဲ့ n % divisor ဟာသုညလားလို့မေးပါတယ်။ 105 ကို 2 နဲ့စားရင် အကြွင်းသုညမရပါဘူး။ ဒီတော့ else နောက်ကအလုပ်ကိုပဲ ကွန်ပျူတာက လုပ်မှာပါ။ divisor ကို 1 တိုးပေးသလို adder= 2 လို့ assign လုပ်ပါတယ်။ divisor= 3 ဖြစ်သွားပါပြီ။ ဒါပေမယ့် n = 105 က မပြောင်းသေးပါဘူး။

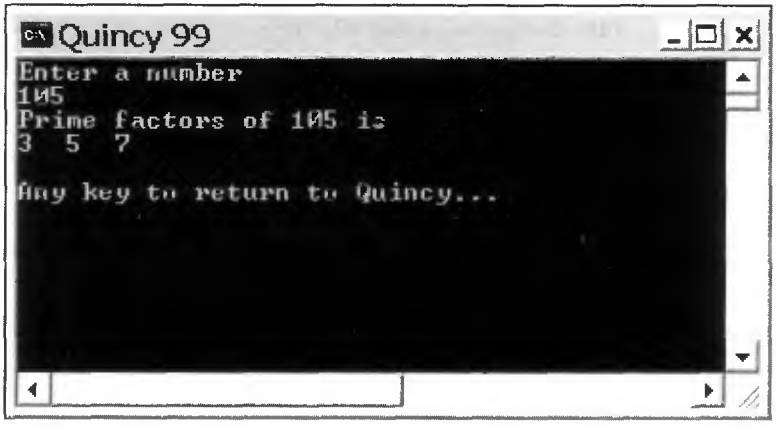
Ex3014.cpp

```
// Listing 3.14: This program prints a list of
// the prime factors of a number
#include <iostream>

int main()
{
        int  divisor=2,adder=1, n;

        cout << "Enter a number\n";
        cin >> n;
        cout << "Prime factors of " << n << " is\n";
        do {
                if (n % divisor == 0)
                {
                        n /= divisor;
                        cout << divisor << "  ";
                }
                else
                {
                        divisor += adder;
                        adder = 2;
                }
        } while (n/2 > 1);
        cout << '\n';

        return 0;
}
```

ပုံ (၃. ၃၁)

- while (n/2 > 1) ဆိုတဲ့အဓိပ္ပါယ်က n/2 ဟာ 1 ထက်ကြီးရင် do-while loop အစကိုပြန်သွားပါတဲ့။ n/2 = 105/2 = 52 ဆိုတော့ 1 ထက်ကြီးတာပေါ့။ do-while loop အစကို ပြန်သွားပါပြီ။ if (n % divisor == 0) ဆိုတာက 105 ကို divisor= 3 နဲ့စားလို့ပြတ်တယ်ဆိုရင် n = n/divisor = 105/3 = 35 လို့ပြင်ရပါမယ်။ အဲဒီ 3 ကိုလည်း ကွန်ပျူတာစကရင်မှာ display လုပ်ရပါမယ်။
- n/2 = 35/2 = 17 က 1 ထက်ကြီးတာမို့ do-while loop အစကိုပြန်သွားပါပြီ။ 35 ကို 3 နဲ့စားလို့မပြတ်ဘူးဆိုရင် else နောက်က divisor = divisor+adder = 3+2 = 5 လို့ပြင်ပြီး

while ဆီကိုရောက်လာပါတယ်။ n/2 = 35/2 = 17 ထက်ကြီးတာမို့ do-while loop အစ
ပြန်သွားပါတယ်။ 35 ကို 5 နဲ့စားလို့ပြတ်ရင် n = n/divisor = 35/5 = 7 လို့ပြင်ရပါမယ်၊
အဲဒီ 5 ကိုလည်း display လုပ်ရပါမယ်။ အဲဒီတိုင်းဆက်လုပ်သွားရင် 105 ရဲ့ဆခွဲကိန်းတွေဖြစ်
3 ၊ 5 ၊ 7 တို့ကို အဖြေထုတ်လို့ရပါပြီ။ ပုံ (၃. ၃၂) မှာ run ပြထားတဲ့ output ကိုကြည့်ပါ။



ပုံ (၃. ၃၂)

Linear Regression Method

၁။ အခုတင်ပြမယ့် Ex3015.cpp program ဟာဆိုရင် ပေးထားတဲ့ x-y data array (5) စုံကိုကိုယ်စားပြုတဲ့
error အနည်းဆုံး မျဉ်းဖြောင့်တစ်ကြောင်းရဲ့ equation ကိုရှာပေးမှာပါ။ ဒီဥစ္စာကို linear regression လုပ်တယ်
လို့ခေါ်ပါတယ်။ တစ်ကယ်လို့ data point တွေဟာ မျဉ်းဖြောင့်ပေါ်မှာ အားလုံးကျနေမယ်ဆိုလို့ရှိရင် equation
က ကွက်တိမှန်နေမှာပါ။ ဒီမျဉ်းဖြောင့်ရဲ့ equation ကို Y = a + bX လို့သတ်မှတ်မယ်ဆိုရင် a နဲ့ b တို့ကို

$$b = \frac{n\,(\Sigma xy) - (\Sigma x)\,(\Sigma y)}{n\,(\Sigma x^2) - (\Sigma x)^2}$$

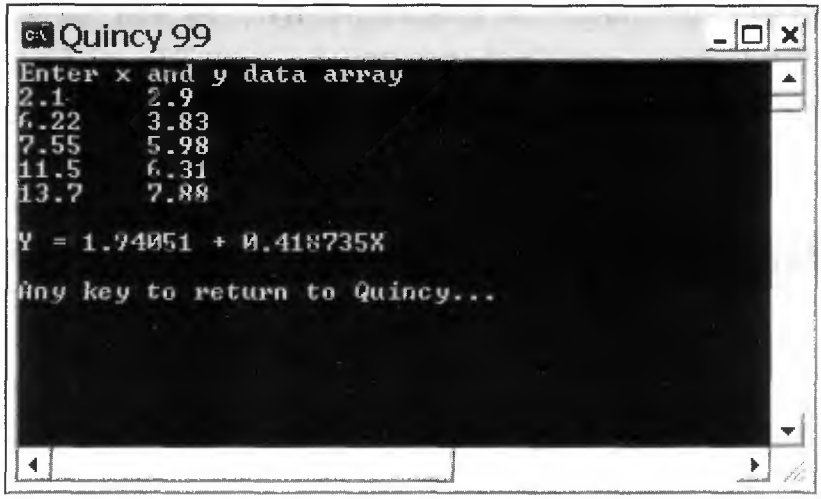
$$a = \frac{\Sigma y - b(\Sigma x)}{n}$$

ဆိုတဲ့ equation တွေနဲ့ တွက်ယူလို့ရပါတယ်။ ဒီညီမျှခြင်းတွေမှာ n ဟာ x-y data pair အရေအတွက်ပါ။ အခု program မှာ n = 5 လို့အသေယူထားပါတယ်။ program ကို general ဖြစ်စေချင်ရင် n ကိုအရှင်ထားပေးပါ။ program အလုပ်လုပ်သွားပုံကို ရှင်းမပြတော့ပါဘူး။ စာဖတ်သူကိုယ်တိုင် လေ့လာဖို့ချန်ထားခဲ့ပါမယ်၊ လေ့လာကြည့်ပါ။

Ex3015.cpp

```
// Listing 3.15: This program reads in pairs of data
// values x and y, calculates,and prints the values of
// the regression coefficients a and b.
#include <iostream>

int main()
{
        int       i=1, n=5;
        double    x, y, sumx, sumy, sumxx, sumxy, a, b;

        sumx = sumy = sumxx = sumxy = 0;
        cout << "Enter x and y data array\n";
        do {
                cin >> x >> y;
                sumx += x;
                sumy += y;
                sumxx += x*x;
                sumxy += x*y;
                ++i;
        } while (i <= n);

//  This is the regression formula
        b = (n*sumxy - sumx*sumy) / (n*sumxx - sumx*sumx);
        a = (sumy - b*sumx) / n;

//   Print result
        if (b<0) cout << "\nY = " << a << b << "X" << endl;
        else     cout << "\nY = " << a << " + " << b << "X" << endl;
        return 0;
}
```

ပုံ (၃.၃၃)



ပုံ (၃.၃၄)

၃.၇ The for Statement

၁။ for statement ဆိုတာ C++ မှာ looping လုပ်ဖို့အတွက် အသုံးများဆုံး control statement တစ်ခုပါပဲ။ ဒီ statement ရဲ့ပုံစံက အခုလိုပါ။

```
for ( [<initialization>] ; [<test expr>] ; [<increment>] )
{
        <statement1>;
        <statement2>;
        . . . . . .
}
```

ဒီပုံစံမှာ <initialization> expression ဟာ for loop ရဲ့စဦး counter ဖြစ်ပါတယ်။ initial index လို့ လည်းပြောလို့ရပါတယ်။ <test expr> က loop ကိုရှေ့ဆက်ပတ်သင့် ၊ မပတ်သင့်ဆိုတာကို စီစစ်တဲ့ဥစ္စာတစ်ခု ပါပဲ။ ဒီ conditional expression ဟာမှန်နေသမျှ looping ဟာဆက်ပတ်နေမှာဖြစ်ပါတယ်။ <increment> expression က counter ကို increment သို့မဟုတ် decrement လုပ်ပေးပါတယ်။ increment operator

နဲ့ decrement operator တွေကို for statement ထဲမှာထည့်သုံးလို့ရပါတယ်။ for(; ;) လို့ရေးမယ်ဆိုရင် never ending loop ဖြစ်သွားမှာပါ။ infinite loop လို့လည်းခေါ်ပါတယ်။ ပြဿနာမတက်အောင် သတိထားကြည့်သုံးပါ။ for(; <condition>; <increment>) လို့လည်း ရေးလို့ရတာရှိပါတယ်။

၂။ အခုတင်ပြမယ့် Ex3016.cpp program ဟာဆိုရင် for loop ကိုအသုံးပြုပြီး ပေးရင်းကိန်းတစ်ခုဖြစ်တဲ့ num = 2.5 ကို (5) ကြိမ်တိတိ ထပ်ကိန်းတင်ပေးတာဖြစ်ပါတယ်။ counter ကို power တင်တဲ့ index အဖြစ် အသုံးပြုထားပါတယ်။ သန်ခ်ာနဲ့ပတ်သက်တဲ့ pow( ) function ကိုအသုံးပြုလို့ရအောင် <cmath> ဆိုတဲ့ math header ကိုထည့်ထားပါတယ်။ ပုံ (၃. ၃၅) ကိုကြည့်ပါ။

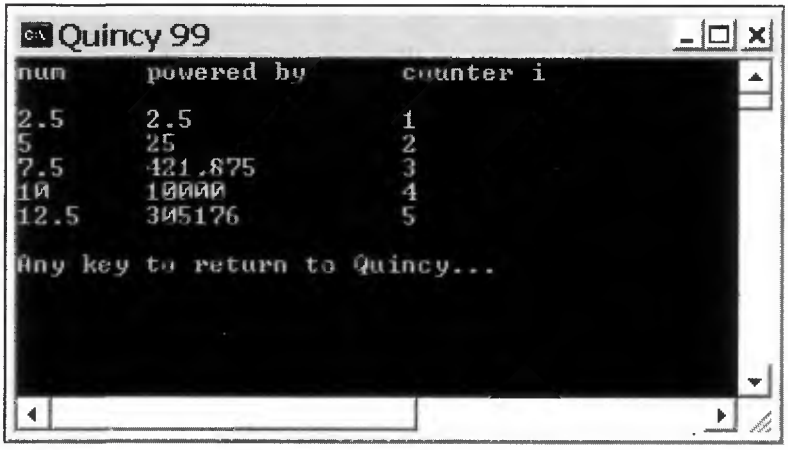
Ex3016.cpp

```
// Listing 3.16: This program illustrates the use of for statement.

# include <iostream>
# include <cmath>

int main()
{
        int       i;
        double    num, x=2.5;

        cout << "num\tpowered by\tcounter i\n\n";
        for ( i = 1; i <= 5; i++)
        {
                num = x*i;
                cout << num << '\t' << pow(num,i)<< "\t\t"
                << i << '\n';
        }
        return 0;
}
```

ပုံ (၃. ၃၅)

၂။ Ex3017.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် စစချင်းမှာ

- counter အဖြစ်အသုံးပြုမယ့် variable i ကို integer လို့ declare လုပ်ထားပါတယ်။ num နဲ့ x တို့ကို double လို့ကြေငြာပါတယ်။ num ၊ powered by နဲ့ counter i တို့ကို tab escape sequence '\t' အသုံးပြုပြီး ခေါင်းစည်းမှာခြားရိုက်ခိုင်းပါတယ်။
- for loop ကို (5) ကြိမ်တိတိပတ်ခိုင်းပါတယ်။ စစချင်းမှာ num = x*i = 2.5*1 = 2.5 ဖြ ကွန်ပျူတာကတွက်ယူလိုက်ပါတယ်။ နောက်ပြီး num=2.5 ၊ powered by= pow(num, i = pow(2.5, 1)= 2.5^1 = 2.5 နဲ့ i = 1 လို့တွက်ယူပြီး ကွန်ပျူတာမှာ display လုပ်ပြပါ လိမ့်မယ်။ ပြီးရင်နောက်တစ်ကြောင်းကိုဆင်းပြီး for loop ကိုပတ်ဦးမှာပါ။ ဒီ program မှာ C++ ရဲ့ output format အသုံးပြုနည်းကိုပြထားပါတယ်။ Ex3016.cpp program ကို run မယ်ဆိုရင် ပုံ (၃. ၃၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၃. ၃၆)

Creating a Temperature Conversion Table

၁။ အခုတင်ပြမယ့် Ex3017.cpp program ဟာဆိုရင် -40 ဒီဂရီစင်တီဂရိတ်ကနေ +40 ဒီဂရီစင်တီဂရိတ် အထိကို ဖာရင်ဟိုက်အဖြစ် တစ်တွဲတည်း conversion table အဖြစ် create လုပ်ပေးတဲ့ program ဖြစ်ပါတယ်။ for loop ဘယ်လိုအလုပ်လုပ်သွားလဲဆိုတာကို လေ့လာကြည့်ရအောင်။

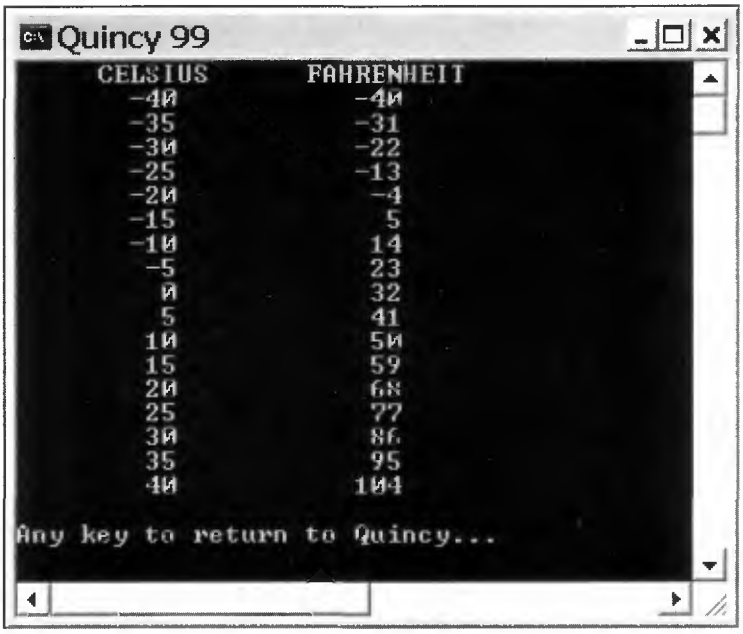
- program စစချင်း cel ကို signed int ၊ fah ကို float လို့ data type ကြေငြာပါတယ်။ heading ကိုရေးပါတယ်။ တစ်ကြောင်းခြားခိုင်းပါတယ်။

Ex3017.cpp

```
// Listing 3.17: This program converts the temperatures
// in Celsius to Fahrenheit.
#include <iostream>

int main()
{
        signed int   cel;
        float   fah;

        cout << "    CELSIUS     FAHRENHEIT\n";
        for (cel = -40; cel <= 40; cel += 10)
        {
                fah = 1.8*cel+32.0;
                cout.setf(ios::fixed);    // set cout flag for fixed-point
                cout.width(10);           // print width specified
                cout.precision(2);        // specify two decimal places
                cout << cel <<'\t';

                // width() manipulator does not stick from one
                // command to the next
                cout.width(10);
                cout << fah <<endl;
        }
        return 0;
}
```

ပုံ (၃.၃၇)

- for loop က <initialization> expression မှာ cel = -40 လို့စထားပါတယ်။ fah ဖော်မြူလာကိုအသုံးပြုပြီး fah =1.8*cel+ 32 = 1.8*(-40)+ 32 = -40 လို့တွက်လိုက်ပါပြီ။ for loop အတွင်းမှာ cel ကို print စလုပ်ပါတယ်။ tab ခြားပါတယ်။ fah ကို print လုပ်ပါမယ်။
- အားလုံး print လုပ်ပြီးတာနဲ့ closing brace (}) နဲ့တွေ့ရင် cel ကို 10 တိုးပါတယ်။ cel = -40+10 = -30 ဖြစ်သွားပါပြီ။ cel= -30 ဟာ last counter ဖြစ်တဲ့ cel = 40 နဲ့ယှဉ်ရင် ငယ်လားညီလားလို့မေးပါတယ်။ ဒီ <test expr> ဟာ TRUE ဖြစ်တာမို့ for loop ကိုဆက်ပတ်ပါတယ်။ တစ်ချိန်မှာ cel=50 ဖြစ်လာမှာပါ။ ဒီခါကျရင် cel=50 ဟာ 40 ထက်ဘယ်ငယ်တော့မလဲ ၊ ညီတော့မလဲ။ FALSE ဖြစ်သွားပြီမို့ for loop ထဲကိုမဝင်တော့ပါဘူး။ program output ကို ပုံ (၃.၃၈) မှာဖော်ပြထားပါတယ်။



ပုံ (၃.၃၈)

၂။ ဒီ program ကိုပဲ setw( ) manipulator အသုံးပြုပြီး အခုလို ပြင်ရေးပြီးတော့ run ရင်လည်းရပါတယ်။

```
#include <iostream>
#include <iomanip>

int  main( )
{
     signed int    cel;
     float   fah;

     cout << "     CELSIUS      FAHRENHEIT\n";
     for  (cel = -40; cel <= 40; cel += 5)
     {
          fah = 1.8*cel + 32.0;
          cout << setw(10) << cel << setw(14) << fah << endl;
     }
     return   0;
}
```

Generating a Table of Multiples of a Number

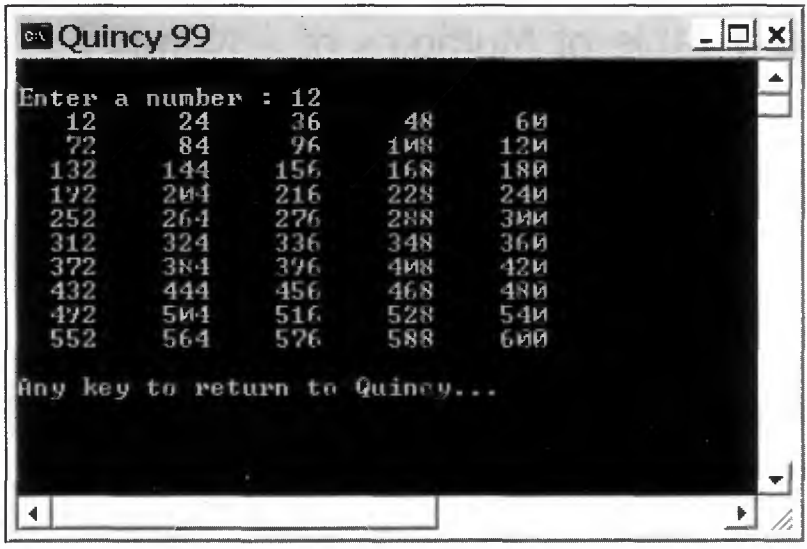
၁။ အခုတင်ပြမယ့် Ex3018.cpp program ဟာဆိုရင် user ကနေ input လုပ်ပေးတဲ့ ဂဏန်းတစ်ခုရဲ့ ဆပ်ွားကိန်းတွေကို (5) လုံးတစ်ကြောင်းပုံစံနဲ့ display လုပ်ပြခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ်။ for loop ဘယ်လိုအလုပ်လုပ်သွားလဲဆိုတာကို ပုံ (၃. ၃၉) က program ကိုလေ့လာကြည့်ပါ။

Ex3018.cpp

```
// Listing 3.18: This program generates a table of multiples
// of a number that you typed in. The program prints multiples
// every  5 numbers and starts next line.

#include <iostream>
#include <iomanip>

int main()
{
        unsigned long   num;

        cout << "\nEnter a number : ";
        cin >> num;
        for (int i=1; i<=50; i++)
        {
                cout << setw(5) << i*num << "  ";
                if (i%5 == 0)  cout << endl;
        }
        return 0;
}
```

ပုံ (၃. ၃၉)

၂။ Ex3018.cpp program ကို run မယ်ဆိုရင် ပုံ (၃. ၄၀) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ 12 အလီကို 50 အထိဖေါ်ပြထားတဲ့ အလီဇယားတစ်ခုပဲပေါ့။ (5) လုံးစာတွက်ပြီးရင် next line ကူးတာကို if (i % 5 == 0) cout << endl; statement ကနေလုပ်ပေးပါတယ်။



ပုံ (၃. ၄၀)

၃.၈ Nested Loops

၁။ loop တစ်ခုကိုတစ်ခြား loop တစ်ခု သို့မဟုတ် တစ်ခုထက်ပိုတဲ့ loop တွေကငုံထားတာကို nested loop လို့ခေါ်ပါတယ်။ inner looping နဲ့ outer looping တို့ရဲ့ control structure အမျိုးအစားတွေဟာ တူစရာမလိုပါဘူး။ inner loop မှာ for နဲ့သုံးမယ်။ outer loop မှာ while သုံးမယ်။ ဒါလည်းရပါတယ်။ ဒါမှမဟုတ် outer loop မှာ do-while သုံးပြီးတော့ inner loop မှာ for ကို သုံးမယ်ဆိုလည်းဖြစ်တာပါပဲ။ တစ်ခုပဲသတိထားရမှာက loop တစ်ခုနဲ့တစ်ခုဟာ crossing ဖြတ်လို့မရပါဘူး။ အပြင် loop က အတွင်း loop ကို လုံးဝငုံနေရမှာပါ။

၂။ အခုတစ်ခါတင်ပြမယ် Ex3019.cpp program ဟာ 2-equation (3-unknowns) ကို nested for loop အသုံးပြုပြီး ဖြေရှင်းထားတာပါ။ ပုံ (၃. ၄၁) မှာ source code တွေကိုဖော်ပြထားပါတယ်။

၃။ Ex3019.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စစချင်း program အသုံးပြုမယ့် variable တွေအားလုံးကို integer လို့ကြေငြာပါတယ်။ outer for loop စပါတယ်။ x = 1 ပါ။ inner for loop ထဲကိုဆက်ဝင်ပါတယ်။ ဒီထဲမှာ y = 1 လို့စပါတယ်။

Ex3019.cpp

```
// Listing 3.19: This program determines the three
// unknowns x,y, and z,which are defined by two equations:
//          x  +  y +    z = 25
//       2.5x +  5y + 0.25z = 25
// where x, y, and z are integers.

#include <iostream>

int main()
{
        int  x,y,z;

        for (x=1; x <= 10; ++x)
                for (y=1; y <= 5; ++y)
                {
                        z = 25 - x- y;
                        float sum = 2.5*x+ 5*y + 0.25*z;
                        if (sum == 25) goto RESULT;
                }
        RESULT : cout << "X = " << x << endl;
        cout << "Y = " << y << endl;
        cout << "Z = " << z << endl;
        return 0;
}
```

ပုံ (၃. ၄၁)

- ကွန်ပျူတာက အခုလိုစပြီးတွက်ယူပါတယ်။

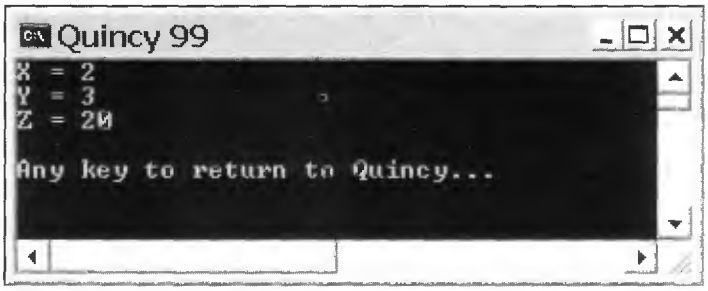
  z = 25-x-y = 25-1-1 = 23

  float sum = 2.5*x + 5*y + 0.25*z

  = 2.5*1 + 5*1 + 0.25*23 = 13.25

- sum = 25 လားလို့မေးပါတယ်။ မှန်တယ်ဆိုရင် RESULT label ရှိတဲ့နေရာကို ကွန်ပျူတာသွားမှာပါ။ sum = 13.25 ဟာ 25 နဲ့မတူဘူးဆိုတော့ အောက်ဆင်းလာပါတယ်။ closing brace နဲ့တွေ့ပြီး inner for loop ကို ဒုတိယအကြိမ်ထပ်ပတ်ပါပြီ။
- ဒီတစ်ခါ x= 1 နဲ့ y ကို 2 လို့ယူပြီးကွန်ပျူတာကပြန်တွက် ၊ ပြန်စစ်လုပ်ပါတယ်။ တစ်ချိန်မှာတော့ တွက်လို့ရတဲ့ sum တန်ဖိုးဟာ 25 ဖြစ်သွားပြီး nested loop ထဲကနေ ခုန်ထွက်လာပါပြီ။ RESULT label ရှိတဲ့နေရာမှာ X = 2 ၊ Y = 3 နဲ့ Z = 20 လို့ print လုပ်ပေးပါလိမ့်မယ်။

■ program output ကို ပုံ (၃.၄၂) မှာဖော်ပြထားပါတယ်။



ပုံ (၃.၄၂)

၄။ Ex3019.cpp program မှာအသုံးပြုတဲ့ nested for loop အစား while ၊ do-while statement တွေကိုအသုံးပြုပြီး အခုလိုပြင်ရေးလို့ရပါတယ်။ ပုံ (၃.၄၃) မှာပြထားတဲ့ Ex3020.cpp program ကိုလေ့လာကြည့်ပါ။

Ex3020.cpp

```
# include <iostream>

int main()
{
        int  x=1, y, z;

        while (1)
        {       y = 1;
                do {
                        z = 25 - x- y;
                        float sum = 2.5*x+ 5*y + 0.25*z;
                        if (sum == 25) goto RESULT;
                                ++y;
                } while (y <=5);
                ++x;
        }
        RESULT : cout << "X = " << x << endl;
                cout << "Y = " << y << endl;
                cout << "Z = " << z << endl;
        return 0;
}
```

ပုံ (၃.၄၃)

၃.၉ The switch Statement

၁။ statement တွေ အများကြီးပါဝင်တဲ့ variable group အုပ်စုကနေ ကိုယ်လိုချင်တဲ့ statement group ဆီကိုခွေးပြီး ရောက်သွားစေချင်ရင် switch ဆိုတဲ့ statement ကိုသုံးရပါမယ်။ switch ရဲ့ syntax က ဒီလိုရှိပါတယ်။

```
switch ( <switch variable> )
{
        case <constant expr1> :
                        <statement1>;
                        [break;]
        case <constant expr2> :
                        <statement2>;
                        [break;]
        . . . . . .
        . . . . . .
        . . . . . .
 default :
                <statement_default>;
}
```

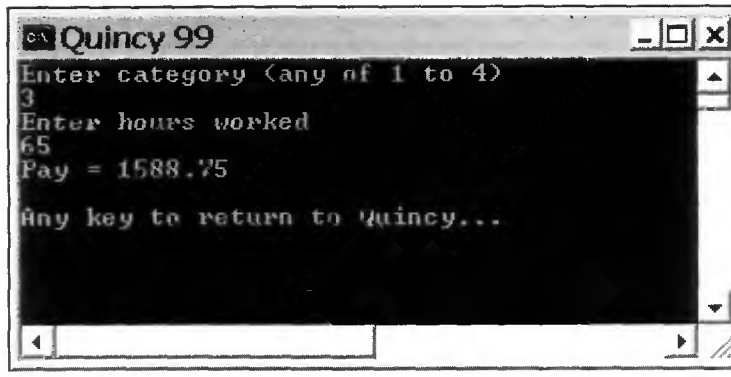
ဒီပုံစံမှာ switch နောက်မှာရှိတဲ့ <switch variable> ရဲ့ current value ဟာ <constant expr1> ရဲ့တန်ဖိုးနဲ့တူမယ်ဆိုလို့ရှိရင် <statement1> ကို execute လုပ်ပြီး break တွေ့တာနဲ့ closing brace (}) အောက်ကို ဆင်းသွားမှာပါ။ break က optional ပါ။ လိုမှထည့်ပေးရမှာပါ။ တစ်ကယ်လို့ <switch variable> ရဲ့ current value ဟာ <constant expr2> ရဲ့တန်ဖိုးနဲ့တူမယ်ဆိုရင် <statement2> ကို execute လုပ်ပြီး closing brace အောက်ကိုဆင်းသွားမှာပါပဲ။ <constant expr> တွေ တစ်ခုမှမတူဘူးဆိုရင် default နောက်က <statement_default> ကို execute လုပ်မှာပါ။ ပြီးရင် switch block က ထွက်သွားမှာဖြစ်ပါတယ်။ Ex3021.cpp program ကိုအရင်လေ့လာကြည့်ပါ။ ပုံ (၃.၄၄) မှာဖော်ပြထားပါတယ်။

Ex3021.cpp

```
// Listing 3.21: This program calaculates the payroll for
// four different classes of workers.
#include <iostream>

int main()
{
        int     category;
        float   wage, hours, pay;

        START : cout << "Enter category (any of 1 to 4)\n";
        cin >> category;
        switch (category)
        {
                case 1:
                        wage = 12.5;  break;
                case 2:
                        wage = 15.5;  break;
                case 3:
                        wage = 20.5;  break;
                case 4:
                        wage = 25.5; break;
                default:  {  cout << "Try again!\n";
                             goto START;          }
        }
        cout << "Enter hours worked\n";
        cin >> hours;
        if (hours <= 40)    pay = wage*hours;
        else                pay = wage*40 + 1.5*wage*(hours-40);
        cout << "Pay = " << pay << endl;
        return 0;
}
```

ပုံ (၃.၄၃)

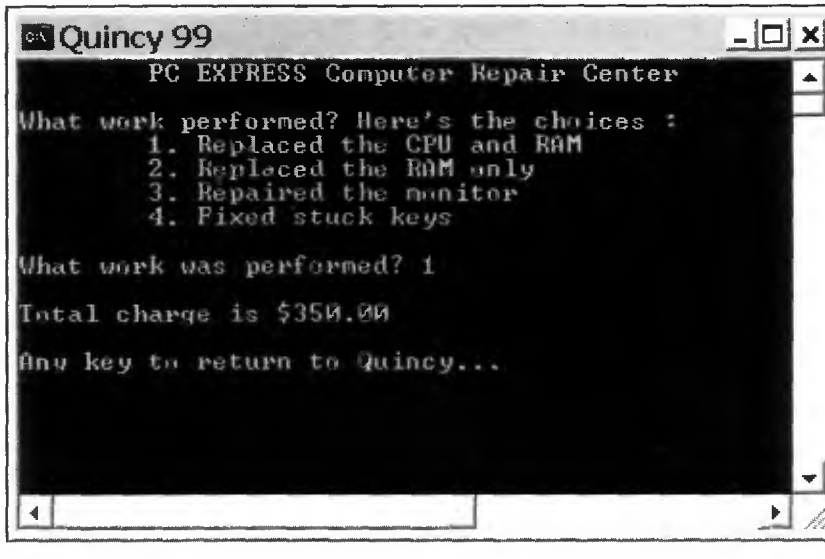
၂။ Ex3021.cpp ကို run လိုက်မယ်ဆိုရင် ပုံ (၃.၄၄) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ category အတွက် 3 နဲ့ working hours အတွက် 65 တို့ကိုရိုက်ထည့်ပေးမယ်ဆိုရင် Pay = 1588.75 လို့တွက်ပေးမှာပါ။ ဒါပေမယ့် category < 1 သို့မဟုတ် category > 4 ဖြစ်နေရင် data ပြန်တောင်းပါလိမ့်မယ်။



ပုံ (၃.၄၄)

Use a switch Statement to Control a Menu

၁။ အခုတင်ပြမယ့် Ex3022.cpp program ဟာ switch statement ကိုအသုံးပြုပြီး menu တစ်ခုကို create လုပ်နည်းဖြစ်ပါတယ်။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၃.၄၅) မှာပြထားတဲ့အတိုင်း menu list တစ်ခုကိုတွေ့ရမှာပါ။ choice (4) မျိုးထဲက နံပါတ် (1) ကို choose လုပ်ထားပါတယ်။



ပုံ (၃.၄၅)

Ex3022.cpp

```
// Listing 3.22: This program uses a switch statement to
// control a user's selection.
#include <iostream>

int main()
{
        int     choice;
        float   charge=0;

        cout << "\tPC EXPRESS Computer Repair Center\n\n";
        cout << "What work performed? Here's the choices :\n";
        cout << "\t1. Replaced the CPU and RAM\n";
        cout << "\t2. Replaced the RAM only\n";
        cout << "\t3. Repaired the monitor\n";
        cout << "\t4. Fixed stuck keys\n";
        do {
                cout << "\nWhat work was performed? ";
                cin >> choice;
        } while (choice < 1 || choice > 4);

        switch (choice)
        {
                case 1 :  charge = 200;     // Notice no break here
                case 2 :  charge += 150;   break;
                case 3 :  charge = 75;     break;
                case 4 :  charge = 12;
        }
        cout.precision(2);
        cout.setf(ios::showpoint);
        cout.setf(ios::fixed);
        cout << "\nTotal charge is $" << charge << endl;
        return 0;
}
```

ပုံ (၃.၄၆)

၂။ ပုံ (၃.၄၅) က program မှာ case 1: statement နောက်ဆုံးမှာ break ကို တမင်ဖြုတ်ထားတာကို သတိပြုကြည့်ပါ။ switch (choice) မှာ choice=1 ဖြစ်မယ်ဆိုရင် charge=200 လို့ကွန်ပျူတာက နားလည်ပါတယ်။ ဒါပေမယ့် case 1 statement မှာ break မပါတဲ့အတွက် switch loop က အပြင်ကိုခုန်မထွက်သေးပါဘူး။

အောက်ကိုဆက်ဆင်းလာပြီး charge += 150 သို့မဟုတ် charge = 200+150 = 350 လို့တွက်ပါလိမ့်မယ်။ break ကိုတွေ့တော့မှ switch loop အပြင်ကိုခုန်ထွက်မှာပါ။ choice=2 ကို data input လုပ်ရင်တော့ အဖြေ charge = 150 ပဲရမှာပါ။

၃.၁၀ The break Statement

break ဆိုတာ looping သို့မဟုတ် switch အုပ်စုကနေ exit လုပ်ဖို့အတွက်အသုံးပြုတာပါ။ while ၊ do-while ၊ for ၊ switch statement တွေနဲ့ break ကိုတွဲသုံးလို့ရပါတယ်။ ဒါပေမယ့် if-statement ကနေ exit လုပ်တာကိုတော့ break နဲ့သုံးလို့မရပါဘူး။ goto ပဲသုံးလို့ရမှာပါ။ Ex3023.cpp ဟာ Ex3020.cpp program

Ex3023.cpp

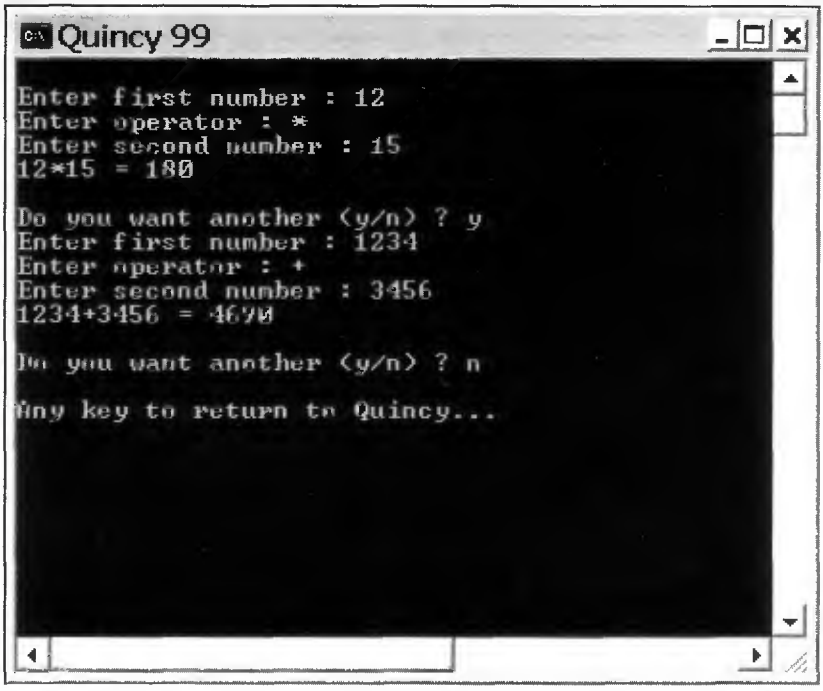
```
// Listing 3.23: This program determines the three
// unknowns two equations.
#include <iostream>

int main()
{
        int     x,y,z;
        float   sum;

        for (x=1; x <= 10; ++x)
        {
                for (y=1; y <= 5; ++y)
                {
                        z = 25 - x- y;
                        sum = 2.5*x+ 5*y + 0.25*z;
                        if (sum == 25)  break;
                }
                if (sum == 25)  break;
        }
        cout << "X = " << x << endl;
        cout << "Y = " << y << endl;
        cout << "Z = " << z << endl;
        return 0;
}
```

ပုံ (၃.၇၇)

ကို break statement အသုံးပြုပြီး နောက်တစ်မျိုးပြင်ရေးထားတာပါပဲ။ break statement ကိုအသုံးပြုတဲ့အခါမှာ သတိထားရမယ့်အချက်က break တစ်ခုဟာ loop တစ်ခုစာပဲခုန်ထွက်လို့ရပါတယ်။ nested loop တွေက exit လုပ်တဲ့အခါမှာ break တစ်ခုမက အသုံးပြုရပါမယ်။ goto statement ကတော့ ကြိုက်သလို ခုန်ကျော်လို့ရပါတယ်။

The Equivalent of a Four-function Calculator

အခုတစ်ခါတင်ပြမယ့် break statement ကို အပေါင်းအနှုတ် ၊ အမြောက်အစားလုပ်တဲ့ calculator program မှာ ပုံ (၃. ၄၈) မှာပြထားတဲ့အတိုင်းအသုံးပြုထားပါတယ်။ ဒီအတွက် program code တွေကို ပုံ (၃. ၄၉) က Ex3024.cpp program မှာရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



ပုံ (၃. ၄၈)

Ex3024.cpp

```
// Listing 3.24: This program creates the equivalent of
// a four-function calculator. The program would request
// the user to enter a number,an operator, and another number.

#include <iostream>
#include <conio.h>

int main( )
{
        double    num1, num2, ans;
        char      opr,ch;

        do {
                cout << "\nEnter first number : "; cin >>  num1;
                cout << "Enter operator : ";     cin >>  opr;
                cout << "Enter second number : "; cin >> num2;
                switch (opr) {
                        case '+' : ans = num1+num2;  break;
                        case '-' : ans = num1-num2;    break;
                        case '*' : ans = num1*num2;   break;
                        case '/' : ans = num1/num2;    break;
                        default : ans = 0;
                }
                cout << num1 << opr << num2 << " = " << ans <<endl;
                cout << "\nDo you want another (y/n) ? ";
                ch = getche( );
         } while (ch != 'n');

         cout << endl;
         return 0;
}
```

ပုံ (၃.၅၉)

၃.၁၁ The continue Statement

၁။ loop တစ်ခုကိုပတ်နေတုန်းမှာ loop ထဲက statement တွေကိုအကုန် execute မလုပ်ပဲနဲ့ လမ်းတစ်ဝက်

ကနေ နောက်ကြောင်းပြန်ပြီးတော့ loop ကို နောက်တစ်ကြိမ်ပြန်ပတ်ချင်ရင် continue ဆိုတဲ့ statement ကိုသုံး ရပါတယ်။ continue ကို while ၊ do-while ၊ for တို့နဲ့တွဲပြီးသုံးနိုင်ပါတယ်။

Ex3025.cpp

```
// Listing 3.25: This program finds the average of the
// nonnegative integers from a list of numbers.
#include <iostream>

int main()
{
        int     n, navg=0;
        float   x, avg, sum=0;

        cout << "How many numbers ? ";
        cin >> n;
        for (int count=1; count<= n; ++count)
        {
                cout << "X = ";
                cin >> x;
                if (x < 0)  continue;
                sum += x;
                ++navg;
        }
        avg = sum / navg;
        cout << "The average is " << avg << endl;
        return 0;
}
```

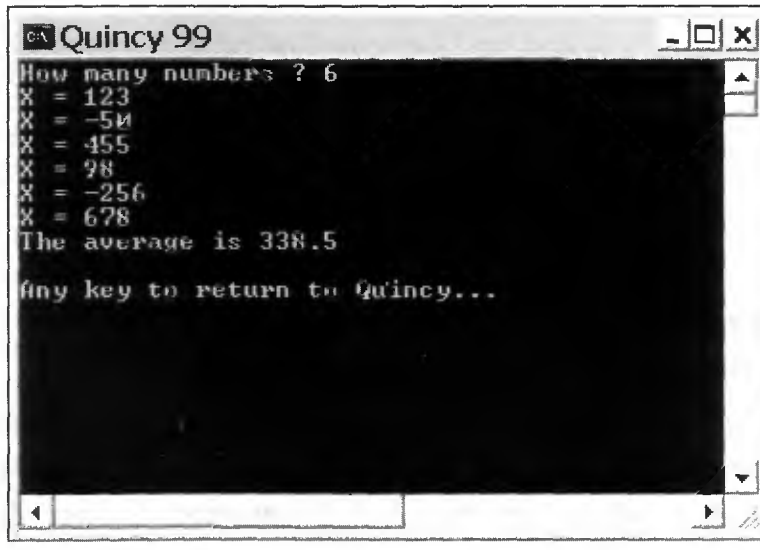
ပုံ (၃.၅၀)

၂။ Ex3025.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

■ ဒီ program မှာ ကျွန်တော်တို့ထည့်လိုက်တဲ့ data တွေထဲက +ive number တွေကိုပဲ ပေါင်းပြီးတော့ average ကိုရှာပေးမှာပါ။ စစချင်းမှာ variable တွေကို type declare လုပ် တယ်။ တစ်ချို့ကို initialize လည်းထပ်လုပ်ပါတယ်။ data တောင်းတဲ့ prompt ပေါ်ခိုင်း data ထည့်တယ်။

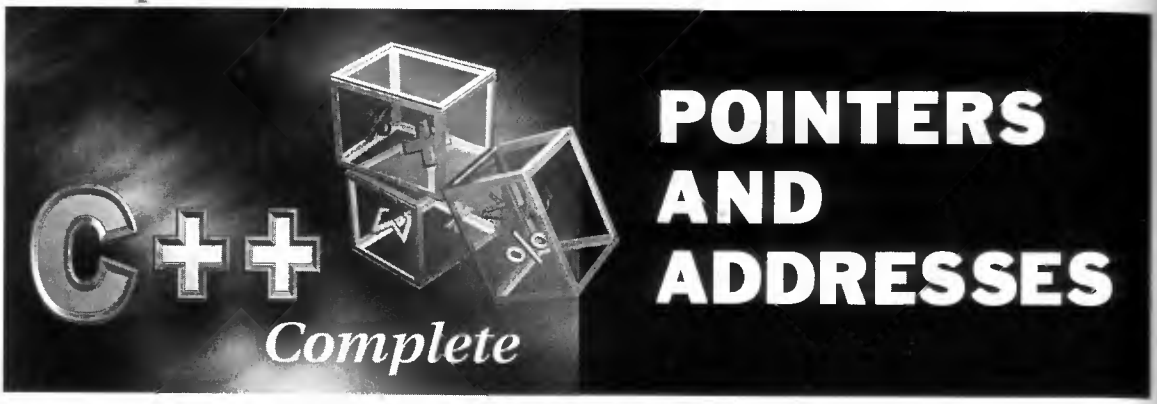
- for loop အတွင်းမှာအလုပ်လုပ်ပါတယ်။ x က သုညထက်ငယ်တယ်ဆိုရင် (negative number ဆိုရင်) သူ့အောက်က statement (2) ခုကိုခုန်ကျော်ပြီး for loop ကို နောက်တစ်ကြိမ်ပတ်ပါလိမ့်မယ်။ x ဟာ positive ဆိုရင် sum ထဲမှာထည့်ပေါင်းမှာပါ။ ပြီးရင် counter ကိုတစ်ခုတိုးပါလိမ့်မယ်။
- ဒီနည်းအတိုင်း program ကိုတွက်သွားမယ်ဆိုရင် sum ထဲမှာ positive number တွေအားလုံးရဲ့ပေါင်းလဒ်တွေကိုရပါပြီ။ နောက်ပြီး navg ထဲမှာလည်း positive number အရေအတွက်ကိုသိနေပြီလေ။ ဒီတော့ avg = sum ၊ navg ဟာအဖြေပဲပေါ့။ program ကိုစာဖတ်သူကိုယ်တိုင် trace လုပ်ကြည့်ပါ ၊ မခက်ပါဘူး။ break နဲ့ continue တို့အကြောင်းတွေကိုလည်း ပိုနားလည်သွားတာပေါ့။

၃။ Ex3025.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၃. ၅၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ ဂဏန်း (6) လုံးကိုထည့်ပေးလိုက်ရင် positive number တွေကိုပဲကွက်ပေါင်းပြီး average ရှာပေးပါလိမ့်မယ်။



ပုံ (၃. ၅၁)

Chapter 4



pointer ဆိုတာ variable တစ်ခု သို့မဟုတ် function တစ်ခုရဲ့ address တည်နေရာကိုသိမ်းဆည်းထားတဲ့ variable တစ်ခုပါပဲ။ ဥပမာ x ဆိုတဲ့ integer variable တစ်ခုရဲ့ value ဟာ 100 ဖြစ်ပြီး address က &x ဖြစ်မယ်။ နောက်ပြီး x ရဲ့ address value ကို store လုပ်ထားတဲ့ variable က xp ဖြစ်မယ် ဆိုရင်အဲဒီ xp ကို pointer လို့ခေါ်ပါတယ်။ x နဲ့ xp တို့ရဲ့ဆက်သွယ်မှုကိုဖော်ပြချင်ရင် int* xp = &x; လို့ရေးရမှာပါ။ & (ampersand) က address operator ပါ။ * ကို indirection operator လို့ခေါ်ပါတယ်။ int* xp ရဲ့အဓိပ္ပါယ်က variable xp ကို pointer လို့ define လုပ်ပြီး x ရဲ့ address value ကို xp နဲ့ assign လုပ်ပေးတဲ့သဘောပါပဲ။ အခုလိုဆက်သွယ်မှုဖြစ်နေရင် x ပဲ print လုပ်လုပ် ၊ *xp ကိုပဲ print လုပ်လုပ် အဖြေတစ်မျိုးတည်းရမှာပါ။ အဲဒါ 10 ပေါ့။ စင်စစ် pointer ရဲ့အလုပ်ဟာ data item တစ်ခုရဲ့တန်ဖိုးကို သာမန် variable name တစ်ခုနဲ့ ဖော်ပြနည်းမျိုးမဟုတ်ပဲ အဲဒီ data ရှိတဲ့ address နေရာကိုသွယ်ဝိုက်ဖော်ပြပြီး သိခြင်းပါ။ pointer တွေနဲ့ပတ်သက်တဲ့ idea ဟာ ရှုပ်ထွေးတယ်လို့မဆိုနိုင်ပါဘူး။ program တစ်ခုကိုကွန်ပျူတာ memory ထဲထည့်သွင်းပေးလိုက်မယ်ဆိုလို့ရှိရင် အဲဒီအတွက် address အုပ်စု တစ်ခုစာတော့နေရာယူမှာပါ။ နောက်ပြီး program

ဆီမှာပါဝင်တဲ့ variable တွေ၊ function တွေက သူတို့အတွက်သတ်မှတ်ထားတဲ့ address နေရာအသီးသီးကနေ ဆောပြုကြပါလိမ့်မယ်။

၄.၁ Referencing Variables with Pointers

ပုံ (၄. ၁) မှာဖော်ပြထားတဲ့ Ex401.cpp program ဟာဆိုရင် char ၊ int ၊ float type variable (3) ခုထဲက data value တွေကို pointer အသုံးပြုပြီး print လုပ်နည်းကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

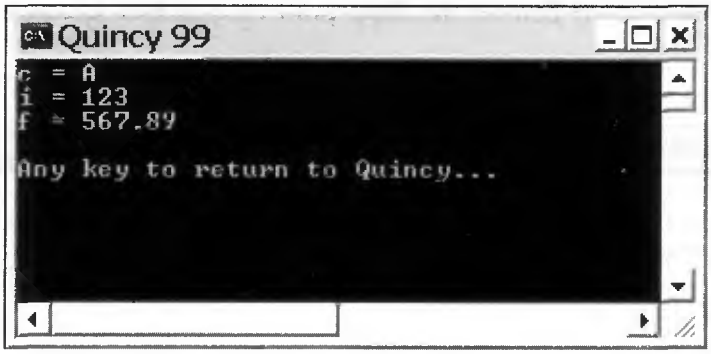
Ex401.cpp

```
// Listing 4.1: Referencing variables with pointers
#include <iostream>

int main()
{
        // initialize variables
        char   c = 'A';
        int    i = 123;
        float  f = 567.89;

        // define pointer types and initialize with address
        char*  cp = &c;
        int*   ip = &i;
        float* fp = &f;

        // print variable values using pointers
        cout << "c = " << *cp << endl;
        cout << "i = " << *ip << endl;
        cout << "f = " << *fp << endl;
        return 0;
}
```

ပုံ (၄. ၁)

၂။ Ex401.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ c ၊ i နဲ့ f ဆိုတဲ့ variable (3) ခုအတွက် data type တွေကိုကြေငြာပြီး သတ်မှတ်ထားတဲ့တန်ဖိုးအသီးသီးနဲ့ initialize လုပ်ပေးပါတယ်။ နောက်ပြီး pointer (3) ခုဖြစ်တဲ့ cp ၊ ip နဲ့ fp တို့ကို define လုပ်ပြီး c ၊ i နဲ့ f variable တွေရဲ့ address value တွေနဲ့ initialize လုပ်ပေးမှာပါ။ variable c ထဲက 'A' ကိုလိုချင်ရင် *cp ကို display လုပ်ခိုင်းရပါမယ်။
- ဒီနည်းအတိုင်း i နဲ့ f တို့ရဲ့ value တွေကိုလည်း display လုပ်လို့ရပါတယ်။ ပုံ (၄. ၂) ဟာ Ex401.cpp program ကို run လိုက်လို့ပေါ်လာတဲ့ output result ဖြစ်ပါတယ်။



ပုံ (၄. ၂)

Using the Pointer in a Loop

၁။ အခုတင်ပြမယ့် Ex402.cpp program ဟာဆိုရင် for loop ထဲမှာ pointer တစ်ခုကိုအသုံးပြုပြီး 10 ကနေ 1 အထိ ကွန်ပျူတာမှာပေါ်ခိုင်းပါလိမ့်မယ်။ ပြီးရင် BOOM! ဆိုတဲ့စာလည်း ဆက်ပေါ်လာမှာပါ။ ပုံ (၄. ၃) မှာရေးထားတဲ့ program က code statement တွေကိုလေ့လာကြည့်ပါ။

၂။ Ex402.cpp program ကို trace လုပ်ကြည့်မယ်ဆိုရင်

- program စစချင်း countDown ဆိုတဲ့ variable ကို type int လို့ကြေငြာပါတယ်။ ပြီးတော့ pointer cdp ကို define လုပ်ပြီး countDown ရဲ့ address value နဲ့ assign လုပ်

Ex402.cpp

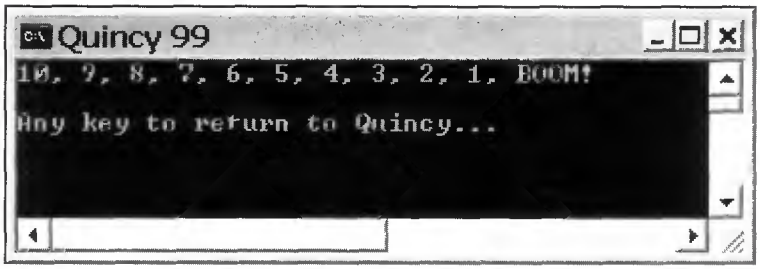
```
// Listing 4.2: Using a pointer, this program
// counts from 10 to 1 displaying the numbers
// followed by BOOM!.

#include <iostream>

int main()
{
        int     countDown;
        int*    cdp = &countDown;

        for (countDown=10; countDown >0; countDown--)
                cout << *cdp << ", ";
        cout << "BOOM!" << endl;
        return 0;
}
```

ပုံ (၄. ၃)

ပါတယ်။ for loop ထဲမှာ countDown ကို 10 ကနေ 1 အထိ decrement လုပ်သွားရင်း *cdp ကို display လုပ်ခိုင်းပါတယ်။

- ပထမဆုံး cdp point လုပ်နေတဲ့ countDown က 10 ပါ။ ဒီတော့ *cdp ကို display လုပ်ရင် ကွန်ပျူတာမှာ 10 ပဲပေါ်လာမှာပေါ့။ ပြီးရင် comma (,) တစ်ခုဆက်ပေါ်ပါလိမ့်မယ်။ for loop က အပေါ်ပြန်တက်သွားတဲ့အခါမှာ countDown က 9 ဖြစ်သွားပါပြီ။ ဒီတစ်ခါ cdp point လုပ်နေတဲ့ countDown က 9 ဖြစ်တဲ့အတွက် *cdp = 9 ဖြစ်ပါတယ်။ *cdp ကို display လုပ်ခိုင်းရင် 9 ပေါ်လာမှာပါ။
- ဒီနည်းအတိုင်း for loop ထဲမှာပတ်နေရင်း 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, တို့ကို display လုပ်ပြပါလိမ့်မယ်။ for loop ထဲကထွက်လာတဲ့အခါမှာ cout << "BOOM!" << endl; statement နဲ့တွေ့ရင် 1 နောက်မှာကပ်ပြီး BOOM! ကို display ဆက်လုပ်ပြမှာပါပဲ။

၃။ Ex402.cpp program ကို ပုံ (၄. ၄) မှာ run ပြထားပါတယ်။ output result တွေကို စောစောက ပြောတာနဲ့တိုက်ကြည့်လို့ရပါတယ်။



ပုံ (၄.၄)

၄.၂ Pointers and Arrays

၁။ ရိုးရိုး variable တွေအတွက် address သတ်မှတ်တဲ့အခါမှာ & (ampersand) ထည့်ပေးဖို့လိုပေမယ့် array တွေကျတော့ array name ကိုယ်တိုင်က address ဖြစ်နေတော့ & ထည့်ဖို့မလိုပါဘူး။ array notation၊ pointer notation နဲ့ pointer အသုံးပြုပုံတွေကို ပုံ (၄.၅) က Ex403.cpp program မှာနှိုင်းယှဉ်ဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

၂။ Ex403.cpp program ကို trace လုပ်ကြည့်မယ်ဆိုရင်

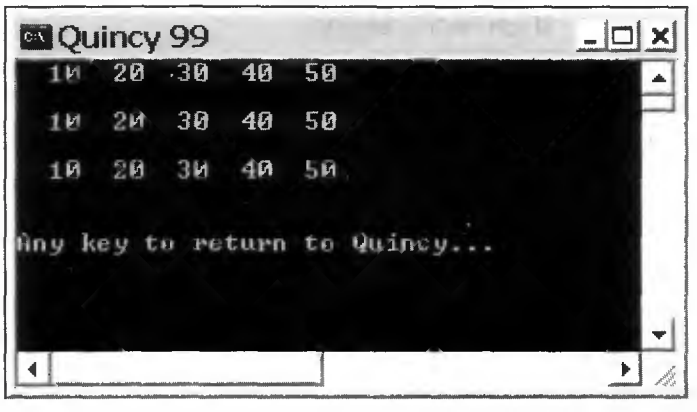
- စစချင်းမှာ type int ဖြစ်တဲ့ array တစ်ခုကို define လုပ်သလို initialize လည်းလုပ်ပါတယ်။ array name က x ပါ။ x ထဲမှာပါဝင်တဲ့ array element တစ်ခုချင်းကို array notation အသုံးပြုပြီး အရင်ဆုံး display လုပ်ခိုင်းပါတယ်။ x[0] = 10 ၊ x[1] = 20 ၊ x[2] = 30 ၊ x[3] = 40 နဲ့ x[4] = 50 တို့ဖြစ်ကြပါတယ်။
- ဒုတိယအုပ်စုမှာလုပ်တာက array element တွေကို pointer notation အသုံးပြုပြီးတော့ display လုပ်ခိုင်းတာပါပဲ။ *(x+i) ဟာ x[i] နဲ့အတူတူပါပဲ။ x+2 ဟာဆိုရင် x array ရဲ့တတိယမြောက် integer ရဲ့ address ကိုဆိုလိုပါတယ်။ ဒီတော့ *(x+2) ဟာ 30 ဆိုတဲ့တန်ဖိုးနဲ့တူမှာပါ။
- address တွေကိုညွှန်ပြတဲ့အခါမှာ x မှာ i ထည့်ပေါင်းသလိုပဲ pointer မှာ increment လုပ်ပေးပြီး ညွှန်လို့ရပါတယ်။ xp ဟာ x ကို point လုပ်နေတဲ့ pointer ဖြစ်တယ်ဆိုရင် *(xp++) ကို loop ထဲမှာ display လုပ်ခိုင်းလိုက်ရင် array element တစ်ခုချင်းပေါ်လာမှာပါပဲ။

၃။ Ex403.cpp ကို ပုံ (၄.၆) မှာ run ပြထားပါတယ်။ program နဲ့တိုက်ပြီး လေ့လာကြည့်လို့ရပါတယ်။

Ex403.cpp

```
// Listing 4.3: This program shows the different
// ways how the arrays are accessed.

#include <iostream>
int  x[5] = {10,20,30,40,50};

int main( )
{
        // array accessed with array notation
        for (int  i=0; i<5; i++)
                cout << "  " << x[i];
        cout << "\n\n";

        // array accessed with pointer notation
        for (int i=0; i<5; i++)
                cout << "  " << *(x+i);
        cout << "\n\n";

        // array accessed with pointer
        int*   xp = x;
        for (int i=0; i<5; i++)
                cout << "  " << *(xp++);
        cout << "\n\n";
        return  0;
}
```

ပုံ (၄.၅)



ပုံ (၄.၆)

Pointer Arithmetic

၁။ အခုတင်ပြမယ့် Ex404.cpp program ဟာဆိုရင် pointer notation အသုံးပြုပြီး array တစ်ခုမှာ store လုပ်ထားတဲ့ value တွေကို average ရှာပေးတဲ့ program ပါ။ ပုံ (၄. ၇) မှာရေးထားတဲ့ program က code statement တွေကိုလေ့လာကြည့်ပါ။

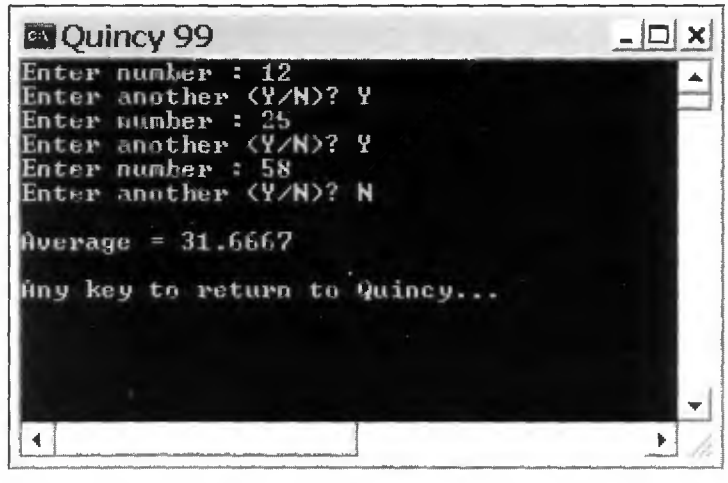
Ex404.cpp

```
// Listing 4.4: This program reads the numbers that
// you type in, and store them in an array. The program
// would average them and display the result.

#include <iostream>

const int   MAX =100;
char        ch;
int         k=0;
float       fx[MAX];

int main( )
{
        for (int i=0; i<MAX; i++)
        {
                cout << "Enter number : ";
                cin >> *(fx+ k++);
                cout << "Enter another (Y/N)? ";
                cin >> ch;
                if (ch != 'Y') break;
        }

        float sum = 0;
        for (int i=0; i<k; i++)
                sum += *(fx+i);
        float avg = sum/k;
        cout << endl << "Average = " << avg << endl;
        return 0;
}
```

ပုံ (၄. ၇)

Ex404.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- for loop နဲ့ pointer notation တို့ကိုအသုံးပြုပြီး x array ရဲ့ element တစ်ခုချင်းအတွက် value တွေကို ရိုက်ထည့်ပေးသွားပါတယ်။ cin >> *(x + k++); statement ကိုသတိပြုကြည့်ပါ။ k က ကျွန်တော်တို့ input လုပ်မယ့် array element အရေအတွက်ကိုမှတ်ပေးမယ့် variable တစ်ခုဖြစ်ပါတယ်။ data ကိုဆက်ထည့်ပေးချင်တယ်ဆိုရင် 'Y' ကိုရိုက်ထည့်ပါ။ data ထပ်မထည့်ချင်ဘူးဆိုရင် 'N' ကိုရိုက်ထည့်ပါ။ ဒါဆိုရင် break;နဲ့တွေ့ပြီး for loop အပြင်ဘက်ကိုခုန်ထွက်လာပါလိမ့်မယ်။
- အခုဆိုရင် count ထဲမှာ တစ်ကယ်အသုံးပြုတဲ့ array element အရေအတွက်ပါလာပါပြီ။ array element စုစုပေါင်းရဲ့တန်ဖိုးကို ပေါင်းခိုင်းပြီးတော့ sum ထဲမှာ store လုပ်ပေးပါတယ်။ နောက်ပြီး avg = sum/k နဲ့တွက်ယူပြီး အဖြေကို display လုပ်ပြလိုက်ပါပြီ။ ပုံ (၄. ၈) မှာ Ex405.cpp program ကို run လိုက်လို့ရလာတဲ့ result ကိုဖော်ပြထားပါတယ်။



ပုံ (၄. ၈)

- တစ်ကယ်လို့ pointer ကို တိုက်ရိုက်အသုံးပြုပြီး Ex404.cpp program ကိုပြင်ရေးမယ်ဆိုရင် ပုံ (၄. ၉) မှာပြထားတဲ့ program မျိုးကိုတွေ့ရမှာပါ။ ဒီ program ကို run ရင်လည်း စောစောက အဖြေပဲရမှာပါ။

Ex404A.cpp

```
#include <iostream>

const int MAX =100;
char  ch;
int   k=0;
float fx[MAX];

int main( )
{
        float* fxp = fx;

        for (int i=0; i<MAX; i++)
        {
                cout << "Enter number : ";
                cin >> *(fx+ k++);
                cout << "Enter another (Y/N)? ";
                cin >> ch;
                if (ch != 'Y') break;
        }

        float sum = 0;
        for (int i=0; i<k; i++)
                sum += *(fxp++);
        float avg = sum/k;
        cout << endl << "Average = " << avg << endl;
        return 0;
}
```

ပုံ (၄.၉)

၄.၃ Pointers to Structures

၁။ structure တွေနဲ့တွဲပြီး အလုပ်လုပ်မယ့် pointer တွေဟာ တစ်ခြား pointer တွေအလုပ်လုပ်သလိုပါပဲ။ structure pointer တစ်ခုဟာ structure type ရဲ့ instance ကို point လုပ်နေပါတယ်။ ဒီတော့ pointer ကို increment သို့မဟုတ် decrement လုပ်တာနဲ့ structure size ဟာ ဆပွားပြီးတိုးသွားနိုင်ပါတယ်။ လျော့နိုင်ပါတယ်။ structure တစ်ခုက member တစ်ခုကို pointer နဲ့ access လုပ်ချင်ရင် member pointer

operator (->) ကိုအသုံးပြုရမှာပါ။ ပုံ (၄. ၁၀) မှာဖော်ပြထားတဲ့ Ex405.cpp program ဟာဆိုရင် pointers to structures ဆိုတဲ့အဓိပ္ပါယ်ကို ပေါ်လွင်အောင်ရေးပြထားတဲ့ program တစ်ခုပါပဲ။

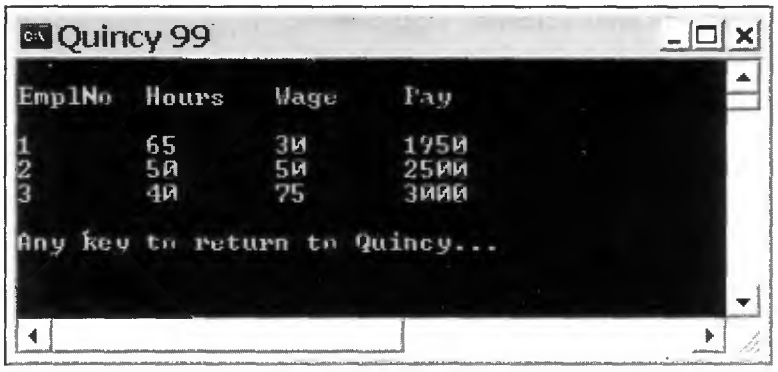
Ex405.cpp

```
// Listing 4.5: Pointers to structures
#include <iostream>

struct employeeRec
{
        int     emplNo;
        float   hours;
        float   wage;
};

void  displayRec(const employeeRec *y)
{
        cout << "\nEmplNo Hours  Wage   Pay\n\n";
        while (y->emplNo != -1)
        {
                cout << y->emplNo << "\t"
                     << y->hours  << "\t"
                     << y->wage   << "\t"
                     << (y->hours)*(y->wage) << endl;
                y++;
        }
}

int main()
{
        employeeRec  x[] = {
                                {1, 65, 30},
                                {2, 50, 50},
                                 {3, 40, 75},
                                {-1, 0, 0 }
                           };
        displayRec(x);
        return 0;
}
```

ပုံ (၄. ၁၀)

၂။ Ex405.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ employeeRec ဆိုတဲ့ structure တစ်ခုကို declare လုပ်ထားပါတယ်။ main( ) function ထဲမှာ employee instance တစ်ခုဖြစ်တဲ့ x[ ] array ကို declare လုပ်ခြင်းအားဖြင့် သူ့ထဲမှာ employeeRec structure variable တွေကို initialize လုပ်ထားလို့ရပါတယ်။ အဲဒီ displayRec( ) function ကို call ခေါ်ပြီး x[0] element ရဲ့ address ကို pass လုပ်ပေး လိုက်ပါပြီ။ displayRec( ) function ထဲမှာ ပထမဆုံး array element အတွက် တွက်တာ ချက်တာတွေလုပ်ပါလိမ့်မယ်။ တွက်ချက်ပြီးရင် pointer y ကို increment လုပ်ခြင်းအားဖြင့် x[1] element ၊ x[2] အတွက်ဆက်တွက်မှာပါ။ y->emplNo = -1 ဖြစ်သွားတဲ့အခါမှာ displayRec( ) ကနေပြန်ထွက်လာပြီး program ပြီးသွားပါပြီ။ ပုံ (၄. ၁၁) မှာ Ex405.cpp program ကို run ပြထားပါတယ်။



ပုံ (၄. ၁၁)

၄.၄ Pointers as Function Arguments

၁။ program တစ်ခုမှာ argument တွေကို function တစ်ခုကနေတစ်ခုဆီကို pass လုပ်နည်း (၃) နည်းရှိပါတယ်။ (၁) passing argument by value (၂) passing arguments by reference နဲ့ (၃) passing argument by pointer တို့ဖြစ်ကြပါတယ်။ အဲဒီထဲက ဒုတိယနည်းနဲ့ တတိယနည်းကိုတင်ပြပါမယ်။ (၄. ၁၀) က Ex406.cpp မှာဖော်ပြထားတဲ့ passing by reference နည်းကို အရင်လေ့လာကြည့်ရအောင်။

Ex406.cpp

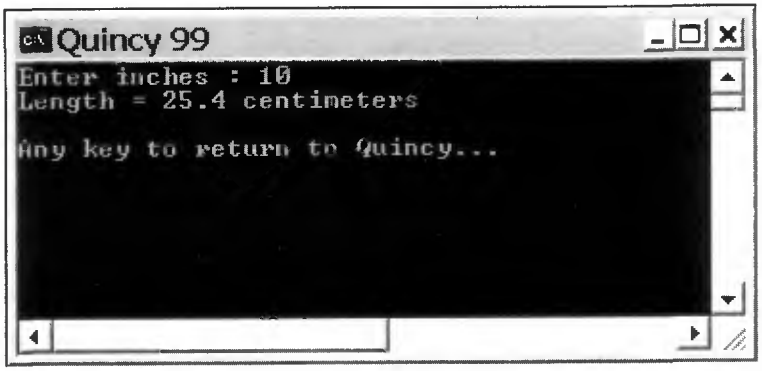
```
// Listing 4.6: This program shows how arguments
// are passed by reference
#include <iostream>

void  convert(double&);

int main( )
{
        double  inches;

        cout << "Enter inches : ";
        cin >> inches;
        convert(inches);
        cout << "Length = " << inches << " centimeters\n";
        return 0;
}

void  convert(double  &x)
{
        // convert inches to centimeters
        x *= 2.54;
}
```

ပုံ (၄.၁၂)

၂။ Ex406.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- main( ) function ထဲက inches ဆိုတဲ့ variable တစ်ခုကို လက်မကနေစင်တီမီတာကိုပြောင်းပေးတာပါ။ inches ကို centimeters ပြောင်းဖို့အတွက် convert( ) function ထဲကို by reference နည်းနဲ့ data pass လုပ်ပေးထားပါတယ်။ by reference နည်းအရ main( ) ထဲက convert( ) function မှာရိုးရိုး variable တစ်ခုကို argument အနေနဲ့ pass လုပ်ထားပေမယ့် prototype မှာ & ကို data type နောက်မှာ ထည့်ရေးထားတာကို သတိပြုကြည့်ပါ။ ဒါမှ data pass ပြန်လုပ်ပေးမှာပါ။ & ကိုဖြုတ်ထားမယ်ဆိုရင် program run ပေမယ့် အဖြေမှန်မှာမဟုတ်ပါဘူး။ တွက်ထားတဲ့ x တန်ဖိုးကို return မပြန်လို့ပါပဲ။

■ Ex406.cpp program ကို run လိုက်ပြီး inches = 10 ကိုရိုက်ထည့်ပေးရင် Length = 25.4 centimeters ဆိုတဲ့အဖြေကို ပုံ (၄. ၁၃) မှာပြထားတဲ့အတိုင်း မြင်ရပါလိမ့်မယ်။



ပုံ (၄. ၁၃)

■ Ex406.cpp program ကို passing by pointer နည်းနဲ့ပြင်ရေးမယ်ဆိုရင် ပုံ (၄. ၁၄) မှာပြထားတဲ့ program မျိုးကိုတွေ့ရမှာပါ။ ဒီ program ကို run ရင်လည်း စောစောကအဖြေပဲရမှာပါ။

Ex406A.cpp

```
#include <iostream>
void  convert(double* xp)
{
        // convert inches to centimeters
        *xp *= 2.54;
}

int main( )
{
        double  inches;
        cout << "Enter inches : ";
        cin >> inches;
        convert(&inches);
        cout << "Length = " << inches << " centimeters\n";
        return 0;
}
```

ပုံ (၄. ၁၄)

၄.၅ Passing Arrays by Pointers

function တွေထဲကို array တွေ pass လုပ်ခိုင်းတဲ့အခါမှာ အများအားဖြင့် array notation နဲ့ pass လုပ်တာထက် pointer နည်းကို အသုံးပြုတာများပါတယ်။ ပုံ (၄.၁၅) က Ex407.cpp program မှာ array passed by pointer နည်းကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

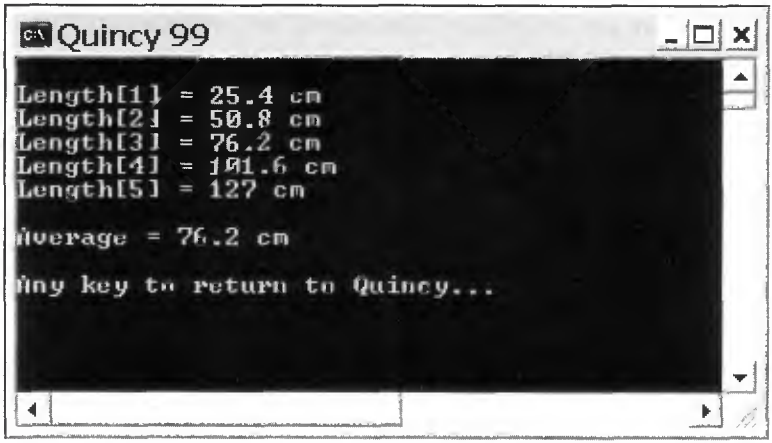
Ex407.cpp

```
// Listing 4.7: This program shows how arrays are
// passed by pointer.

#include <iostream>
const int    MAX =5;
void    convert(double*);

int main( )
{
        double  avg, sum= 0;
        double  inches[MAX] = {10,20,30,40,50};

        convert(inches);
        for (int i=0; i<MAX; i++)
        {
                cout << endl << "Length[" << i+1 << "] = "
                     << inches[i] << " cm";
                sum += inches[i];
        }
        avg = sum/MAX;
        cout << "\n\n" << "Average = " << avg << " cm"
             << end;
        return 0;
}

void convert(double* xp)
{
        for (int i=0; i<MAX; i++)
                *xp++ *= 2.54;
}
```

ပုံ (၄.၁၅)

၂။ Ex407.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်း main( ) function ထဲမှာ type double variable (2) ဖြစ်တဲ့ avg နဲ့ sum တို့ကို define လုပ်ပြီး sum = 0 လို့ initialize လုပ်ပါတယ်။ နောက်ပြီး inches နာမည်နဲ့ array တစ်ခုကို define ၊ initialize လုပ်ပါတယ်။ array element အရေအတွက်ဖြစ်တဲ့ MAX ကို 5 လို့ အစကတည်းက သတ်မှတ်ပေးထားပြီးသားပါ။ ဒီတော့ inches[0] = 10 ၊ linches[1] = 20 ၊ inches[2] = 30 အစရှိသည်တို့ ဖြစ်မှာပါ။
- convert( ) function ကို call ခေါ်ပါတယ်။ passing argument က inches array ရဲ့ address ဖြစ်ပါတယ်။ array name ဟာ address ပဲ ဖြစ်တာမို့လို့ convert (inches) လို့ pass လုပ်ရင်ရပါတယ်။ inches array ကို pointer to double xp ကနေလက်ခံပါတယ်။
- for loop ကိုပတ်ခိုင်းပြီး xp က point လုပ်နေတဲ့ inches address နေရာက element တစ်ခုချင်းကိုဆွဲယူပြီး လက်မကနေစင်တီမီတာကို ပြောင်းပေးနေပါပြီ။ array element တစ်ခုကနေ နောက် element နေရာကို ပြောင်းပေးချင်ရင် pointer xp ကို indirection opertor အသုံးပြုပြီး *xp++ လို့ increment တိုးပေးရင်ရပါတယ်။ တစ်ချိန်တည်းမှာပဲ main( ) ထဲက inches element တွေအားလုံးဟာ စင်တီမီတာတွေဖြစ်နေပြီလေ။
- သေချာအောင် main( ) function ထဲက inches array element တန်ဖိုးတွေကို display လုပ်ကြည့်ပါ။ sum+= inches[i]; ကိုရေးထားတဲ့ အဓိပ္ပါယ်က array element အားလုံးရဲ့ တန်ဖိုးတွေကိုပေါင်းခိုင်းပြီး sum နဲ့ assign လုပ်ပေးတာပါ။ နောက်ပြီး array element တွေရဲ့ ပျမ်းမျှတန်ဖိုးကို တွက်ခိုင်းပြီး display လုပ်ခိုင်းပါမယ်။ ပုံ (၄. ၁၆) ဟာဆိုရင် program run ပြထားတာပါ။ output result ကိုလေ့လာကြည့်ပါ။



ပုံ (၄. ၁၆)

Swapping Two Numbers with Pointers

ဒီ ကျွန်တော်တို့တွေ pointer အသုံးပြုပြီး array တွေ access လုပ်နည်းကို သိသွားပြီဆိုလို့ရှိရင် တစ်ဆင့် တက်ပြီးတော့ array element တွေစီတဲ့နည်းကို လက်တွေ့စမ်းကြည့်လို့ရပါပြီ။ အလွယ်ဆုံးအနေနဲ့ဂဏန်း (2) ခု ကို function တစ်ခုမှာ pass လုပ်ခိုင်းပြီး pointer အကူအညီနဲ့ သူတို့ကိုနေရာချင်းပြောင်းပေးမယ်လေ။ ပုံ (၄. ၁၇) မှာရေးထားတဲ့ program ကိုလေ့လာကြည့်ပါ။

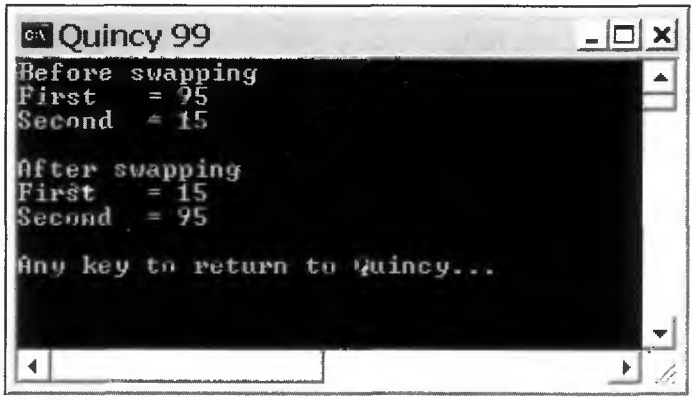
Ex408.cpp

```
// Listing 4.8: This program swaps two numbers
// passed to function as argument.

#include <iostream>
void  swap(int*, int*);

int main()
{
        int    first=95, second=15;
        cout << "Before swapping";
        cout << "\nFirst  = " << first
             << "\nSecond  = " << second;

        swap(&first, &second);
        cout << "\n\nAfter swapping";
        cout << "\nFirst  = " << first
             << "\nSecond  = " << second << endl;
        return 0;
}

void  swap(int *x, int *y)
{
        int  temp = *x;
        *x = *y;
        *y = temp;
}
```

ပုံ (၄. ၁၇)

၂။ Ex408.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ first = 95 ၊ second = 15 လို့ integer variable (2) ခုကို initialize လုပ်ပါတယ်။ swap မလုပ်ခင်မှာဒီဂဏန်း (2) ခုကို display လုပ်ပြီးတော့ swap( ) function ကို call ခေါ်ပါတယ်။ passing argument တွေက first နဲ့ second တို့ရဲ့ address တွေပါ။
- ဒီ address တွေဟာ pointers to integers first နဲ့ second ဖြစ်တဲ့ x နဲ့ y ထဲကို pass လုပ်ဝင်သွားပါပြီ။ swap( ) function ထဲမှာ *x နဲ့ *y ကိုနေရာချင်းပြောင်းပေးပါတယ်။ ဒါဆိုရင် *x = 15 နဲ့ *y = 95 ဖြစ်သွားပါပြီ။ main( ) function ကိုပြန်ရောက်တဲ့အခါမှာ ဒီဂဏန်း (2) ခုကို display လုပ်ပြဦးမှာပါ။
- Ex408.cpp program ကို run လိုက်ရင် output result ကို ပုံ (၄. ၁၈) မှာပြထားတဲ့အတိုင်းမြင်ရပါလိမ့်မယ်။



ပုံ (၄. ၁၈)

Bubble Sort

၁။ ပုံ (၄. ၁၉) မှာပြထားတဲ့ Ex409.cpp program ဟာဆိုရင် array တစ်ခုထဲက element တွေကို ငယ်စဉ်ကြီးလိုက်ဖြစ်အောင် pointer အသုံးပြုပြီး bubble sort နည်းနဲ့စီပေးတဲ့ program ဖြစ်ပါတယ်။ ဒီဥစ္စာကို trace လုပ်ကြည့်မယ်ဆိုရင်

Ex409.cpp

```
// Listing 4.9: This program sorts an array
// using the bubble sort method

#include <iostream>
void    swap(int*, int*);
void    bubbleSort(int*, int);

int main()
{
        const int  N=10;
        int  x[N] = {10,50,20,70,40,30,90,80,60,100};

        cout << "\nßefore sorting\n";
        for(int j=0;j<N;j++)
                cout << x[j] << "  ";
        bubbleSort(x,N);
        cout << "\n\nAfter sorting\n";
        for(int j=0;j<N;j++)
                cout << x[j] << endl;
        return  0;
}

void  swap(int *x, int *y)
{
        if(*x > *y)  {
                int  temp = *x;
                *x = *y;
                *y = temp;
        }
}

void  bubbleSort(int*  xp, int  n)
{
        for(int j=0; j<(n-1); j++)
                for(int k=j+1; k<n;k++)
                        swap(xp+j, xp+k);
}
```

ပုံ (၄.၁၉)

၂။ Ex409.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- bubbleSort( ) function ထဲမှာ inner loop ပတ်နေတုန်း ပထမဂဏန်း (10) နဲ့ ဒုတိယဂဏန်း (50) ကိုနှိုင်းယှဉ်ကြည့်ပါတယ်။ ဒုတိယဂဏန်းက ပထမဂဏန်းထက်ငယ်တယ်ဆိုရင် နေရာချင်းလဲပါမယ်။ မငယ်ဘူးဆိုရင် ပထမဂဏန်းနဲ့ တတိယဂဏန်း (20) နဲ့ဆက်ယှဉ်ကြည့်မယ်။ တတိယဂဏန်းက ပထမဂဏန်းထက်မငယ်ဘူးဆိုတော့ နေရာချင်းမလဲတော့ပါဘူး။ ဒီနည်းအတိုင်း ဆက်လုပ်သွားရင် outer loop အတွက် ပထမအကြိမ်ပြီးသွားပါပြီ။ ပထမအကြိမ်ပြီးတဲ့အခါမှာ x[ ] array ကအခုလိုဖြစ်နေပါလိမ့်မယ်။ ဘာမှပြောင်းလဲမှုမရှိသေးပါဘူး။

  10, 50, 20, 70, 40, 30, 90, 80, 60, 100

- outer loop ဒုတိယအကြိမ်စရင် x[ ] array ရဲ့ဘယ်ဘက်အစွန်ဆုံးဂဏန်းဟာ အငယ်ဆုံးဖြစ်နေပါလိမ့်မယ်။ အဲဒီဂဏန်းကို ခနဖယ်ထားပါ။ အခုတစ်ခါတော့ ပထမဂဏန်းကို (50) လို့သတ်မှတ်ရပါမယ်။ ကျန်တဲ့ဂဏန်း (9) လုံးထဲက အငယ်ဆုံးဖြစ်တဲ့ (20) ကို ဒုတိယဘယ်အစွန်နေရာရောက်အောင် စောစောကလိုပဲ swap လုပ်ခဲ့ရင် x[ ] array ကအခုလိုဖြစ်နေပါလိမ့်မယ်။

  10, 20, 50, 70, 40, 30, 90, 80, 60, 100

- ဒီနည်းအတိုင်းဆက်လုပ်သွားရင် x[ ] array element တွေဟာ အခုလို တစ်ဆင့်ပြီးတစ်ဆင့် နေရာပြောင်းသွားပြီး sorting ဟာပြီးသွားပါလိမ့်မယ်။

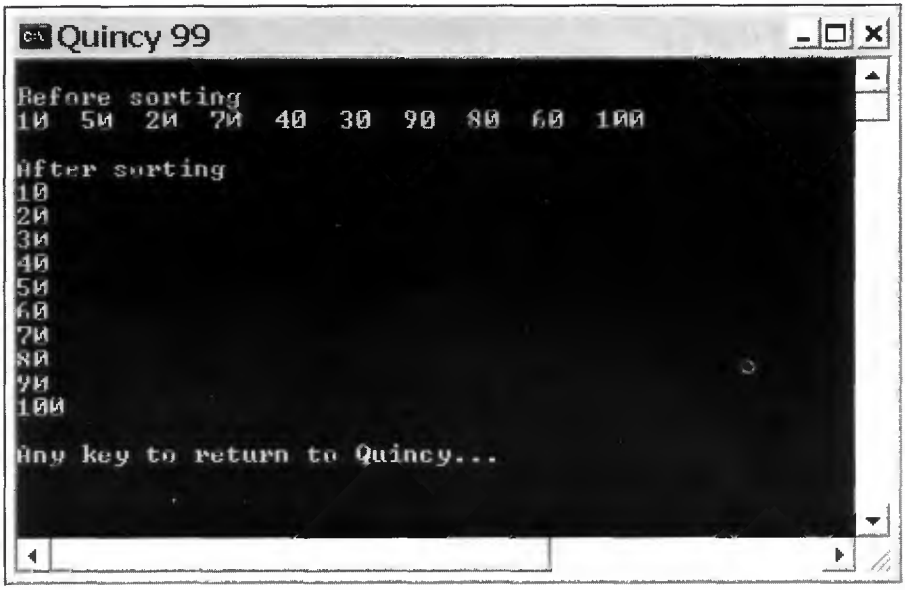
  10, 20, 30, 50, 70, 40, 90, 80, 60, 100

  10, 20, 30, 40, 50, 70, 90, 80, 60, 100

  10, 20, 30, 40, 50, 60, 70, 90, 80, 100

  10, 20, 30, 40, 50, 60, 70, 80, 90,100

- Ex409.cpp program ကို run လိုက်ရင် output result ကို ပုံ (၄. ၁၈) မှာပြထားတဲ့အတိုင်း မြင်ရပါလိမ့်မယ်။



ပုံ (၄. ၂၀)

၄.၆ Pointers and Strings

၁။ string ဆိုတာ character type array အမျိုးအစားတစ်ခုဖြစ်ပါတယ်။ ဒီတော့ array element တွေကို pointer notation နဲ့ access လုပ်လို့ရရင် string တွေကိုလည်းလုပ်လို့ရပြီပေါ့။ string array တစ်ခုကို ဒီတိုင်း increment လုပ်လို့မရပေမယ့် pointer ကိုတော့လုပ်လို့ရပါတယ်။ ပုံ (၄. ၂၀) မှာဖော်ပြထားတဲ့ Ex4010.cpp program ဟာဆိုရင် pointer notation ကိုအသုံးပြုပြီး ရေးထားတာဖြစ်ပါတယ်။

၂။ Ex4010.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

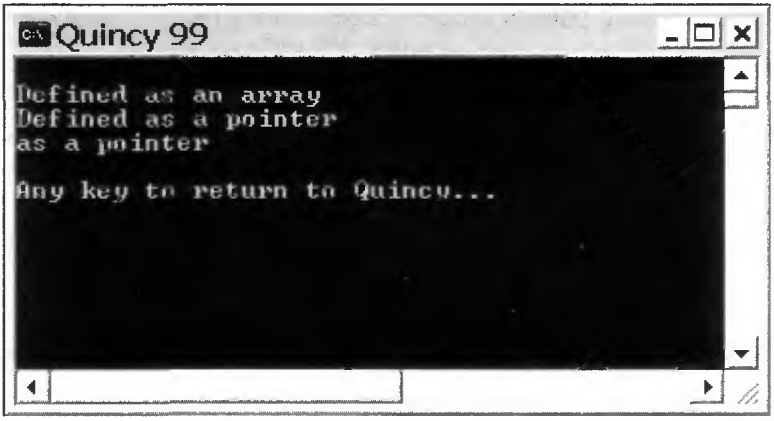
- str နဲ့ strp (pointer to str) တို့ကို define လုပ်ပြီး initialize လည်းလုပ်ပါတယ်။ နောက်ပြီး str နဲ့ strp တို့ကို display လုပ်ကြည့်တော့ရပါတယ်။
- ဒီတစ်ခါ for loop ထဲမှာ strp ကို increment လုပ်ပေးခြင်းအားဖြင့် string str ထဲက character (8) လုံးဟာ pointer variable ထဲမှာ store ဖြစ်သွားပါပြီ။ strp ကို display လုပ်ကြည့်ရင် အဖြေပေါ်လာပါလိမ့်မယ်။

Ex4010.cpp

```
// Listing 4.10: This program shows how the strings
// can be defined using array and pointer notations

#include <iostream>

int main( )
{
        char  str[ ]= "Defined as an array";
        char* strp = "Defined as a pointer";

        cout << endl << str << endl << strp;

        for (int i=0; i<8; i++)
                strp++;
        cout << endl << strp << endl;
        return 0;
}
```

ပုံ (၄.၂၁)

■ Ex4010.cpp program ကို run လိုက်ရင် output result ကို ပုံ (၄.၂၂) မှာပြထားတဲ့အတိုင်း မြင်ရပါလိမ့်မယ်။ str string ထဲက character (8) လုံးပဲပေါ်နေတာကိုတွေ့မှာပါ။



ပုံ (၄.၂၂)

Changing Characters of an Array to Uppercase

ပုံ (၄.၂၃) မှာပြထားတဲ့ Ex4011.cpp program ဟာ character array တစ်ခုထဲက element တွေကို uppercase letter တွေချည်းဖြစ်အောင် pointer အသုံးပြုပြီးပြောင်းပေးတဲ့ program ဖြစ်ပါတယ်။ ပုံ (၄.၂၄) မှာ Ex4011.cpp program ကို run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex4011.cpp

```
// Listing 4.11: This program requests a string from
// the user and then displays the string in uppercase letters
#include <iostream>
#include <cctype>

void copyUpper(char* strp)
{
        while(*strp)     *strp++ = toupper(*strp);
}

int main( )
{
        char  str[80];

        cout << "\nEnter a string\n";
        cin.get(str,80);
        copyUpper(str);
        cout << endl << str << endl;
        return 0;
}
```

ပုံ (၄.၂၃)

Quincy 99

```
Enter a string
Pointers are fun to use.

POINTERS ARE FUN TO USE.

Any key to return to Quincy...
```

ပုံ (၄.၂၄)

Strings as Function Arguments

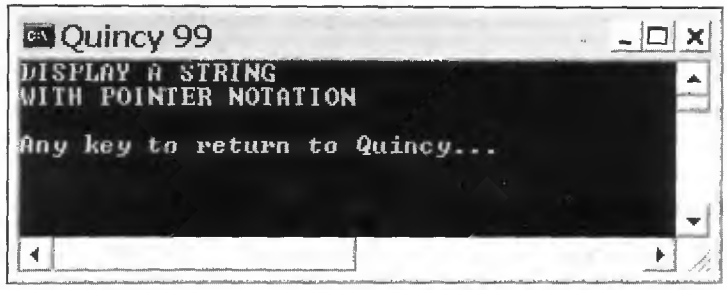
၁။ ပုံ (၄.၂၅) မှာပြထားတဲ့ Ex4012.cpp program ဟာဆိုရင် string တစ်ခုကို function တစ်ခုမှာ passing by pointer နည်းနဲ့ pass လုပ်တာကို program ရေးပြထားတာဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex4012.cpp

```
// Listing 4.12: This program shows a string
// used as a function argument. The function
// displays the string each character in turn.

#include <iostream>
void  displayStr(char*);

int main()
{
        char  str1[] = "DISPLAY A STRING\n";
        char  str2[] = "WITH POINTER NOTATION\n";

        displayStr(str1);
        displayStr(str2);
        return 0;
}

void  displayStr(char*  cp)
{
        while (*cp)  cout << *cp++;
}
```

ပုံ (၄.၂၅)

၂။ Ex4012.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၄.၂၆) မှာပြထားတဲ့အတိုင်း output result ကိုမြင်ရမှာပါ။ string တစ်ခုကို pointer ထဲ pass လုပ်ပေးပြီး pointer ကို increment လုပ်ရင်း string ထဲက character တစ်ခုချင်းကို display လုပ်ပြသွားတာပါပဲ။



ပုံ (၄. ၂၆)

Copying a String Using Pointers

၁။ ပုံ (၄. ၂၇) မှာဖော်ပြထားတဲ့ Ex4013.cpp program ထဲက main( ) function ထဲမှာ copyStr( ) function ကို call ခေါ်ပြီး blank string တစ်ခုနဲ့ pointer array တစ်ခုကို pass လုပ်ပေးပါတယ်။ copyStr

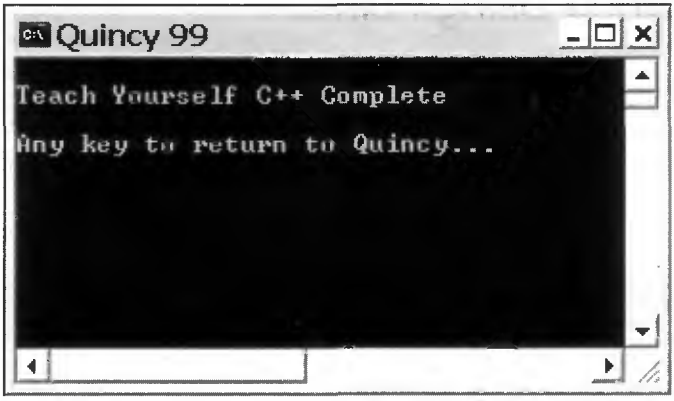
Ex4013.cpp

```
// Listing 4.13: This program copies one string
// to another using pointers.

#include <iostream>
void    copyStr(char*, char*);

int main( )
{
        char*   strp = "Teach Yourself C++ Complete";
        char    str[80];

        copyStr(str, strp );
        cout << endl << str << endl;
        return 0;
}

void  copyStr(char*  to, char*  from)
{
        while (*from)
                *to++ = *from++;
        *to = '\0';
}
```

ပုံ (၄. ၂၇)

function ထဲ ရောက်လာတဲ့အခါမှာ from pointer array ကို increment လုပ်ရင်းနဲ့ blank string array ကို point လုပ်နေတဲ့ to pointer array ကိုလည်း increment လုပ်သွားပေးပါမယ်။ to++ ကို from++ နဲ့ တစ်ခုပြီးတစ်ခု assign လုပ်ပေးသွားတဲ့သဘောပေါ့။ ဒါဆိုရင် copy ကူးတဲ့အလုပ်ပြီးသွားပါပြီ။

၂။ Ex4013.cpp program ကို run လိုက်မယ်ဆိုရင် Teach Yourself C++ Complete ဆိုတဲ့ string ဟာ string str ထဲကို copy ကူးဝင်သွားပါတယ်။ ဒီတော့ str ကို display လုပ်ရင် စောစောကကူးဝင်လာတဲ့ စာကြောင်းကို ပုံ (၄. ၂၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၄. ၂၆)

၄.၇ Arrays of Pointers to Strings

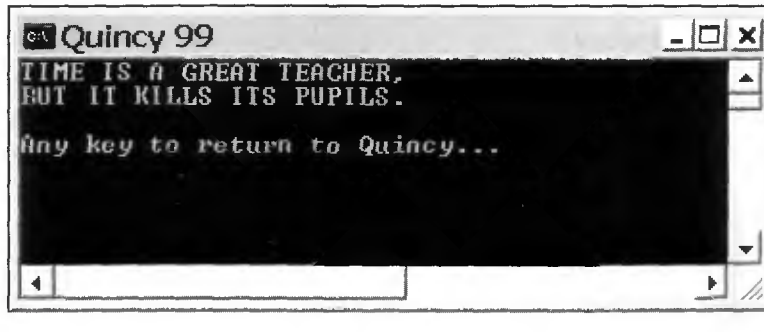
၁။ integer type ၊ float type ဖြစ်တဲ့ array တွေရှိသလို pointer type array တွေလည်း ရှိပါတယ်။ string တွေကို ရိုးရိုးတန်းတန်း array ပုံစံမျိုးနဲ့ access လုပ်တဲ့အခါမှာ အားနည်းချက်တစ်ခုက subarray တွေဟာ တူညီတဲ့ string length လေးတွေကိုပဲ အသုံးပြုလို့ရပါတယ်။ ဒီတော့ မလိုတဲ့ space နေရာတွေဟာ ဖြုန်းသလိုဖြစ်နေ တာပေါ့။ အဲဒီအားနည်းချက်ကို pointer type array တွေအသုံးပြုခြင်းအားဖြင့် လုံးဝချဖျက်လို့ရပါပြီ။ ပုံ (၄. ၂၇) မှာဖော်ပြထားတဲ့ Ex4014.cpp program ကိုလေ့လာကြည့်ရင် သဘောပေါက်ပါလိမ့်မယ်။

Ex4014.cpp

```
// Listing 4.14: This program creates an array of pointers
// to strings, rather than an array of strings.
// to another using pointers.

#include <iostream>
const int MAX= 7;

int main()
{
        char* strp[MAX] = {"TIME ", "IS ", "A GREAT ",
                           "TEACHER,\n", "BUT ","IT ",
                           "KILLS ITS PUPILS."};

        for (int i=0; i<MAX; i++)
                cout << strp[i];

        cout << endl;
        return 0;
}
```

ပုံ (၄. ၂၇)

၂။ Ex4014.cpp program ကို run လိုက်မယ်ဆိုလို့ရှိရင် strp[i] pointer array ထဲက pointer element တစ်ခုချင်းကနေ point လုပ်နေတဲ့ string အရွယ်အစားအမျိုးမျိုးကို ရိုးရိုး array တစ်ခုရဲ့ element တွေကို display လုပ်နည်းအတိုင်း display လုပ်ပြပါလိမ့်မယ်။ ပုံ (၄. ၂၈) ကိုကြည့်ပါ။



ပုံ (၄. ၂၈)

၄.၈ Pointers to Pointers

၁။ pointer ကနေ pointer ကို point လုပ်ပေးချင်ရင် * (asterisk) (2) ခုကို အသုံးပြုရပါမယ်။ ဥပမာ cp ဟာ char pointer တစ်ခုဆိုရင် char** cpp = &cp; လို့ရေးလို့ရပါတယ်။ cpp ဟာ pointer to pointer ဖြစ်ပါတယ်။ pointer to pointer ကို initialize လုပ်ပေးချင်ရင် အခုလိုရေးကြည့်ပါ။

```
char   c = 'A';          // char variable
char*  cp = &c;          // pointer to char variable
char** cpp = &cp;        // pointer to pointer
```

pointer to pointer ကနေ point လုပ်နေတဲ့ pointer သို့မဟုတ် char variable တွေက data တွေ ကိုထုတ်ယူချင်ရင် char*cp1 = *cpp; သို့မဟုတ် char c1 = **cpp; လို့ရေးပြီး retrieve လုပ်ရင်ရပါတယ်။ ကောင်းပြီ ၊ ပုံ (၄. ၂၉) မှာရေးထားတဲ့ Ex4016.cpp program ကိုလေ့လာကြည့်ရင် pointer to pointer ကို သဘောပေါက်ပါလိမ့်မယ်။

Ex4016.cpp

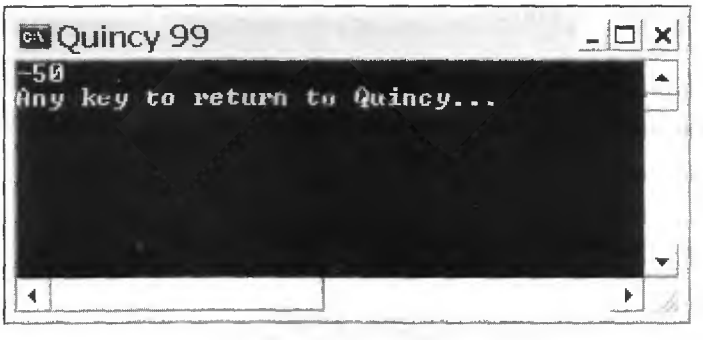
```
// Listing 4.16: Pointers to pointers
#include <iostream>

void   firstNeg(float** xpp)
{
        while (**xpp != 0)
                if (**xpp < 0)   break;
                else             (*xpp)++;
}

int main( )
{
        float   x[ ] = {10,25,-50,36.5,-42,0};
        float*  xp = x;

        firstNeg(&xp);
        cout << *xp;
        return 0;
}
```

ပုံ (၄. ၂၉)

Ex4016.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ float type array ဖြစ်တဲ့ x ကို initialize လုပ်ပါတယ်။ နောက်ပြီး x ကို point လုပ်နေတဲ့ pointer xp ကို define လုပ်ပါတယ်။ firstNeg( ) function ကို call ခေါ်ပါတယ်။ function မှာ pointer address ကို pass လုပ်ပေးပါတယ်။
- called function မှာ argument က pointer to pointer xpp ဖြစ်ပါတယ်။ float **xpp လို့ define လုပ်ထားတာကိုကြည့်ပါ။ while(**xpp != 0) ဆိုတဲ့ statement ဟာ ဆိုရင် pass ဝင်လာတဲ့ (**xpp point လုပ်နေတဲ့ x[ ] array element) ကို သုညဟုတ်မဟုတ်စီစစ်မှာပါ။ မဟုတ်ဘူးဆိုရင် if (**xpp < 0) break; statement ကနေ (x[ ] array element ဟာ negative value ဖြစ်ခဲ့ရင်) function ထဲကပြန်ထွက်ခိုင်းပါလိမ့်မယ်။ တစ်ကယ်လို့ negative value မဟုတ်သေးဘူးဆိုရင် **xpp ကို increment လုပ်ခြင်းအားဖြင့် next element ကို point လုပ်ပေးမှာပါပဲ။

၃။ Ex4016.cpp program ကို run လိုက်မယ်ဆိုရင် ပထမဆုံးတွေ့တဲ့ negative value ကို display လုပ်ပြပါလိမ့်မယ်။ ပုံ (၄.၃၀) ကိုကြည့်ပါ။



ပုံ (၄.၃၀)

Pointers to Arrays of Pointers

၁။ pointer to pointer ကိုနောက်တစ်မျိုးအသုံးပြုလို့ရတာက pointer array တွေကို point လုပ်ချင်တဲ့အခါမှာဖြစ်ပါတယ်။ တစ်ချို့က ရိုးရိုး array မျိုးကိုအသုံးပြုတာထက် pointer array အမျိုးအစားကိုအသုံးပြုတာ

ပိုကောင်းတယ်လို့ပြောပါတယ်။ ပုံ (၄. ၃၁) က Ex4017.cpp program ဟာဆိုရင် pointer array တစ်ခုကို pointer to pointer နဲ့ access လုပ်နည်းကိုတင်ပြထားတာပါ ၊ လေ့လာကြည့်ပါ။

Ex4017.cpp

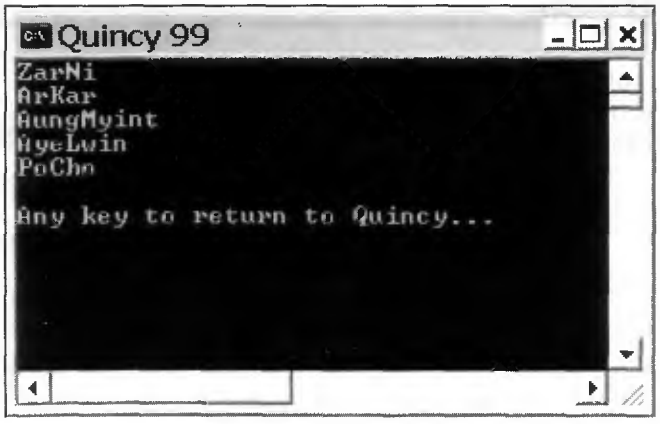
```
// Listing 4.17: Pointers to arrays of pointers
#include <iostream>

char* str[ ] = { "ZarNi",
                 "ArKar",
                 "AungMyint",
                 "AyeLwin",
                 "PoCho",
                 0   };

int main( )
{
        char** cp = str;

        while (*cp != 0)
                cout << *cp++ << endl;
        return 0;
}
```

ပုံ (၄. ၃၁)

၂။ Ex4017.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ char pointer array တစ်ခုကို နာမည်တွေနဲ့ initialize လုပ်ပါတယ်။ point array ရဲ့နောက်ဆုံး element ကို null pointer လုပ်ထားပါတယ်။ main( ) function ထဲမှာ pointer to pointer cp ကို pointer array နဲ့ assign လုပ်ပေးပြီး while loop ထဲမှာ *cp ကို pointer array မကုန်မချင်း တစ်ခုစီ display လုပ်သွားပါတယ်။ *cp++ ဆိုတာ next pointer array element တွေကို point ဆက်လုပ်သွားတာဖြစ်ပါတယ်။

- ပုံ (၄. ၃၂) မှာ Ex4017.cpp program ကို run ပြထားပါတယ်၊



ပုံ (၄.၃၂)

Pointers to const Variables

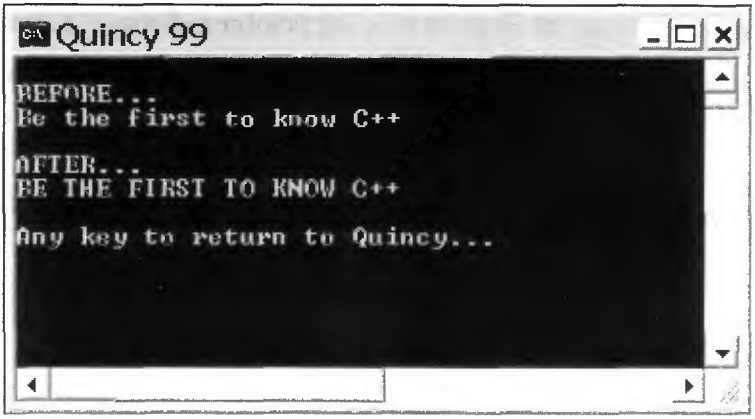
၁။ ပုံ (၄.၃၃) မှာရေးထားတဲ့ Ex4018.cpp program မှာ pointer to const variable ရေးနည်းကို တင်ပြထားပါတယ်။ main( ) function ထဲမှာ const char variable တစ်ခုဖြစ်တဲ့ str2[ ] ကို string တစ်ခုနဲ့ initialize လုပ်ပေးပြီး copyUpper( ) function ကို call ခေါ်ပါတယ်။ calling function မှာ str1 နဲ့ str2 char array (2) ခုကို pass လုပ်ပေးပြီး called function ကနေ char pointer တွေဖြစ်တဲ့ s1 ၊ s2 တို့က လက်ခံပါတယ်။ function return type ကို char pointer လို့သတ်မှတ်ပေးထားတာကို သတိပြုကြည့်ပါ။ called function ထဲမှာ s1 ကို point လုပ်နေတဲ့ pointer ကို s လို့ assign လုပ်ပေးပြီး s2 point လုပ်နေတဲ့ str2 ထဲက character တွေကိုတစ်ခုချင်း uppercase letter ဖြစ်အောင်ပြောင်းရင်းနဲ့ *s1++ နဲ့ assign လုပ်ပေးပါတယ်။ အားလုံးပြောင်းပြီးသွားရင် s1 ကို point လုပ်နေတဲ့ pointer s ကို main( ) ဆီ return လုပ်ပေးမှာပါ။ ဒီတော့ main( ) function မှာ str1 ကို display လုပ်ခိုင်းရင် str2 ကို uppercase ပြောင်းထားတာကို တွေ့ရပါလိမ့်မယ်။

၂။ Ex4018.cpp program ကို run လိုက်မယ်ဆိုရင် ကွန်ပျူတာမှာ BEFORE... နဲ့ AFTER... ဆိုတဲ့ စာကြောင်း (4) ကြောင်းကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၄.၃၄) ကိုကြည့်ပါ။

Ex4018.cpp

```
// Listing 4.18: const pointer arguments
#include <iostream>
#include <cctype>

char* copyUpper(char* s1, const char* s2)
{
        char* s = s1;

        while ((*s1++ = (toupper(*s2++))) != '\0');
        return s;
}

int main( )
{
        char   str1[80];
        const char str2[ ] = "Be the first to know C++";

        cout << "\nBEFORE...\n" << str2 << endl;

        copyUpper(str1, str2);
        cout << "\nAFTER...\n" << str1 << endl;
        return 0;
}
```

ပုံ (၄.၃၃)



ပုံ (၄.၃၄)

const Pointer Variables

ပုံ (၄.၃၄) က Ex4019.cpp program မှာ char array s[ ] ကို showUpper( ) function ထဲ pass လုပ်တဲ့အခါမှာ array address အဖြစ် pass လုပ်ပေးပြီး called function မှာ char const pointer variable တစ်ခုအဖြစ်လက်ခံပါတယ်။ အဲဒီ const pointer variable ကို uppercase ဖြစ်အောင်ပြောင်းလိုက်တဲ့အခါမှာ မူလ char array s ဟာ uppercase letter တွေချည်းဖြစ်ကုန်ပါပြီ။ ဒီ program ကို run လိုက်ရင် Hello C++ complete နဲ့ HELLO C++ COMPLETE ဆိုတဲ့စာကြောင်းတွေပေါ်လာမှာပါ။ စာဖတ်သူကိုယ်တိုင် လက်တွေ့လုပ်ကြည့်ပါ။

Ex4019.cpp

```
// Listing 4.19: A const pointer to char
#include <iostream>
#include <cctype>

void  showUpper(char* const  str)
{
        int  i = 0;

        while (*(str+i))
        {
                *(str+i) = toupper(*(str+i));
                i++;
        }
}

int main( )
{
        char  s[ ] = "Hello C++ Complete";

        cout << s << endl;
        showUpper(s);
        cout << s << endl;
        return 0;
}
```

ပုံ (၄.၃၄)

၄.၉ Memory Allocation

၁။ ကွန်ပျူတာတစ်ခုရဲ့ global storage memory ကို free store သို့မဟုတ် heap လို့ခေါ်ပါတယ်။ heap ဆိုတာ program အတွက် အသုံးပြုလို့ရတဲ့ RAM အရွယ်အစား ဖြစ်ပါတယ်။ program တစ်ခုအတွက် memory allocation ကို သတ်မှတ်ပေးချင်ရင် new ဆိုတဲ့ operator ကိုအသုံးပြုရမှာပါ။

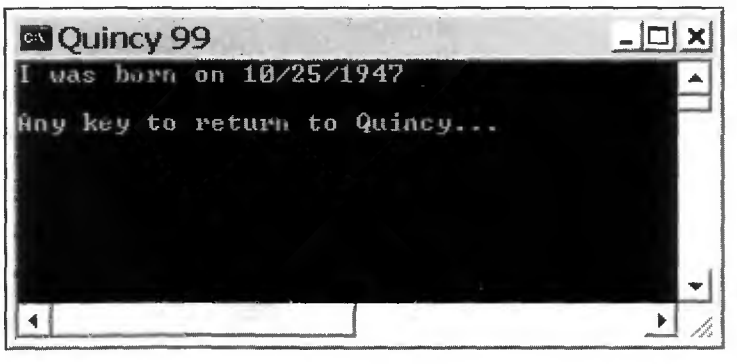
Ex4020.cpp

```
// Listing 4.20: The new and delete operators
#include <iostream>

// Declare a date structure
struct Date
{
        int  month;
        int  day;
        int  year;
};

int main()
{
        // Allocate memory for a date structure
        Date*  birthday = new Date;

        // Assign values to the structure members
        birthday->month = 10;
        birthday->day = 25;
        birthday->year = 1947;

        // Display the date structure
        cout << "I was born on "
             << birthday->month << '/'
             << birthday->day << '/'
             << birthday->year << endl;

        // Return the allocated memory to the heap
        delete  birthday;
        return 0;
}
```

ပုံ (၄.၃၅)

ဒါဆိုရင် heap memory ကနေ program အတွက် memory block တွေကို assign လုပ်ပေးပါလိမ့်မယ်။ program ပြီးသွားလို့ block တွေကို heap ဆီ ပြန်ပို့ပေးချင်ရင် delete operator ကိုအသုံးပြုရပါမယ်။ ကောင်းပြီ၊ ပုံ (၄. ၃၅) မှာဖော်ပြထားတဲ့ Ex4020.cpp program မှာ new နဲ့ delete operator တွေအသုံးပြုပုံကို တင်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex4020.cpp program ကို လေ့လာကြည့်မယ်ဆိုရင် main( ) function ထဲမှာ စစချင်း birthday date တစ်ခုအတွက် memory block ကို heap ကနေရယူပါတယ်။ ပြီးတော့ရင် အဲဒီ date မှာ month ၊ day၊ year value တွေကို assign လုပ်ပေးပါတယ်။ ဒါဆိုရင် birthdaymonth ကို display လုပ်ရင် birthday အတွက် month value ကို တွေ့ရမှာပါ။ birthday date အတွက် memory allocation ကို ပြန်ဖျက်ချင်ရင် delete birthday; လို့ ရေးတာနဲ့ရပါတယ်။ ပုံ (၄. ၃၆) မှာ Ex4020.cpp program ကို run ပြထားပါတယ်။



ပုံ (၄. ၃၆)

Using new and delete Operators with Arrays

တစ်ကယ်လို့ birthday date ကို array တစ်ခုအနေနဲ့ဖော်ပြမယ်ဆိုရင် memory allocation ကိုအခုလိုပြင်ရေးရပါမယ်။ ပုံ (၄. ၃၇) မှာဖော်ပြထားတဲ့ Ex4021.cpp program ဟာ Ex4020.cpp program ကို ပြင်ရေးထားတာဖြစ်ပါတယ်။ array အတွက် allocated memory ကို ဖျက်ချင်ရင် delete [ ] birthday လို့ ရေးရပါမယ် ၊ သတိထားပါ။ ဒီ program ကို run ရင် စောစောကအဖြေပဲရမှာပါ။

Ex4021.cpp

```
// Listing 4.21: The new and delete operators
#include <iostream>

int main()
{
        // Get memory for an array of integers
        int* birthday = new int[3];

        // Assign values to the array elements
        birthday[0] = 10;
        birthday[1] = 25;
        birthday[2] = 1947;

        // Display the values in the array
        cout << "I was born on "
             << birthday[0] << '/'
             << birthday[1] << '/'
             << birthday[2] << endl;

        // Return the allocated memory to the heap
        delete [] birthday;
        return 0;
}
```

ပုံ (၄.၃၇)

Allocating Dynamic Arrays

၁။ ပုံ (၄.၃၈) မှာဖော်ပြထားတဲ့ Ex4022.cpp program ဟာ variable array dimension တစ်ခုကို new operator အသုံးပြုပြီး allocate လုပ်နည်းကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

၂။ Ex4022.cpp program ကို run လိုက်ပြီး Enter the array size : လို့ prompt လုပ်ပြတဲ့အခါမှာ (5) ကိုရိုက်ထည့်မယ်ဆိုရင် ကွန်ပျူတာမှာ random number (5) လုံးပေါ်လာတာကိုတွေ့ရပါလိမ့်မယ်။ အဲဒါ လုပ်ပြီးရင် array အတွက် memory allocation ကို program က delete လုပ်ပေးလိုက်ပါပြီ။ ပုံ (၄.၃၉) ကိုကြည့်ပါ။

Ex4022.cpp

```cpp
// Listing 4.22: The new operator and dynamic arrays
# include <iostream>
# include <cstdlib>

int      main( )
{
         int    size;

         // Get the array size from the user
         cout << "Enter the array size: ";
         cin >> size;

         // Allocate memory for the array
         int*    arrayX = new   int[size];

         // Load the array with random numbers
         for (int i=0; i<size; i++)
         arrayX[i] = rand( );

         // Display the array with random numbers
         for (int i=0; i<size; i++)
         cout << endl << arrayX[i];

         // Return the allocated memory to the heap
         delete [ ] arrayX;
         return 0;
}
```

ပုံ (၄. ၃၈)

Quincy 99

```
Enter the array size: 5

41
18467
6334
26500
19169
Any key to return to Quincy...
```

ပုံ (၄. ၃၉)

၄.၁၀ Using Reference Variables

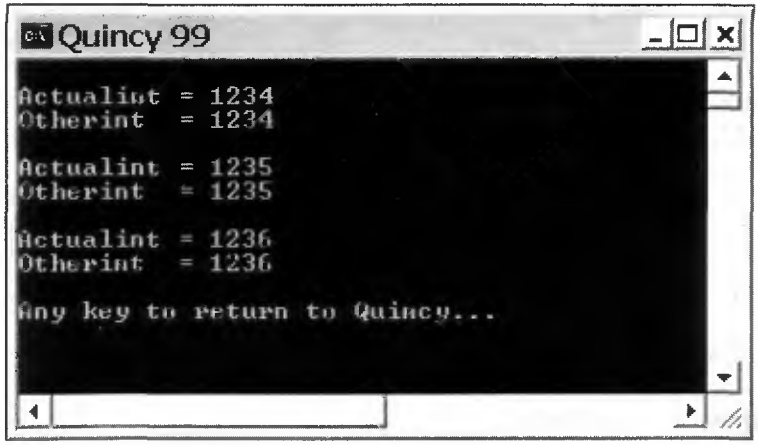
၁။ reference variable ဆိုတာ data object တစ်ခုရဲ့ copy မဟုတ်သလို data object ကို point လုပ်နေတဲ့ pointer လည်းမဟုတ်ပါဘူး။ address ချင်းတူညီတဲ့ တစ်ခြား variable တစ်ခုရဲ့အမည်ကွဲ (alias) တစ်ခုပါပဲ။ variable တစ်ခုကို &(ampersand) operator အသုံးပြုပြီး reference variable ဖြစ်အောင် သတ်မှတ်ပေးလို့ရပါတယ်။ ပုံ (၄. ၄၀) မှာဖော်ပြထားတဲ့ Ex4023.cpp program မှာ reference variable ရဲ့ သဘောတရားကို လေ့လာလို့ရပါတယ်။

Ex4023.cpp

```
// Listing 4.23: Reference variable
# include <iostream>

int     main( )
{
        int     actualint = 1234;
        int&    otherint  = actualint;

        cout << "\nActualint = " << actualint << endl;
        cout << "Otherint  = " << otherint  << endl;
        otherint++;

        cout << "\nActualint = " << actualint << endl;
        cout << "Otherint  = " << otherint  << endl;
        actualint++;

        cout << "\nActualint = " << actualint << endl;
        cout << "Otherint  = " << otherint  << endl;
        return 0;
}
```

ပုံ (၄. ၄၀)

၂။ Ex4023.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၄. ၄၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ actualint နဲ့ otherint တို့ရဲ့ ဆက်သွယ်မှုကိုလေ့လာကြည့်ပါ။



ပုံ (၄.၄၁)

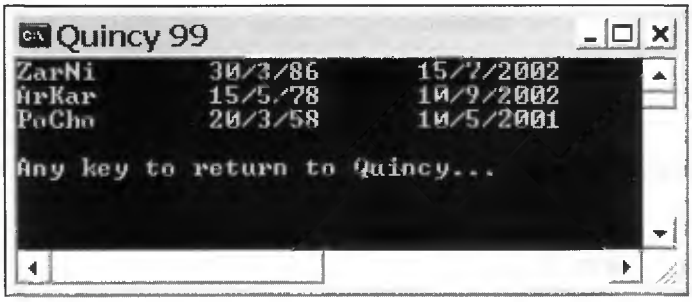
References to Reduce Complex Notation

၁။ C++ program မှာ structure member တွေလို အရှည်ကြီးဖေါ်ပြရမယ့်နေရာမှာ reference ကို အသုံးပြုရင် လွယ်ကူတာကိုတွေ့ရပါတယ်။ အဲဒီအတွက် Ex4024.cpp program ကို နမူနာအဖြစ် အောက်မှာရေး ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

```
// Listing 4.24: Initializing a reference

#include <iostream>

struct  Date
{
        int     month,day,year;
};

struct EmployeeRec
{
        int     emplNo;
        char    name [80];
        Date    dates [3];
        float   salary;
};
```

```
EmployeeRec     staff[ ] =
{
        {001, "ZarNi",{{30,3,86},{15,7,2002}}, 45000},
        {002, "ArKar",{{15,5,78},{10,9,2002}}, 38000},
        {003, "PoCho",{{20,3,58},{10,5,2001}}, 35000},
        {0}
};


int  main( )
{
        EmployeeRec*   xp = staff;

        while (xp->emplNo != 0)
        {
                cout << xp->name << "\t     ";
                for (int i=0; i<2; i++)
                {
                        Date&   rd = xp->dates[i];
                        cout << rd.month << '/'
                    << rd.day   << '/'
                    << rd.year  << "       ";
                }
                cout << endl;
                xp++;
        }
        return 0;
}
```

ဒီ Ex4023.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၄.၄၂) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၄.၄၂)

Passing References

၁။ function တစ်ခုကနေ reference တစ်ခုကို တစ်ခြား function ဆီ pass လုပ်ပေးမယ်ဆိုရင် calling function က argument copy ကို called function ကအသုံးပြုမှာပါ။ called function ရဲ့ local copy အသုံးမပြုပါဘူး။ ပုံ (၄. ၄၃) မှာဖော်ပြထားတဲ့ Ex4025.cpp program ဟာ reference pass လုပ်နည်းကို ရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

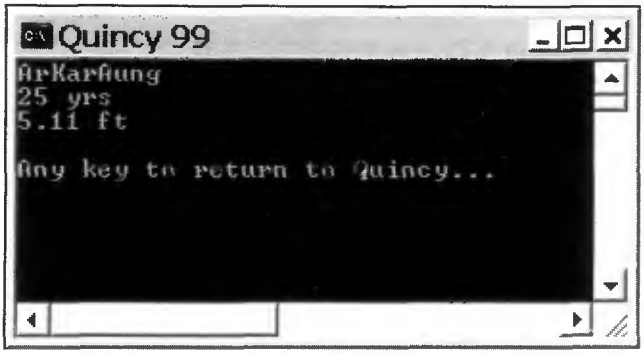
Ex4025.cpp

```
// Listing 4.25: Passing references
#include <iostream>

struct history
{
        char   name[80];
        int    age;
        float  height;
};

void  show(history&);  // call by reference

int   main()
{
        history  hs = {"ArKarAung",25, 5.11};

        show(hs);
        return 0;
}

void  show(history&  x)
{
        cout << x.name << endl
             << x.age << " yrs\n"
             << x.height << " ft\n";
}
```

ပုံ (၄. ၄၃)

၂။ Ex4025.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ structure type history ကို define လုပ်ပြီး show( ) ဆိုတဲ့ prototype ကိုလည်း declare လုပ်ပါတယ်။
- main( ) function ထဲမှာ history instance တစ်ခုဖြစ်တဲ့ hs ကို initialize လုပ်ပြီးတော့ show( ) function ကို call ခေါ်ပါတယ်။ called function ထဲမှာ call by reference နည်းကိုအသုံးပြုပြီး hs member တစ်ခုချင်းကို display လုပ်ပြခိုင်းထားပါတယ်။
- ပုံ (၄. ၄၄) ဟာ Ex4025.cpp program ကို run ပြထားတာပါ။



ပုံ (၄. ၄၄)

Returning References

၁။ reference တစ်ခုကို parameter တစ်ခုအနေနဲ့ function ထဲ pass လုပ်ပေးတဲ့နည်းကို စာဖတ်သူသိသွားပြီမဟုတ်လား။ တစ်ကယ်လို့ reference တစ်ခုကို function ကနေ return လုပ်ပေးတာမျိုးကိုလုပ်ချင်ရင်ရမလား။ ဥပမာ Date& getDate( ); ဆိုတာမျိုးပေါ့။ အဲဒါမျိုးလည်း လုပ်လို့ရပါတယ်။ ပုံ (၄. ၄၅) မှာဖော်ပြထားတဲ့ Ex4026.cpp program ဟာ reference return လုပ်နည်းကို ရေးပြထားပါတယ်။

၂။ Ex4026.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စစချင်းမှာ structure type Date ကို define လုပ်ပြီး Date instance ဖြစ်တဲ့ birthdays

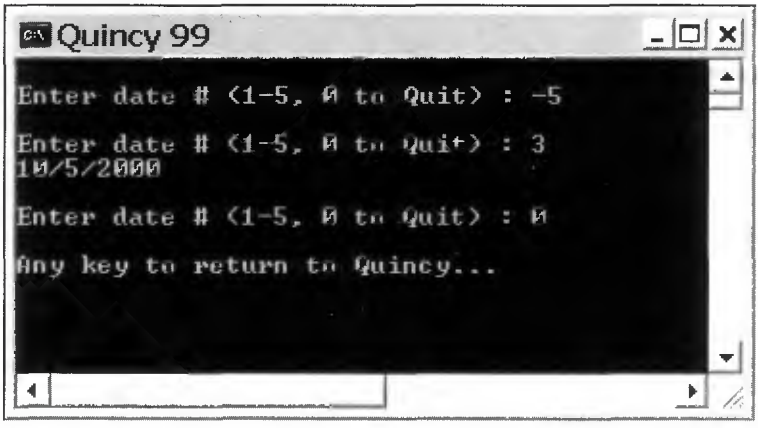
Ex4026.cpp

```
// Listing 4.26: Returning references
#include <iostream>

struct Date
{
        int   month, day, year;
};

Date   birthdays[ ] = {   {12, 10, 1995},
                         { 7, 11, 1998},
                         {10,  5, 2000},
                         {15,  7, 2001},
                        { 9, 10, 2002},
};

const  Date&  getDate(int  n)
{
        return  birthdays[n-1];
}

int   main( )
{
        int  dt = 2003;
        while (dt != 0)
        {
                cout << "\nEnter date # (1-5, 0 to Quit) : ";
                cin >> dt;
                if (dt>0 && dt<6)
                {
                        const  Date&  rd = getDate(dt);
                        cout << rd.month << '/' << rd.day << '/'
                             << rd.year << endl;
                }
        }
        return 0;
}
```

ပုံ (၄.၅၅)

ကို initialize လုပ်ထားပါတယ်။ နောက်ပြီး Date တစ်ခုကို retrieve လုပ်ပေးနိုင်တဲ့ getDate function ကိုလည်းရေးထားပါတယ်။

- main( ) function ထဲမှာစစချင်း dt ကို positive ဂဏန်းတစ်ခုနဲ့ assign လုပ်ပေးခြင်းအားဖြင့် while loop ထဲကိုဝင်လို့ရပါပြီ။ dt value ကို (-5) လို့ထည့်ကြည့်ပါ။ if block statement ထဲဝင်လို့မရပါဘူး။ ဒါဆိုရင် dt value ကို (3) ရိုက်ထည့်ကြည့်ပါ။ if block ထဲကိုဝင်လို့ရပါပြီ။ const Date& rd = getDate(dt); ဆိုတဲ့ statement ကနေ local reference rd ကို returned reference getDate(dt) နဲ့ initialize လုပ်ပေးပါတယ်။
- returned reference function ထဲရောက်လာတဲ့အခါမှာ n = 3 ဖြစ်သွားပါပြီ။ ဒီတော့ birthdays[3-1] ကို return လုပ်ခိုင်းတဲ့အခါမှာ birthdays[2] = {10,5,2000} ကို main( ) ထဲက rd နဲ့ assign ပြန်လုပ်ပေးမှာပါ။ နောက်ပြီး rd member တွေကိုတစ်ခုချင်း display လုပ်ခိုင်းထားပါတယ်။ ပုံ (၄. ၄၆) ဟာ Ex4026.cpp program ကို run ပြထားတာပါပဲ။



ပုံ (၄. ၄၆)

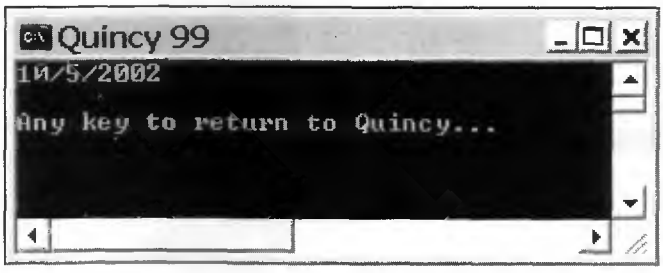
const References

ပုံ (၄. ၄၇) မှာဖော်ပြထားတဲ့ Ex4027.cpp program မှာစစချင်း struct type Date ရဲ့ instance တစ်ခုဖြစ်တဲ့ dt ကို {10, 5, 2002} လို့ initialize လုပ်ပေးထားပါတယ်။ နောက်ပြီး const reference to Date တစ်ခုဖြစ်တဲ့ drကို dt နဲ့ assign လုပ်ပေးပြီး dr member တစ်ခုချင်းကို ကွန်ပျူတာမှာ display ပြန်လုပ်ခိုင်းထားတာပါ။ Ex4027.cpp program ကို ပုံ (၄. ၄၈) မှာ run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

Ex4027.cpp

```
// Listing 4.27: const references
#include <iostream>

struct Date
{
        int  month, day, year;
} dt = {10,5,2002};

int  main()
{
        const Date&  dr = dt;

        cout << dr.month << '/'
             << dr.day << '/'
             << dr.year << endl;
        return 0;
}
```

ပုံ (၄. ၄၇)



ပုံ (၄. ၄၈)

const Reference Parameters

Ex4027.cpp program ကိုပဲ const reference parameter pass လုပ်နည်းနဲ့ပြင်ရေးမယ်ဆိုရင် ပုံ (၄. ၄၉) ကအတိုင်းရေးလို့ရပါတယ်။ program run ရင်လည်းစောစောကအဖြေပဲရမှာပါ။ ပုံ (၄. ၅၀) ကိုကြည့်ပါ

Ex4028.cpp

```
// Listing 4.28: const reference parameters

#include <iostream>
struct  Date
{
        int     month, day, year;
}  dt = {10,5,2002};

void    displayDate (const  Date&);

int     main( )
{
        const   Date&  dr = dt;

        displayDate (dr);
        return 0;
}

void    displayDate (const  Date&     x)
{
        cout << x.month << '/'
             << x.day << '/'
             << x.year << endl;
}
```

ပုံ (၄.၄၉)

Quincy 99

```
10/5/2002

Any key to return to Quincy...
```

ပုံ (၄.၅၀)