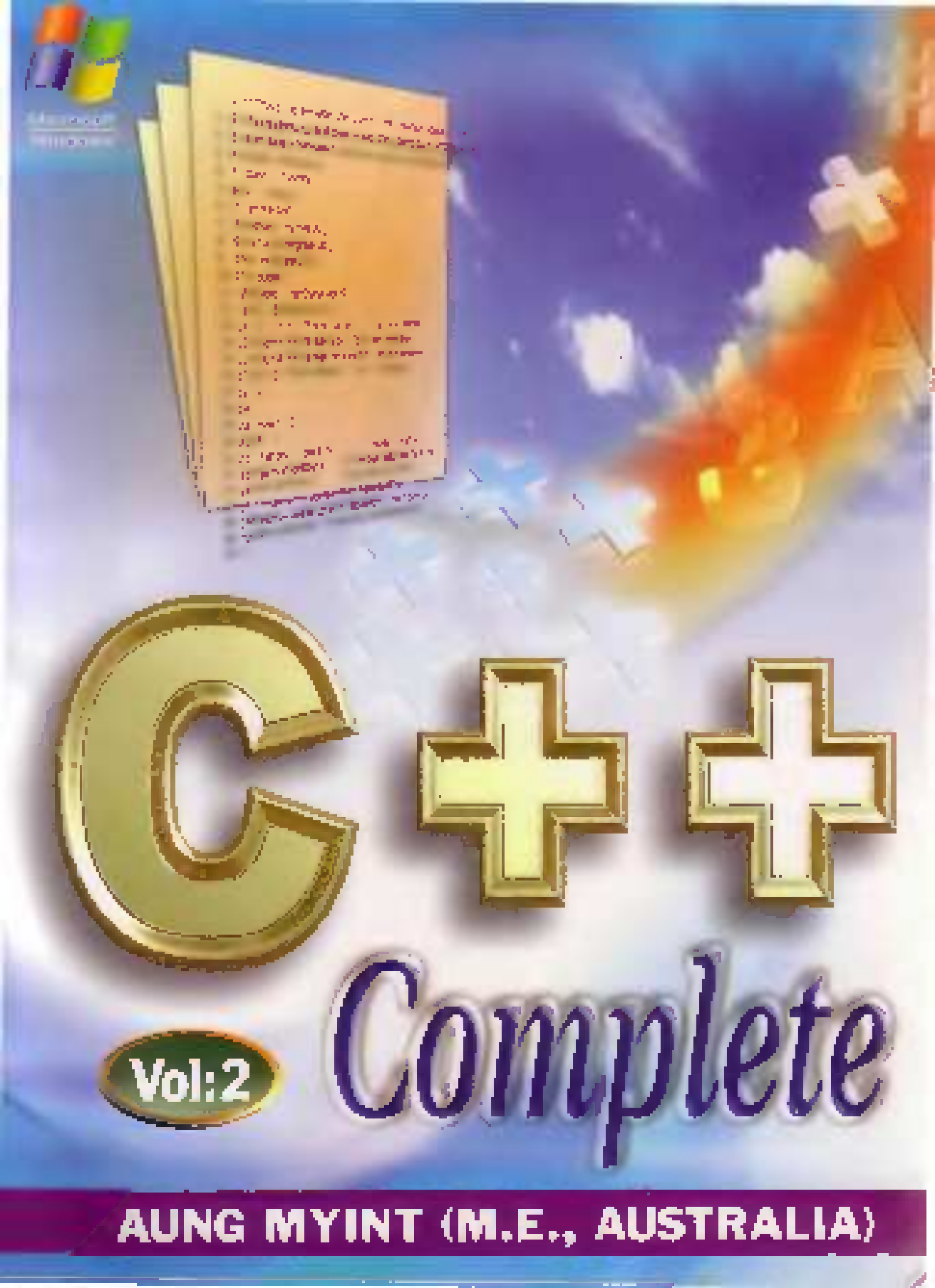


C++ Complete

Vol:2

AUNG MYINT (M.E., AUSTRALIA)

CHAPTER 5
C++ REFERENCES

Complete

CHAPTER 6
C++ CLASSES

Complete

CHAPTER 7
C++ ARRAYS

Complete

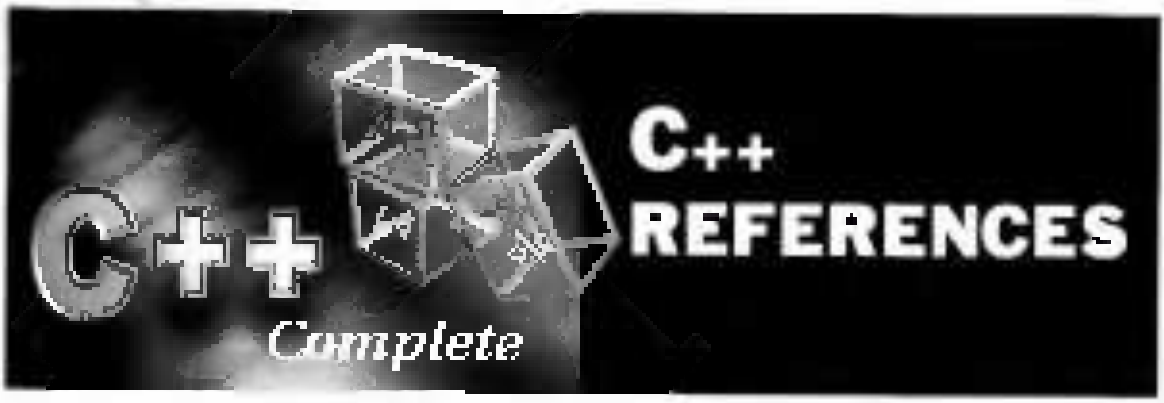
CHAPTER 8
OVERLOADED OPERATORS

CHAPTER 9
INHERITANCE

CHAPTER 10
LIBRARY FUNCTIONS

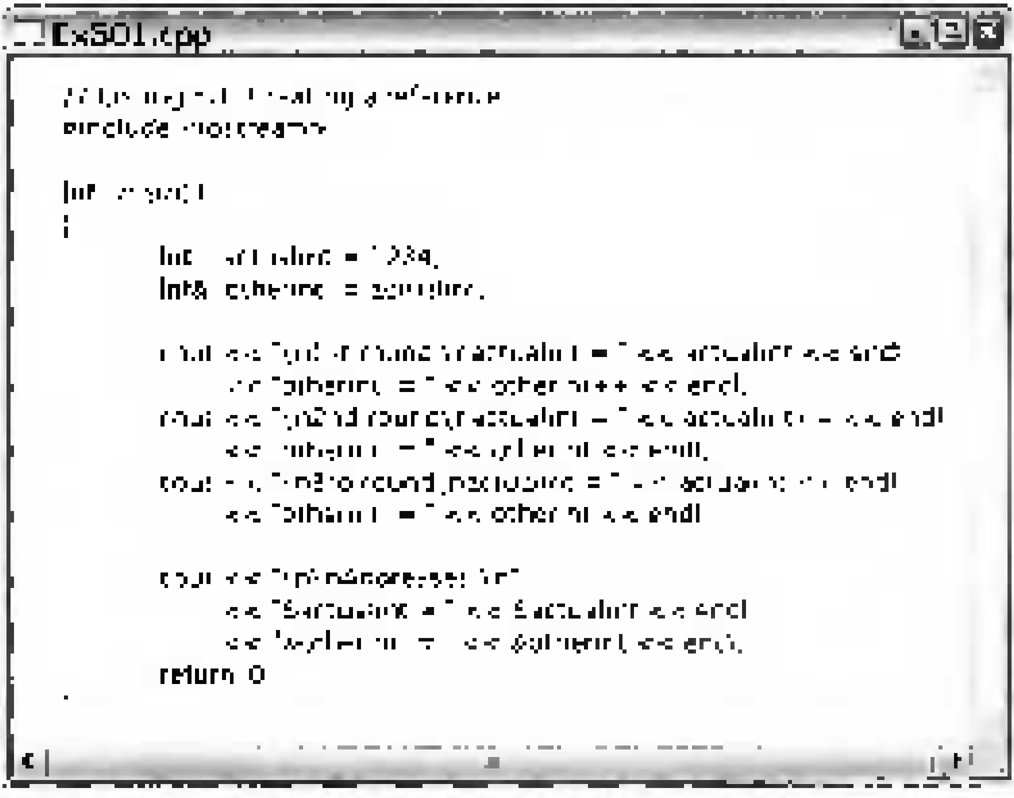
Complete

Chapter 5



၅.၁ Reference is an Alias

[illegible] Ex501.cpp program [illegible] actualInt [illegible] 1234 [illegible] initialize [illegible] variable [illegible] reference [illegible] otherInt [illegible] alias variable [illegible] define [illegible] variable [illegible] display [illegible] 1234 [illegible] reference [illegible] increment [illegible] reference [illegible] 1234 [illegible] actualInt [illegible] 1234 [illegible] actualInt [illegible] increment [illegible] otherInt [illegible]



ပုံ (၅-၁)

[illegible] Ex501.cpp program [illegible] run [illegible] actualInt [illegible] otherInt [illegible] output [illegible]

ပုံ (၂)

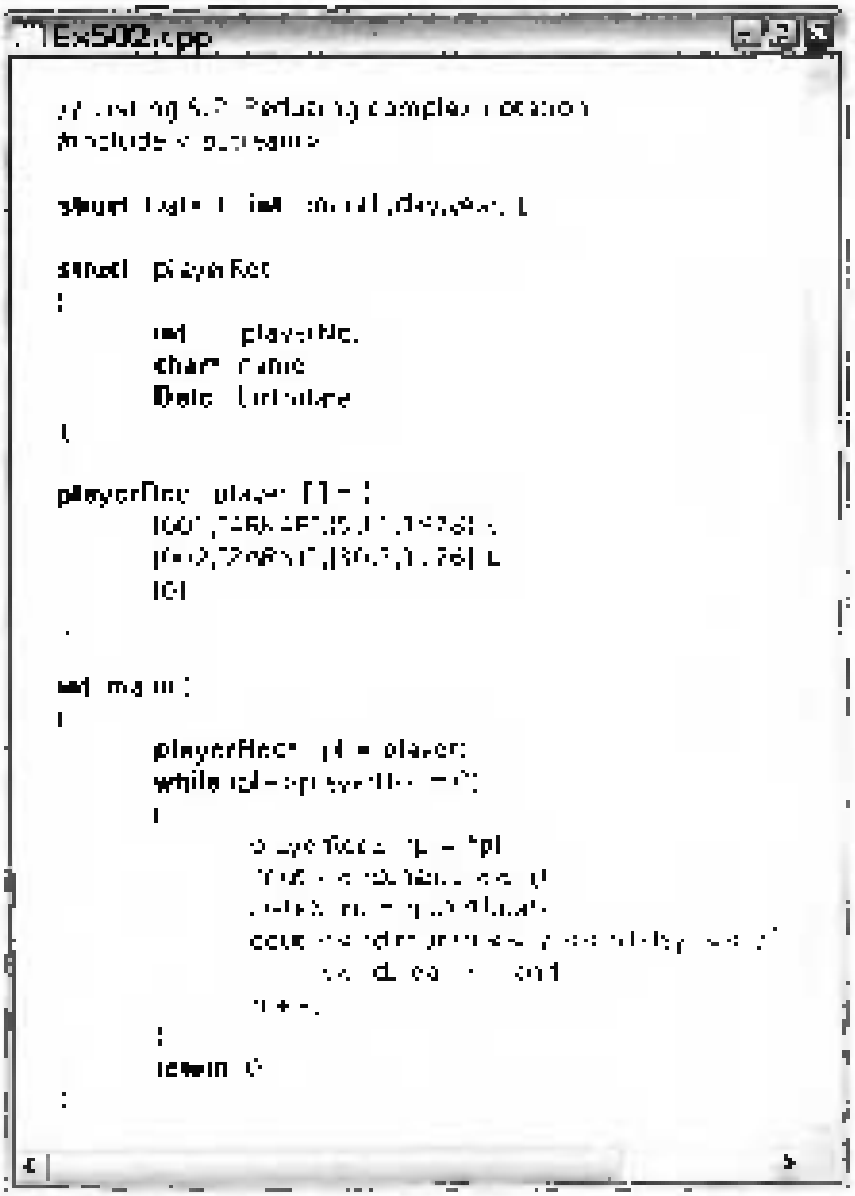
Reducing Complex Notation by References

ပုံ (၂) ကတော့ Ex502.cpp program ကဲ့သို့ structure member တွေကို အသုံးပြုသော reference သုံးပြီးရေးသားထားသော program ကိုလေ့လာနိုင်ပါတယ်။

- main() ထဲတွင်ကြေညာထားသည့် player array pointer ဖြစ်သော players[] ကို point လုပ်ပြီး players element တစ်ခုချင်းစီသည် struct type တစ်ခုစီဖြစ်သောကြောင့် while (ptr->playerNo != 0) ဆိုသည့် expression တွင် players[0] ၏ first member ကို zero စစ်ခြင်းဖြင့် end of data ဟုသိရှိသွား၍ 001 ဂျိုးဘိုင်ဒင် ဆိုသည့် while loop သို့ ဝင်ရောက်မည်ဖြစ်ပါသည်။
- plrec point လုပ်ထားသည့် players[0] ရဲ့ reference ကို rp ကို ကြေညာလိုက်သည်။ ထို့ကြောင့် rp.name ကို print လုပ်ခဲ့သောအခါ player ရဲ့အချက်အလက်တွေထဲမှ ပထမဆုံးဖြစ်သော players[0] ရဲ့ birthdate ကို print ထုတ်ခြင်းဖြစ်ပါသည်။ rp.birthdate ဆိုသည်မှာ reference ကို rd ဆိုပြီး define လုပ်ထားသော data type ဖြစ်သည့် struct Date ဖြစ်သောကြောင့် rd.month ကို print

ပုံမှာ rp.birthdate.month ဆိုတာက pl->birthdate.month ကို print ထုတ်ကြည့်ခြင်းဖြစ်ပြီး rd.month နဲ့အတူ rd.day၊ rd.year တို့ကိုလည်း print ထုတ်ပေးထားတဲ့ player ရဲ့ birthdate ဖြစ်ပါတယ်

- player အစီအစဉ်တစ်ခုတစ်ခုစီအတွက် print ထုတ်ပြသပေးဖို့ array element အရေအတွက်ကိုဖြတ်ပြီး pl ကို increment လုပ်ဖို့ while loop ကိုသုံးထားခြင်းဖြစ်ပါတယ်။ rp->playerNo



= a [illegible] while loop statement program step [illegible]

; Ex502.cpp program [illegible]

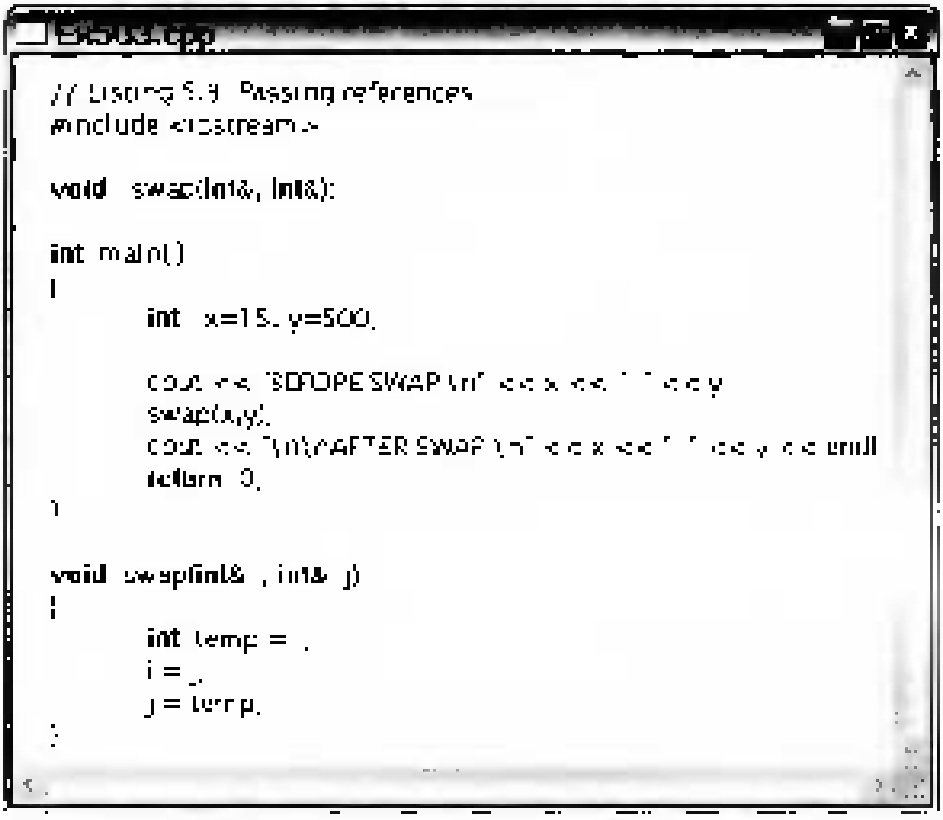
[illegible]

[illegible] Passing References

[illegible] Ex503.cpp program [illegible] main( ) function [illegible] pass [illegible] called function [illegible] reference [illegible] calling function [illegible] argument [illegible] pass [illegible] called function [illegible]

; Ex503.cpp program [illegible]

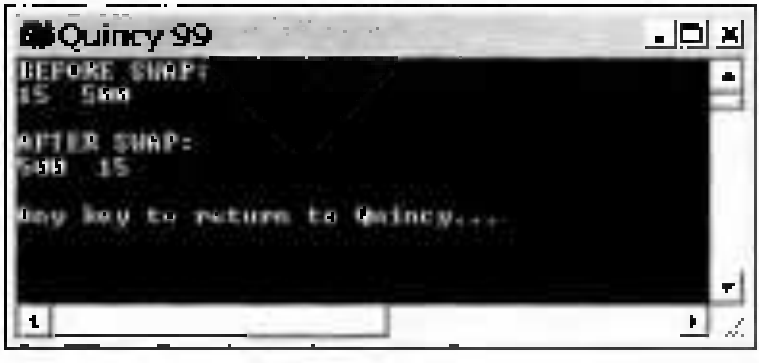
- main( ) [illegible] int x, y [illegible] 15 [illegible] display [illegible] swap( ) function [illegible] x, y [illegible] pass [illegible] swap( ) function [illegible] int& i = x – 15 int& j = y = 500 [illegible] function [illegible] i, j [illegible] x = 500, y = 15 [illegible] main( ) [illegible] x, y [illegible] display [illegible] x = 500, y = 15 [illegible]



```
// Listing 5.9  Passing references
#include <iostream.h>

void  swap(int&, int&);

int main()
{
        int  x=15, y=500;

        cout << "BEFORE SWAP\n" << x << " " << y
        swap(x,y);
        cout << "\n\nAFTER SWAP\n" << x << " " << y << endl
        return 0;
}

void swap(int& i, int& j)
{
        int temp = i;
        i = j;
        j = temp;
}
```

ပုံ (၅.၄)

ဥပမာ ပုံ (၅.၄) ဟာ Ex503.cpp program ကို run ပြုလုပ်လိုက်ရင် 15 နဲ့ 500 တို့ကို pass သွင်းပေးလိုက်တဲ့ data တစ်တွဲဟာ swap( ) function ကနေတဆင့်ပြန်ပြီး x နဲ့ y value တို့ဟာ အစားထိုးပြောင်းလဲသွားတာကို မြင်ရမှာပါ။



ပုံ (၅.၅)

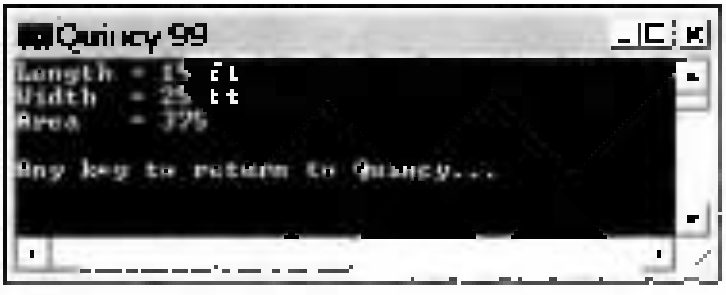
Passing Structure Data References

data structure တစ်ခုကို pointer အသုံးပြုပြီး function တစ်ခုသို့ pass လုပ်သည်နှင့် တူစွာပဲ reference အကြောင်းကြားခြင်းဖြင့်လည်း ရရှိအကောင်အထည်ဖော်နိုင်ပါသည်။ ဥပမာ Ex504 program တွင် main( ) function သို့ structure data ကိုပို့ pass လုပ်ပုံဖြစ်သည်။

```
Ex504.cpp
// Passing Structure Data by reference
#include <iostream>

struct sides
{
        int length, width;
};

void calcreal sides& s)
{
        int& l = s.length;
        int& w = s.width;

        cout << "Length = " << l << endl
             << "Width = " << w << endl
             << "Area = " << l * w << endl;
}

int main()
{
        sides rect = {15,23};

        calcreal(rect);
        return 0;
}
```

Ex504.cpp program တွင်ပြသထားသည့်အတိုင်း main( ) ထဲမှ struct type rect ကို create လုပ်ပြီး length နှင့် width တို့ကို initialize လုပ်ထားတဲ့အပြင် ပြောင်းရွှေ့ calcReal( ) function သို့ rect ကို

pass ပုံစံအသစ်အဖြစ် rect ဟာ CalArea( ) function ထဲကို address ပေး rect အဖြစ်ကူးယူသွား ပြီး function ထဲမှာပဲ ရှိနေ ဖို့ reference သုံးခြင်းဟာ define သတ်မှတ်ပေးထားသည်ဟု ဆိုလျှင် ရပြီး ဖြစ်ပါ display ပေးခိုင်းထားတဲ့နေရာတွေမှာ length , width နဲ့ area တန်ဖိုးတွေကို ပြန်ရပါမည်။ Ex504.cpp program ကို run ကြည့်လိုက်ရင် 15 နဲ့ 25 ကိုက် pass လုပ်ပေးလိုက်တဲ့ structure data တစ်ခု ဖြစ်သော rect.length နဲ့ rect.width တို့ ကို ရပြီး ပြန်ရောက်လာပါမည်။



ပုံ (၃-၁)

၅.၇ Returning References

ကျွန်တော်တို့ function ထဲကို reference အဖြစ် pass လုပ်နည်းကို Ex504.cpp program မှာ လေ့လာခဲ့ကြပြီးပါပြီ။ ဒါဆိုရင် related function တစ်ခုမှ reference အဖြစ် return ပြန်ပေးနိုင်ပုံကိုလည်း ပုံပြပြီး ရှင်းပြပေးမှာပါ။ Ex505.cpp program မှာ getLength (choice) ဆိုပြီး reference ကို main( ) function ထဲကို return ပြန်ပေးထားပါတယ်။ လေ့လာကြည့်ပါ။

Ex505.cpp program ထဲမှာ ပါဝင်တာတွေက

- struct type array တစ်ခုကြေညာပြီး shapes[ ] ဆိုပြီး program ကစပြီး define လုပ်ပေးထား တဲ့ main( ) ထဲမှာ int type choice ကို ကြေညာပြီး do loop တစ်ခုနဲ့ ပတ်ထားပါတယ်။ choice ဟာ 1 , 2 , 3 , 4 တန်ဖိုးတွေဖြစ်နေ ရင် switch statement ထဲကနေ ဆက်လုပ်ပေးမှာပါ။ ဒီထဲမှာ choice = 3 ဖြစ်နေရင် getLength( ) ကို ခေါ်ပေးမှာပါ။

Ex5b3.cpp

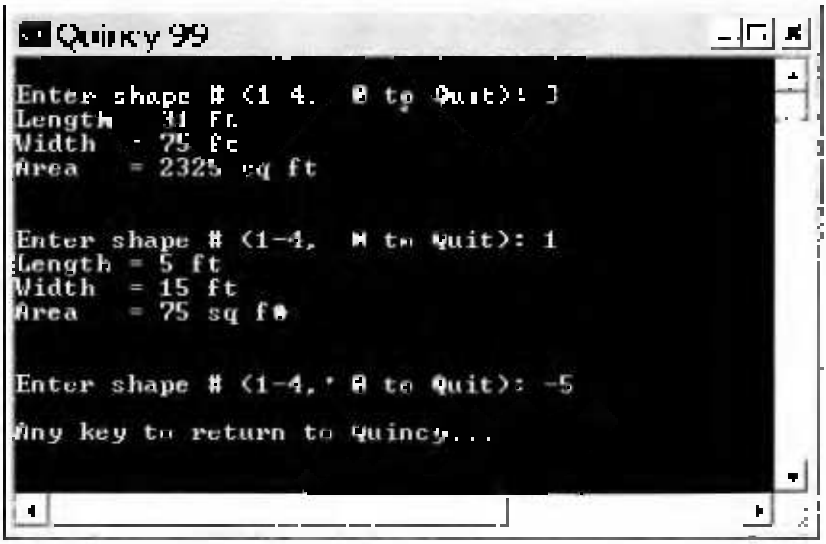
```
// Listing 5.3  Returning references
#include <iostream>
using namespace std;

struct sides
{
        int length, width;
};

sides shapes[] = {
                        {5, 10},
                        {12, 27},
                        {31, 75},
                        {45, 80},
};

sides& getLength(int i)
{
        return shapes[i-1];
}

int main()
{
        int choice;
        do {
                cout << "\n Enter shape # (1-4, 0/<0 to quit): ";
                cin >> choice;
                if (choice > 0 && choice < 5) {
                        sides& rs = getLength(choice);
                        int& rl = rs.length;
                        int& rw = rs.width;
                        cout << "Length = " << rl << "\n"
                             << "Width = " << rw << "\n"
                             << "Area = " << rl * rw << "\n\n";
                }
        } while (choice > 0 && choice < 5);
        return 0;
}
```

(5.3.cpp)

- ■ ปิดปีกกา { } block ของคำสั่ง sides& rs = getLength(choice); เป็น statement ของ getLength( ) function จะมี reference ที่ rs ที่ define ขึ้นมาใหม่

data type ကပ် sides ဆိုပါ။ အဲဒီနောက် getLength( ) function ကို call ထားပြီး choice ကို pass လုပ်ပေးတဲ့အခါမှာ called function ထဲက i = choice = 3 ဖြစ်နေမှာဖြစ်ပြီး called function က return လုပ်ပေးတာက shapes[i-1] ဖြစ်တဲ့ shapes[2] ဖြစ်ပါတယ်။

- main( ) ကိုပြန်ရောက်သွားတဲ့အခါမှာ rs = getLength(3) = shapes[2] ဖြစ်သွားပါပြီ။ ဆိုတော့ rs.length = 12 နဲ့ rs.width = 27 ဖြစ်သွားပြီဖြစ်လို့ rl = rs.length နဲ့ rw = rs.width ကို define လုပ်ပေးပြီး rl*rw ဆိုပြီး print ပုံစံမျိုးထုတ်ခိုင်းထားတဲ့ shapes[2] ရဲ့ length , width နဲ့ area တို့ကို display ပြပြီးပါပြီ။
- do loop ဆက်မှာ choice ထဲကနေကြည့်ရင် True ဖြစ်နေသေးတဲ့အတွက် do loop ထဲကိုပြန်ပြေးပါဦးမယ်။ ဒီတစ်ခါ choice = 1 ဖြစ်နေသွားတဲ့အတွက် shapes[0] အတွက် အချက်အလက်တွေကိုပြပေးပါလိမ့်မယ်။ choice = -5 ဖြစ်သွားတဲ့အတွက် do loop ထဲကိုပြန်ရောက်လာပါပြီ။ choice = -5 ဖြစ်တဲ့အတွက် do loop ထဲကိုဆက်မသွားတော့ဘဲ program exit ဖြစ်သွားမှာပါ။ ဒုတိယ (ဥပမာ) ကိုကြည့်ပါ။

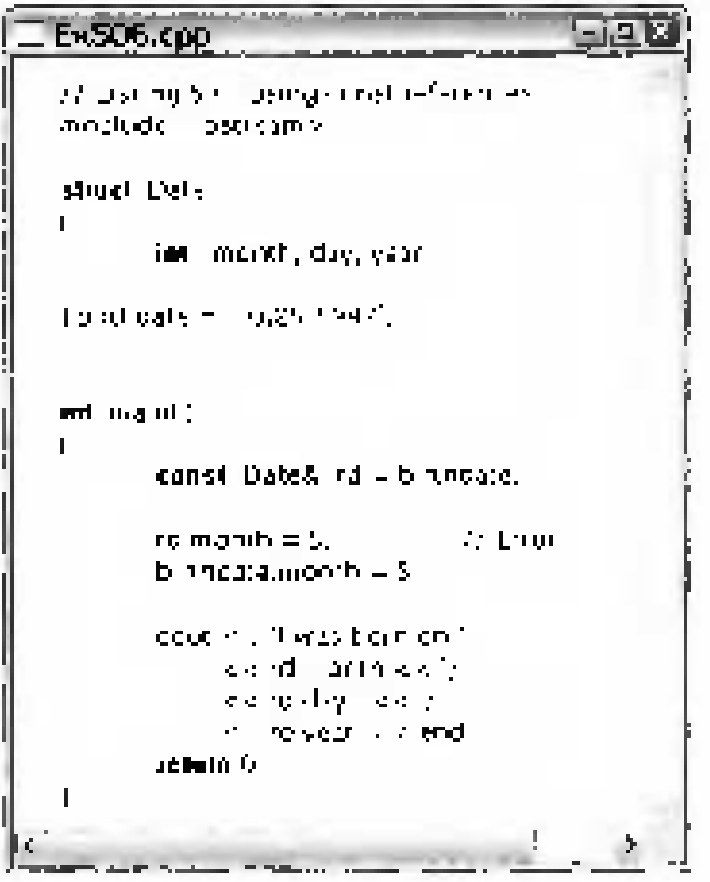


ဒုတိယ (ဥပမာ)

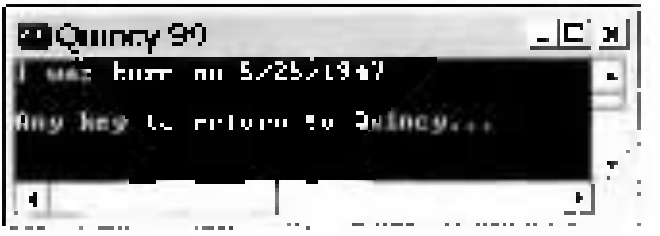
၅.၄ Using const References

ယခင် const ဆိုတဲ့ specifier ကို reference အဖြစ်သုံးသည့်အခါတွင်လဲ main( ) program အတွင်းက object ကို reference အနေဖြင့်ခေါ်ယူသည့်အခါ ထို object ကိုပုံစံတစ်မျိုးပြောင်းလဲခြင်းမရှိစေရန် ဒုတိယ (ဥပမာ) အစီအစဉ်တွင်

Ex506.cpp program မှာ rd.month = 5; ဆိုတဲ့ statement ကြောင့် error ဖြစ်ပေါ်ပါတယ်။ birthdate object ကိုကြေညာတဲ့အခါမှာ ထည့်သွင်းပေးလိုက်တဲ့ const specifier ကြောင့်ပါ။



ဥပမာ rd.month = 5; statement ကိုဖျက်ပြီး Ex506.cpp ကို run လိုက်ရင် month ကိုပြောင်းလို့ မရတဲ့အတွက် ပုံအောက်ပါအတိုင်း ပြသပေးပါလိမ့်မယ်။

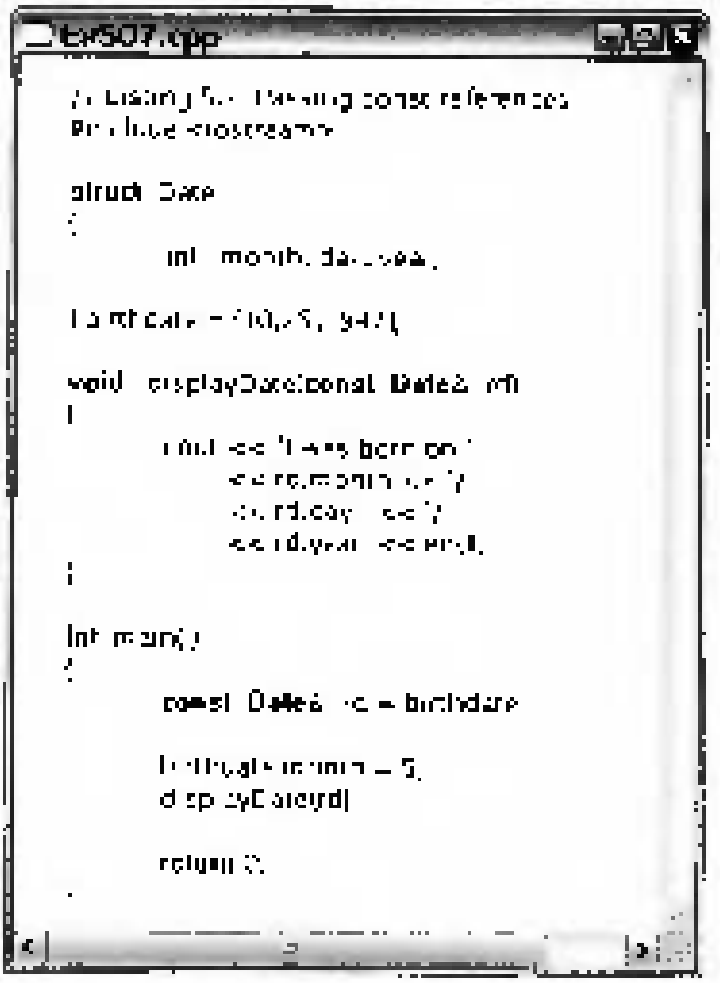


[illegible] const [illegible] const object [illegible] non-constant reference [illegible] initialize [illegible] program [illegible] run [illegible] run [illegible] Date birthdate = {10, 25, 1947}; [illegible] const [illegible] Date& rd = birthdate [illegible] initialization [illegible] non constant reference [illegible] const [illegible] program run [illegible]

```
// Listing 5.5 Using const references
#include <iostream>

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

int main()
{
        const Date birthdate = {10,25,1947};
        Date& rd = birthdate; // Error

        cout << "I was born on "
             << rd.month << '/'
             << rd.day << '/'
             << rd.year << endl;
        return 0;
```

[illegible]

[illegible] Ex507.cpp program [illegible] const reference [illegible] parameter [illegible] pass [illegible] program [illegible] run [illegible] main( ) [illegible] birthdate.month = 5; [illegible] statement [illegible] rd.month = 5; [illegible] statement [illegible] displayDate( ) function body [illegible] run [illegible] program run [illegible]



9.0 C++ Preprocessor

C++ preprocessor ဆိုတာကတော့ ကျနော်တို့ရေးသားမယ့် C++ program ရဲ့ source code ကနေ compile မလုပ်မီမှာ preprocessor အနေနဲ့ကြိုတင် ဆောင်ရွက်ပေးတဲ့ translation ပြုပြင်ပေးခြင်းတွေ၊ program comment တွေ၊ white space တွေကိုဖယ်ရှားတဲ့ compiler ဆောင်ရွက်ပေးခြင်းတွေ၊ macro definition တွေကိုရှာပြီး translate ပြုပြင်ပေးတာ၊ header file တွေကို source code ထဲကိုထည့်ပေးတဲ့ source code ပြောင်းလဲမှုတွေကို compiler က compile မလုပ်ခင်မှာ

preprocessor ရဲ့ output သည် compiler ရဲ့ input ဖြစ်ပါသည်။ ဒီအခန်းတွင်မှ preprocessing directive များကို ရေးသွားမှာဖြစ်ပါသောကြောင့် # (pound) သင်္ကေတနှင့်အတူ code အတွဲ preprocessing directive များကိုသာ ဖော်ပြထားပါသည်။ အောက်ပါဇယား (၇-၁) ကိုကြည့်ပါ။

ဇယား (၇-၁) Preprocessing Directives

Directive	Meaning
#	Null directive, no action
#include	Include a source code file
#define	Define a macro
#undef	Remove the definition of a macro
#if	Compile code if condition is true
#ifdef	Compile code if macro is defined
#ifndef	Compile code if macro is not defined
#else	Compile code if previous #if condition is not true
#elif	Compile code if previous #if condition is not true and current condition is true
#endif	Terminate #if ... #else conditional block
#error	Stop compiling and replay error message

Including Files: #include

C++ program တစ်ခုမှာ header ဖိုင်များကို ပုံမှန် (၂) နည်းအတိုင်းခေါ်ယူနိုင်ပါသည်။

```
#include <iostream>
#include "menu.h"
```

ပထမနည်းမှာ name ကို <(less than) နှင့် >(greater than) အကြားတွင်ထည့်ပေးရပါသည်။

compiler system ရဲ့ header ဖိုင်များထဲတွင်ပါဝင်သော iostream.h ဟူသည့် preprocessor လမ်းညွှန်ချက်သည် ထိုဖိုင်၏ name ကို double quotation marks (" ") များဖြင့်ဝိုင်းထားသောကြောင့် ဖိုင်ကို compile ပြုလုပ်သည့် application source code များတွင်သာရှိသော header ဖိုင်များကို include ပြုလုပ်ပေးရန် compiler system ကိုညွှန်ကြားလိုက်ခြင်းဖြစ်သည်။

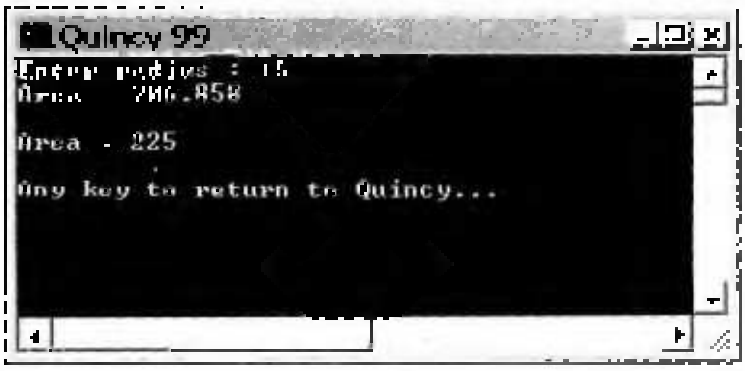
Macros: #define and #undef

#define preprocessing directive ဖြင့်ကြေညာသည့် macro definition များကိုရည်ညွှန်းရန် #define MAXI 100 ဖြစ်လျှင် #define Square(x) x*x ဖြင့်ကြေညာထားသည်။ #define directive definition ကို ပြန်လည်ပျက်ဖျက်ရန် #undef MAXI ဖြင့်ရေးသားခြင်းဖြင့် macro declaration ပြုလုပ်ထားမှုကို ပျက်ဖျက်သည်။ #define directive ကျွန်တော်တို့ကြည့်ကြပါစို့ ရေးကြည့်ပါမည်။

ပုံ ၅-၅ ကို

၂.၂ Ex508.cpp program ကို trace လုပ်ခြင်းဥပမာစဉ်းစားခြင်း

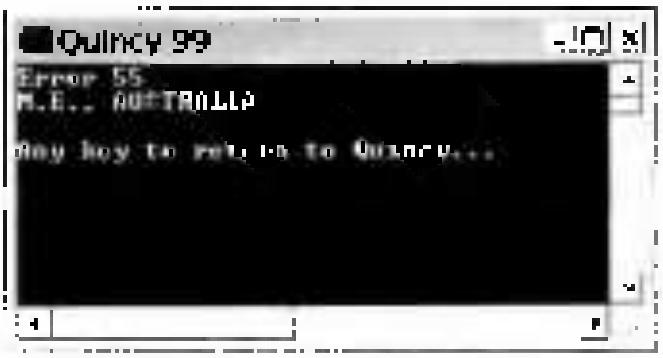
- program ထိပ်ဆုံးမှာ macro (2) ခုကို define လုပ်ထားပါသည်။ PI နဲ့ AREA(r) ပါ။ main( ) function ထဲမှာ radius အတွက် data ကိုစကင်းလုပ်ထားခါတွင် 15 ကိုပြောင်းထည့်ပေးပြီးပေးမည်ဆိုပါစို့။ ဒါဆိုရင် AREA(radius) ကနေ PI*r*r = 3.14159*15*15 = 706.858 ကိုရလာမှာဖြစ်သည်။
- #undef PI; ဆိုတဲ့ statement ကတော့ PI နဲ့ macro definition ကို cancel လုပ်ပစ်လိုက်ပါ။ ပြန်ကြေငြာမဲ့ PI ကို new value တစ်ခုနဲ့ assign လုပ်ပေးရမှာဖြစ်သည်။ ဥပမာ။ PI = 1 ဆိုပါစို့။ ဒါဆိုရင် PI*r*r = 1*15*15 = 225 ဖြစ် မောက်သစ်မှုယွင်းပေါ်ပေါ်မှာပါ။
- ပုံ (၂.၁၀) အဆိုပါ Ex508.cpp program ကို ပေးစောလေးပြောင်းထဲ data ပေးတဲ့အခါ run ပြုတာကတော့ဖြစ်ပါသည်။



ပုံ (၂.၁၀)

Stringizing and Concatenation Operators

ဒီ macro definition တစ်ခုမှာ stringizing operator (#) တစ်ခု ပြောင်း #r ကိုလုပ်ခြင်းဆောင်ရွက်ပေးသည်ဆိုရင် argument ကို string အနေနဲ့ပြောင်းပေးမှာပါ။ argurment တွေးကိုတစ်ခုချင်းစီ concatenation operator (##) ကိုသုံးချိတ်ဆက်ပေးပါမည်။ ပုံ (၂.၁၀) မှာ ဒီ operator (2) ခုအသုံးပြုပုံကို program ရေးသွားပါအယ်။ ဆောင်ရွက်ကြည့်ပါ။



Compile-time Conditionals Directives

... : directive ... program ... code statement ... compile ... #if ... #endif ... #if ... #else ... #endif ... #if ... #elif ... #else ... #endif ... directive ... compile-time condition ... control ... exBillD.cpp program ... #if ... #endif ... #if defined ... #endif ... conditional directive (2) ...

... exBillD.cpp program ... trace ...

- program ... macro (4) ... #define directive ... #define DEBUG ... #define CHECK ... define ...
- main() ... DEBUG ... #if ... #if (true) ... main() ... data ... #endif ... #if block ...
- ... #if defined CHECK ... true ... #if defined block ... #endif ... macro ... #else ... program ...

Ex5010.cpp

```
// Using #if, #elif, #else, #endif, and compile-time constants
#include <iostream>

#define  PI          3.14159
#define  AREA(r)     PI*r*r
#define  DEBUG       1
#define  CHECK

int main()
{

        float radius;

        #if DEBUG
                cout << "Enter radius: ";
                cin >> radius;
                cout << "Area = " << AREA(radius) << endl;
        #endif

        #if defined CHECK
                #undef PI
                float PI = 1;
                cout << "\nArea = " << AREA(radius) << endl;
        #endif
        return 0;
}
```

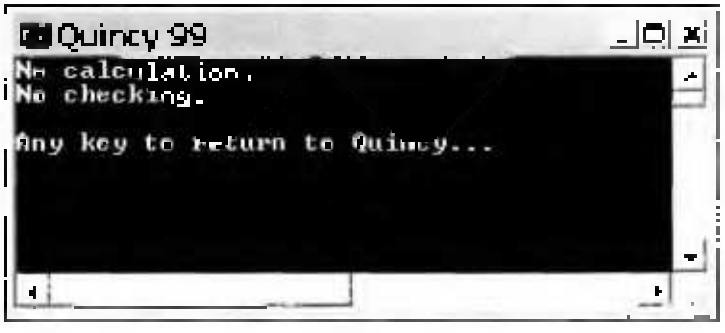
ပုံ (၅.၀၆)

၅.၈ အထက်ပါ Ex5010.cpp program ကို ပုံ (၅.၀၆) ပုံအတိုင်းရိုက်ထည့်ပြီး run ပေးခဲ့ရင် #if DEBUG ဟာ false ဖြစ်သွားတဲ့အတွက် ဘဲ့ #if block အပိုင်းတစ်ခုလုံးဟာ ဖယ်ရှားခံသွားရပြီး #elif !(defined CHECK) ကလည်း false ပြန်ဖြစ်သွား ဒီတော့ ဖယ်ထုတ်ခံရတာကြောင့် #else အောက်ပိုင်း စုံစမ်းတဲ့ ပြဿနာတွေ No calculation. No checking. စိတ်ပျော်စရာကြီး (2) ခုကြောင့် တိုးတိုးလေး ပါဝင်ဆိုပေတော့

၅.၉ ပုံ (၅.၀၇) မှာ Ex5011.cpp program ကို run ပြထားပါတယ်

Ex5011.cpp

```
// Listing 5.11: Compile-time conditionals
#include <iostream>

#define PI          3.1415293
#define AREA(r)     PI*r*r
#define DEBUG       0
#define CHECK

int main()
{

        float radius;

        #if DEBUG
                cout << "Enter radius: ";
                cin >> radius;
                cout << "Area     " << AREA(radius) << endl;
        #elif !defined CHECK
                #undef PI
                float PI = 1;
                cout << "\nArea = " << AREA(radius) << endl;
        #else
                cout << "No calculation.\nNo checking.\n";
        #endif
        return 0;
}
```

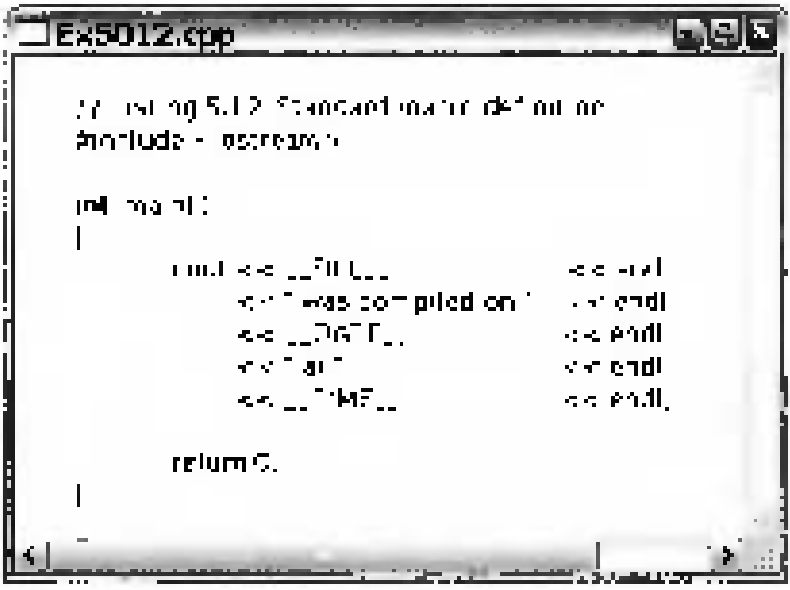


Standard Defined Macro Names

compiler system သည် define ပြုလုပ်ထားသည့် standard macro များကို အသုံး [illegible] ကို ကြည့်ရှုခြင်းဖြင့် သိရှိနိုင်ပါသည်။ ဥပမာအနေဖြင့် ဤ [illegible] ကို Ex50.cpp program တွင် အသုံးပြုထားပါသည်။

[illegible] Standard Defined Macro Names

Macro Name	Meaning
__LINE__	The line number of the current source code line
__FILE__	The name of of the current source code line
__DATE__	The date the source code was compiled ("mmm dd yy")
__TIME__	The time the source code was compiled ("hh:mm:ss")
__STDC__	Identified but not defined by the standard
__cplusplus__	Defined as the constant long integer value 199711L when the source code being compiled is C++ code, but not defined otherwise.
__STDC	is typically defined to specify a program should be compiled by using Standard C++ conventions only and no compiler-specific language extensions.
__cplusplus__	is typically used in header files that can be shared between C and C++ development environments and need to provide different source code for the two language processors.

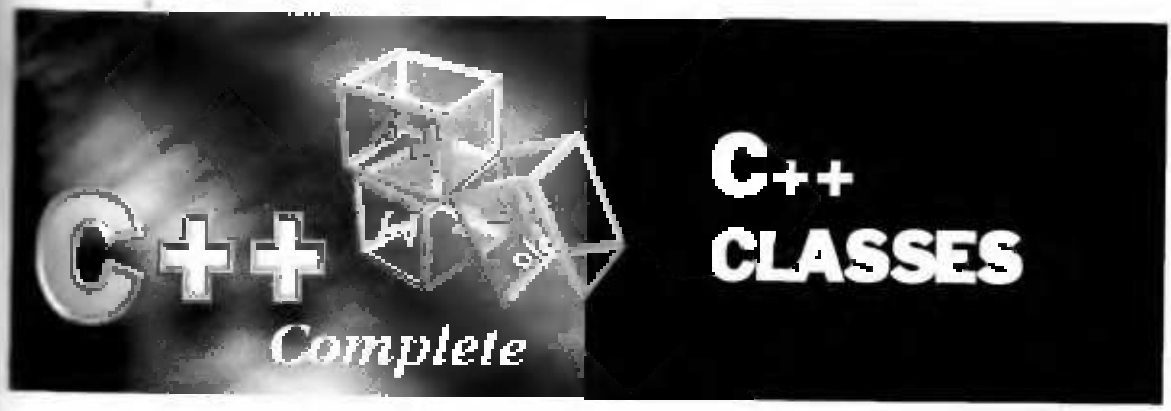


ပုံ (၅-၁၂)

၂။ Ex5012.cpp program ကို run ပြီးသည်အခါ ပုံ (၅-၁၃) ကဲ့သို့ ရလဒ်များပေါ်ပြီး ပြတင်းပေါက်သည် ခဏတာ ပေါ်လာပြီးနောက် ပျောက်ကွယ်သွားတတ်ကြောင်း

ပုံ (၅-၁၃)

Chapter 6



class ဆိုတာ C++ program တစ်ခုလုံးအတွက်လုပ်ဆောင်ချက်အချို့ကိုပြန်လည်အသုံးပြုနိုင်ဖို့ ပြင်ဆင်ပေးထားတဲ့ code ပုံစံတစ်ခုပါပဲ။ C++ မှာ အများအသုံးများတဲ့ variable တွေ function တွေကိုရေးပြီးတဲ့အခါ class တစ်ခုအဖြစ်ပြောင်းလဲ ရေးသားလေ့ရှိကြပါတယ်။ class လိုခေါ်ရခြင်းရဲ့အကြောင်းက class ဟာအများသုံးနိုင်တဲ့ object တွေအတွက် (instance) တွေကို create လုပ်နိုင်ဖို့ လုပ်ပေးနိုင်ခြင်းမျိုးပေး class ကစ်ခဲ့ပြောင်းလဲခြင်းအတွက် class ဆိုတဲ့ keyword ကိုအသုံးပြုရပါတယ်။ အောက်ပါ class declaration တစ်ခုရဲ့ပုံစံကိုကြည့်ပါ။

```
class  className
{
        private:
                // private member functions and variables
        public:
                // public member functions and variables
};
```

ဒီအရာတို့ကို private က default ဖြစ်နေပေမယ့် class declaration မှာတော့မရေးရပါဘူး။ class မှာပြုလုပ်ထားတဲ့ function နဲ့ variable တွေကို member လုပ်ခြင်းလုပ်ခံရပါတယ်။ private

ဒေတာနဲ့ပတ်သက်တဲ့ member တွေကို class member function တွေက access လုပ်ခွင့်ပေးနိုင်ပါတယ်။ class တစ်ခုပေါ်က တစ်ခြား function တွေကနေ call ခေါ်လို့ရတဲ့ function တွေဟာ public member တွေဖြစ်ပြီး program ပုံပေါ်ပေါ်တန်ရာမှာ function တွေကို call မခေါ်ခွင့်ပေးချင်ရင်တော့ private member တွေကို ထူးခြားတဲ့ကိုယ်ပိုင်ရည်ရွယ်ချက်နဲ့ public member တွေကနေတဆင့် call ခေါ်ခိုင်းရပါတယ်။ public member တွေကိုတော့ public ဆိုတဲ့ keyword ကို အသုံးပြုပေါ်ပေးပြီး public keyword နောက်မှာ colon (:) တစ်ခုထည့်ပေးဖို့လိုပါတယ်။ class တစ်ခုကို create လုပ်ခဲ့တဲ့ အကောင်အထည်ဖော်ပုံ program ကိုလေ့လာကြည့်ရအောင်ပါ။

```
// This program calculates the area of a circle of the known radius
#include <iostream>

const  float  PI =3.141593;
float    radius =  5.5;
float    calArea( );

int  main( )
{
      cout << "Radius = " << radius << " in\n"
           << "Area of circle = " << calArea( ) << " sq.in\n";
      return 0;
}

float calArea( )
{
      return   PI*radius*radius;
}
```

ဒီအတိုင်း program မှာဆိုရင် radius ကို global variable အနေနဲ့ကြေငြာလိုက်တာတွေကြောင့် radius ဟာ program ထဲက function အကုန်လုံးကတော့ public ဖြစ်နေပါတယ်။ ထို့ကြောင့် calArea( ) function ဟာ လည်း public ဖြစ်နေသင့်တယ်။ ဒီ program မှာ private ဖြစ်တဲ့ variable သို့ function တို့ပဲဝေပါပါတယ်။ ဒီ program ကို class ပုံစံနဲ့ရေးသားပြီးပြောင်းလဲကြည့်ပါမယ်။ ပုံ (၆.၂) မှာပေါ်ပြထားတဲ့ Ex601.cpp program ကိုဖတ်ကြည့်ပါ။

6.5 Create A Simple Class

Ex601.cpp program ကိုရေးကြည့်ရအောင်

- class type ကဲ့သို့ဖြစ်ကြောင်း private member ကိုဝင်ရောက်သုံးခွင့်ပေးသော public member variable radius နှင့် member function calcArea() ပါဝင်သည့် circle ခေါ်သော class တစ်ခုကို ဖန်တီးပြီး main() function ကိုအသုံးပြုကာ circle class ၏ object တစ်ခုကို ဖန်တီးပြီး radius ကို assign ပေးကာ area ကို ရှာဖွေပုံကို ဖော်ပြထားသည်။

Ex601.cpp

```
// Listing 6.1  Creating a simple class
#include <iostream>

const  float  PI=3.14159;

class  circle
{
    public
    :       float radius;       // public member variable
            float calcArea();   // public member function
};

float circle::calcArea()
{
        return PI*radius*radius;
}

int main()
{
        circle c1;

        float objectArea = 0.0;
        cout << "Please enter the radius of the circle: ";
            cin >> c1.radius;
            objectArea = c1.calcArea();
        cout << "The area of the circle is " << objectArea << endl;
        return 0;
}
```

class instance တစ်ခုကို create လုပ်ပုံနဲ့ main( ) function ထဲမှာ circle class instance တစ်ခုဖြစ်တဲ့ obj ကို create လုပ်ခဲ့တာမှာ obj ရဲ့ radius = 5.5 ကို သတ်မှတ်ပေးခဲ့ဖို့ class declaration ထဲမှာ radius = 5.5 ပြင်လိုက်ပါမယ်။

- obj.calArea( ) ဆိုတာ class declaration ထဲက member function ကို call ခေါ်လိုက်တာပေမယ့် ဒီတစ်ခါ float circle::calArea( ) ဆိုတဲ့ class member function တစ်ခုလုံးကို member function header တစ်ကြောင်းတည်းဖြစ်အောင် function return type ရဲ့နောက်မှာ class name နဲ့ scope resolution operator (::) နဲ့ ဆက်ပြီးတော့ function name နဲ့ parameter list ထဲက parameter list အတွက် argument တွေကို ပြင်ပြီး ရေးခဲ့ပေးပါ။ calArea( ) function ထဲမှာ PI*radius*radius = 3.141593* 5.5 *5.5 = 95.0332 ဖြစ်သွားပြီးတော့ main( ) ကို return ပြန်ပေးလိုက်ပါမယ်။
- main( ) function ကိုပြန်ရောက်သွားတဲ့အခါ obj.calArea( ) = 95.0332 ဖြစ်နေတာကို ပြန်ရောက်သွားမှာပါ။ Ex601.cpp program ကို run ပြီး ရလဒ်ကို ကြည့်ပြီး ပုံမှန် ကျနေတာကို သတ်မှတ်ပေးပါ။

ပုံ ၆-၁

Radius is Made Private

ဒစ်ကယ်ခဲ့တဲ့ Ex601.cpp program မှာ radius member ကို public ကနေ private သတ်မှတ်ခဲ့ရင်း program ကို ပြင်ပြီး ပုံမှန်အလုပ်လုပ်အောင် ပြင်ဆင်ရေးသားပါမယ်။ Ex602.cpp program ကိုနည်းပြင်ရေးသားကြည့်ပါ။

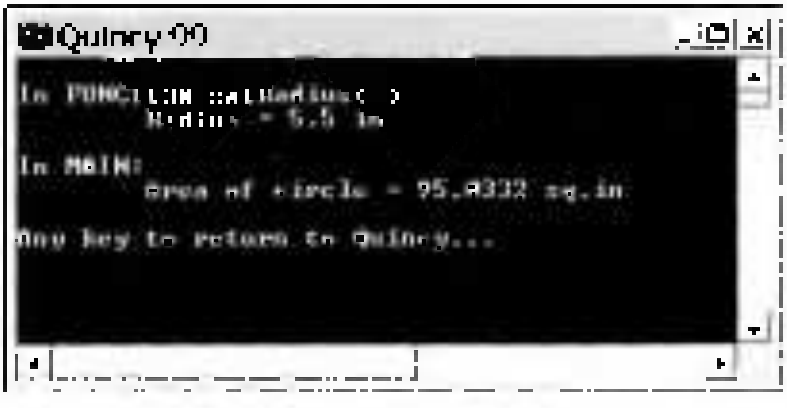
Ex02.cpp

```
// Using the private member variable
#include <iostream.h>
const float PI=3.141593

class circle

        float radius;   // private member variable
  public
        void setRadius(float r); // public member functions
        float calArea();
};

void circle::setRadius(float r)
{
        radius = r;
        cout << "\nIn FUNCTION setRadius()\n"
             << "\tRadius = " << radius << endl;
}

float circle::calArea()
{
        return PI*radius*radius;
}

int main()
{
        circle cObj;

        cObj.setRadius(5.5);    // call public function
        cout << "\nIn Main()\n"
             << "\tArea of circle = " << cObj.calArea()
             << " sq. m\n";
        return 0;
}
```

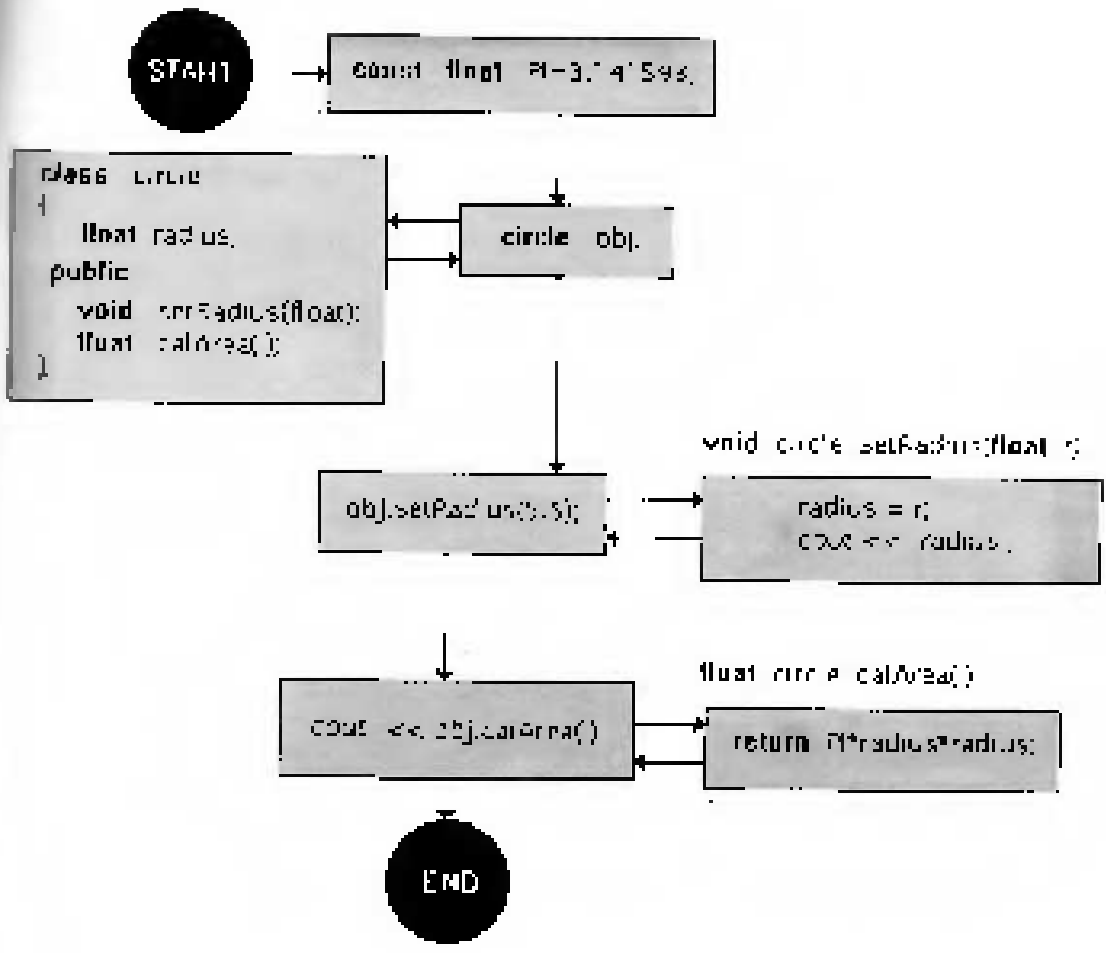

- 위에서 const float PI=3.141593 은 global value 로서 initialize 되었고 circle class instance 인 cObj 에 의해 public function 인 setRadius(5.5); 에 의해 argument 로 5.5

::ii pass လုပ်မယ့် value 5.5 ကို public member function မှာ r = 5.5 အဖြစ်ပြောင်း
ပြီး radius ကို class declaration မှာ private ဒေတာအဖြစ် main ( ) function
မှာရှိ radius အတွက် data ကိုဖို့ class ထဲကိုရောက်သွားမှာ setRadius(float r)
function ကို radius = r; ဆိုတဲ့ assignment statement အတိုင်းလုပ်ဆောင်မယ်။
class ထဲရောက်သွား public member မှ private member ကို access လုပ်ခွင့်ရတော့
သည်ကို radius = 5.5 ကို ကြိုးကြေးသိမ်းဆည်းပြီးတဲ့ member function တိုင်း radius
ကို display ပြသပါမယ်။

- ဒေတာပြန်ယူသုံးတဲ့ main ( ) ကိုပြန်ရောက်တဲ့အခါ class ထဲက public member function
တစ်ခုဖြစ်တဲ့ calArea ( ) ကနေမှ radius ဒေတာကို ဆွဲယူပြီး ဒေတာတွေ main ( ) ထဲရောက်
သွားပြီး calArea ( ) function call ဆိုတဲ့အခါ area = PI*radius*radius ကိုဆိုင်ရာ function
ကနေတိုက်ရိုက်ဆွဲယူ main ( ) ကိုပြန်ပြီးတော့မှ objArea ( ) ဆိုတဲ့ object ကနေ
objArea ( ) ပေါ်ကနေ display ပြသပြီးတဲ့ circle area ကိုဖြစ်ပါတယ်။
- Ex602.cpp program ကို run ပြီးရင် ပုံ (5.4) ကဲ့သို့ပြသသွားမှာဖြစ်ပြီး main ( )
ထဲက pass ပြုလုပ်ထားတဲ့ radius ဒေတာ data ကို setRadius( ) function ကနေ
display ပြသပြီးတဲ့ area တန်ဖိုးကို main ( ) ကနေမှ display ပြသပေးပါတယ်။
- Ex602.cpp program ကို အသေးစိတ်လေ့လာသွားဖို့ trace လုပ်ပြုလုပ်ထားတဲ့ flow
diagram တစ်ခုကိုရှင်းလင်းတင်ပြထားပါတယ်။ ပုံ (5.5) ကိုကြည့်ပြီး class အတွက်ကို ပေး
ထားတဲ့အတွက် အမှန်တကယ် flow diagram ဖြစ်ပေါ်လာပုံကို



ပုံ (5.4)



ပုံ (6-1)

6.1 Using the Inline Function

ယခင်ကပြခဲ့သော setRadius( ) နှင့် calArea( ) function တွင်ပါသော class declaration ထဲတွင် ပုံသေသတ်မှတ်ထားသည့် program ဖော်ပြချက်များ ဖြစ်ကြောင်း inline function declaration ဖြစ်ကြောင်း ပုံ (6-1) ပြထားသော Ex603.cpp program ဟာ inline function ဖြစ်ပြီး Ex602.cpp program ကိုပြောင်းလဲထားခြင်းဖြစ်ကြောင်း

Ex603.cpp

```
// Listing 6.3  Using more inline functions
#include <iostream>
const float PI=3.14159;

class circle
{
        float radius;
    public
        void setRadius(float r)
                {       radius = r;
                        cout << "\n\n FUNCTION setRadius() \n"
                             << "\tRadius    << radius << " in\n\n";
                }
        float calArea()
                { return PI*radius*radius }
};

int main()
{
        circle obj;

        obj.setRadius(5.5);     // call public function
        cout << "In MAIN \n"
             << "\tArea of circle = " << obj.calArea()
             << " sq. in\n";
        return 0;
}
```

ပုံ (၆.၅)

- ex603.cpp program ကို run ပြီးနိုင်ရန် Ex602.cpp program ကို run သည်ထိ အဆင့်အတူတူပါပဲ

More on the Inline Function

in inc function ပထဝီဝင်ကို ပြန်ပြင်ပြီးလျှင်းသည်ဖြစ်ပါသည် ထောင် program အဆန်းကြယ်ပြီးပြီး ပုံ (၆.၄) ဖြစ်ပါသည်။ Ex604.cpp program ဟာ cube အဆန် volume ကိုတွက်ပေးပါတယ်။

Ex604.cpp

```
// Listing 5.4  More on the inline function
#include <iostream>

class  box
{
        float  height, width, depth;
    public:
        void  setBox(float h, float w, float d)
        {
                height = h;
                width  = w;
                depth  = d;
                cout << "Height = " << height << "\n"
                     << "Width  = " << width  << "\n"
                     << "Depth  = " << depth  << "\n\n";
                }

                float  calVol()
                {
                        return  height*width*depth;
                }
};

int  main()
{
        box  obj;

        obj.setBox(5.5, 7.5, 10.5);
        cout << "\nVolume = " << obj.calVol() << " cu.ft\n";
        return 0;
}
```

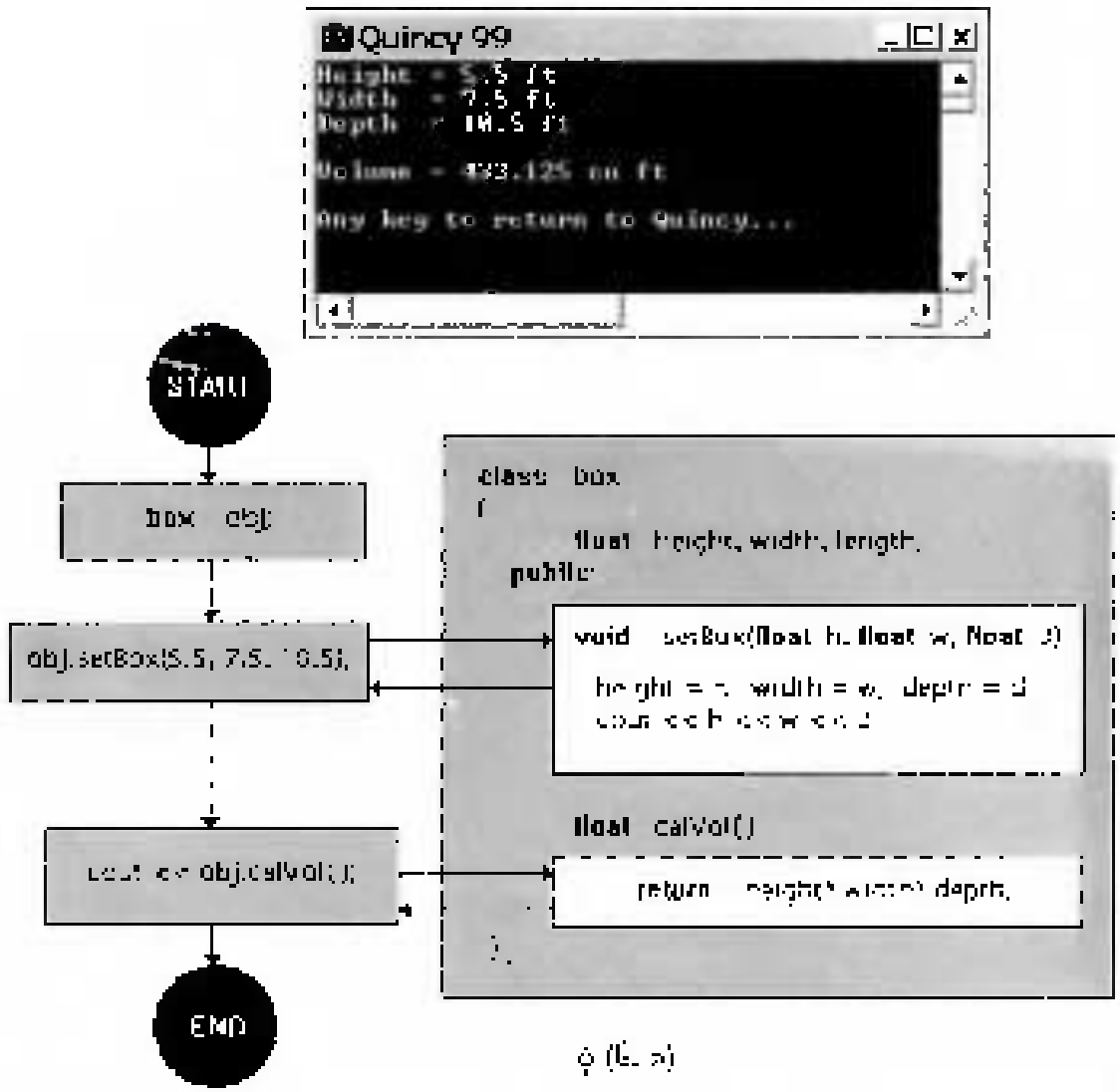
ပုံ (၆-၃)

Ex604.cpp program ကိုစမ်းသပ်ကြည့်မယ်ဆိုရင်

- main( ) ထဲမှာ box class instance တစ်ခုဖြစ်တဲ့ obj ကို create လုပ်ပြီး setBox( ) function ကို call သင်္ကေတ inline function ထဲကို constant value (3) နဲ့ pass လုပ်ပေးမယ်။ 3 data တွေဟာ inline function ထဲမှာ height = h = 5.5, width = w =

7.5 နဲ့ depth = d = 10.5 အနေနဲ့ assign သွားသေးတယ်။ ဒီနေရာမှာ private member တွေ
ကို public member function တွေက ဘယ်လိုမျိုးနည်း obj.calVol( ) ကနေ calVol( ) ဆိုပြီး
calVol( ) function အဖော် volume ကို 5.5*7.5*10.5 = 433.125 လို့ တွက်ပါဝင်မှုမည်။
နောက်ဆုံးအနေနဲ့ကို main( ) ဆီ return သွားသေးတယ်။

- main( ) ကိုပြန်ရောက်တဲ့အခါမှာ obj.calVol( ) ကို print ထုတ်ပြီးအနေနဲ့ 433.125 ဆိုတဲ့
အဖြေကိုတွေ့ရပါလိမ့်မယ်။ အခုနေ Ex604.cpp program ကို run လိုက်ပါဦးဗျာ။ ပုံ
(၆.၁) မှာပြထားတဲ့အတိုင်းပြပါလိမ့်မယ်။ flow diagram ကိုကြည့်ရတာလည်း ပြေလည်ပါတယ်



ပုံ (၆.၁)

6.5 Constructor and Destructor

C++ program တစ်ခုမှာ class declaration မှ action statement တစ်ခုကိုကြေငြာချင်တဲ့အခါ constructor function ကိုသုံးပြီးလုပ်ပါတယ်။ inline function ရဲ့အထဲမှာ ရေးသားတဲ့ constructor နဲ့ပုံစံတူ

Ex605.cpp

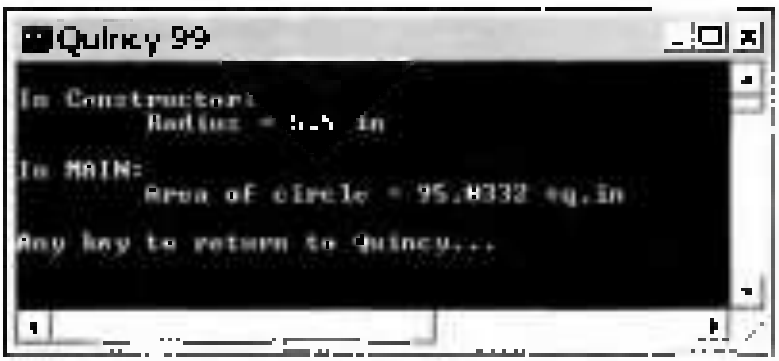
```
// Listing 6.5  Using constructor and destructor
#include <iostream>
const  float  PI=3.14159;

class  circle
{
        float radius;
    public:
        circle(float);   // constructor
        ~circle();       // destructor
        float   caArea();
};

circle::circle(float r)
{
        radius = r;
        cout << "\nIn Constructor\n"
             << "\tRadius = " << radius << "\n\n";
}

circle ~circle() {
}

float circle::caArea()
{
        return PI*radius*radius;
}

int main()
{
        circle obj(5.5);

        cout << "\nAREA\n\tArea of circle = "
             << obj.caArea() << "sqr unit\n";
        return 0;
}
```

ပုံ (၆.၉)

သည်ပါဝင်မှုကို class name ဖြစ်စေသော်လည်း သူ့ return type တည်ဆိုချက်ကိုလည်း constructor function တည်ဆောက်ခြင်းဖြင့် ကျော်လွှားနိုင်ပါသည်။ ဤကိစ္စကို ပြသရန် Ex605 program ကိုဖြည့်သွင်း ဖန်တီးထားပြီး Ex601 program ကို အခြေခံပြုပြင်ထားခြင်းဖြစ်ပါသည်။

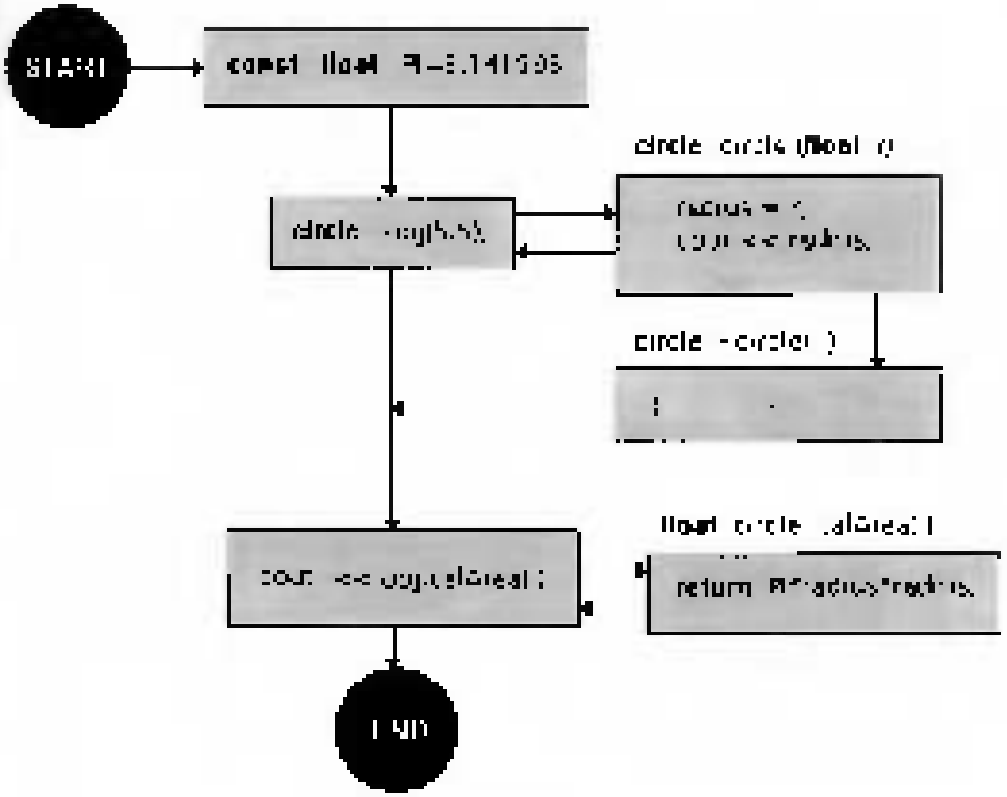
Ex605.cpp program ကိုလေ့လာခြင်းအကြောင်း

- program ရဲ့ main( ) function ထဲမှာ ဘဲ့ဂဏန်း class ကိုသုံးပြီး create ပုံစံနဲ့ object တစ်ခုကို ကြေညာလိုက်ပါတယ်။ ဒီအခါ ကျွန်တော်တို့ရေးထားတဲ့ constructor ကနေ pass ထိုးလိုက်တဲ့ 5.5 ကို ကနေတစ်ဆင့် radius ကိန်းရှင်ကိုသတ်မှတ်ပေးပါတယ်။ ပြီးတဲ့နောက် constructor ထဲမှာ radius value ကို display ပြသပေးပါတယ်။
- နောက် main( ) ထဲကနေတဲ့ obj.calArea( ); ဆိုတဲ့ calArea( ) function ကို call ခေါ်ထားပါတယ်။ ဒီလိုခေါ်လိုက်တဲ့အခါ public member function calArea( ) ထဲကအလုပ်တွေ PI*radius*radius= 3.141593*5.5*5.5 = 95.0332 ကိုလုပ်ဆောင်ပြီးတဲ့အခါ main( ) ဆီ return ပြန်လာပြီး main( ) ကနေ obj.calArea( ) ကို print ထုတ်ပေးခြင်းဖြစ်ပါတယ်။
- ~circle( ) { } ဆိုတဲ့ statement က destructor function ကိုကြေညာထားတာပါ။ ဒီ program မှာအသုံးပြုထားခြင်းမရှိသော်လည်း constructor တွေကနေခေါ်ယူအသုံးပြုပြီး memory allocation လုပ်ထားတဲ့ အပိုင်းတွေကိုဖြေလွှတ်ပေးဖို့ destructor ကို clear လုပ်ပေးသင့်ပါတယ်။ Ex605.cpp program ကို run လိုက်ရင် အဖြေကို ဒုတိယ အကြိမ်ကျော်ကြည့်ပါ။



ဒုတိယ အကြိမ်

- Ex605.cpp program ရဲ့ flow diagram ကို ဒုတိယ အကြိမ်ကြည့်ပြီး ကိုယ်ပိုင် ပြန်လည်ပြုပြင်ပါ။

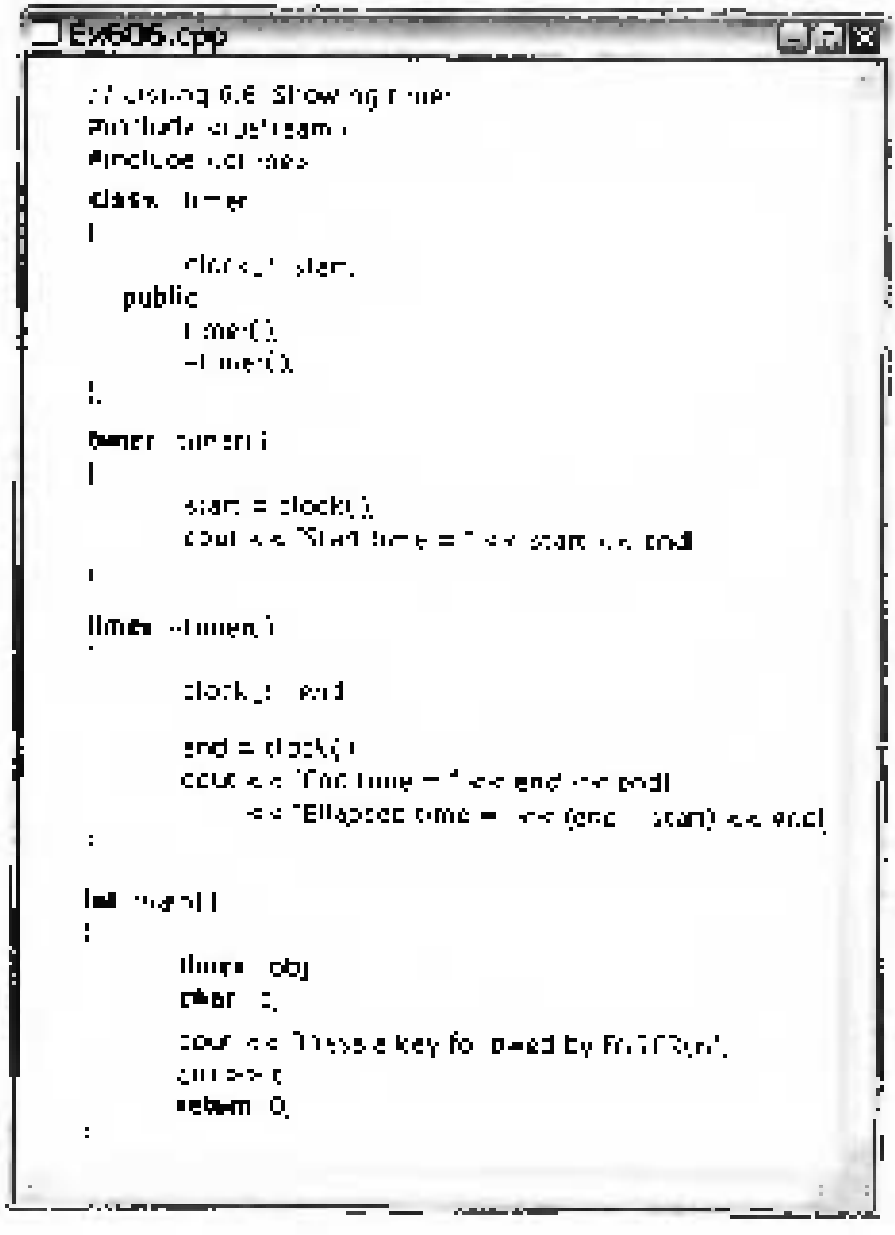


ပုံ (.)

Show Timer

ဥ Prog3.cpp program တွင်ပုံ timer အမည်ရှိသည့်အတွဲ constructor နှင့် destructor တို့ကို အသုံးပြု၍ program စတင်အလုပ်လုပ်ခဲ့သော ပုံ (.) ကို ဖြစ်စေသော program ကိုရေးသားပြုစုပါ။

- main( ) အတွင်း obj ဟူသော timer object အသစ်ကို define ပြုလုပ်ခြင်းအားဖြင့် constructor function အား အလုပ်လုပ်စေပြီး clock( ) ဟူသော timer( ) ကွန်ပျူတာအချိန်ကို ယူသည့် standard library function ကိုအသုံးပြု၍ start = clock( ) =0 ဟု initialize ပြုလုပ်ခဲ့ကြောင်း ကိုက်ညီကြောင်းကို Start time = 0 ဟူသော စာသားဖြင့် ဖော်ပြသည်ကို main( ) ကြီး၏ နောက်ဆုံးတွင် Press a key followed by ENTER: ဟူသော စာသားဖြင့် ဖော်ပြထားသည်ကို



ပုံ (၆.၁၂)

ကျွန်တော်တို့က key ကိုခေါက်ပြီး ENTER key ကို နှိပ်လိုက်တဲ့အခါ program ထဲက destructor function ဖြစ်တဲ့ ~timer() ကို ခေါ်သွားပြီး destructor ထဲက code တွေ အလုပ်လုပ်သွားတဲ့အခါ timer ရဲ့ start time နဲ့ end time တွေကို ပြသပေးသွားမှာဖြစ်ပါတယ်။

- destructor function ထဲမှာ endt = clock( ); ကို assign လုပ်ပေးလိုက်တဲ့အတွက် ကြေး ဆောင်ယူ ရောက်သောအချိန်ကို end သို့ထည့်ပေးမယ် end ကို display လုပ်ခြင်းဖြင့် အချိန်ကိုးခုမြောက်ဖယ် elapsed time ကတော့ endt - start အနှုတ်ခြင်းကြောင့် ဖြစ်ပေါ်လာတဲ့ display လုပ်ပြတဲ့ destructor ဖြစ်တဲ့ object ကို ဖျက်သိမ်းသွားမှာဖြစ်ပါတယ်
- ဒုတိယ ဥပမာအဖြစ် Ex606.cpp program ကို run ကြည့်ပါ။ Press a key followed by ENTER ဆိုတာကြောင့် keyboard က character တစ်လုံးကို ဖြတ်ပေးရင်း timer ဘယ်လိုလုပ်ဆောင်ပေးသွားမလဲဆိုတာ

ဒုတိယ ဥပမာ

8.5 Classes and const

။။ C++ program တစ်ခုမှာ object တစ်ခုကို const အဖြစ် declare လုပ်ထားမယ်ဆိုရင် const အဖြစ်ကြေး class member function တွေကနေ object ကို call ခေါ်လို့မရပါဘူး call ခေါ်ခိုင်းရင် လုပ်ဆောင်ချက် ။။ member function declaration မှာ const qualifier ထိုးပေးထားရပါတယ်။ ဒုတိယ ကျော်ခဲ့ Ex607.cpp program တစ်ခုမှာ const qualifier တွေကြေးမည်ကိုဆိုကြည့်ပါမယ်။ ရလဒ်ကြည့်ပါ

- Ex607.cpp program ကို run ပြီးပြန်ဆိုရင် Ex605.cpp program ကို run ထားတဲ့အရာ ပုံစံတူပါတယ်။ အဖြေတွေကတော့ Ex607.cpp ထဲက const qualifier ကြောင့်ပြီး Ex605 program ကိုပြန်ပေးသွားတဲ့ပုံစံပါ

Ex607.cpp

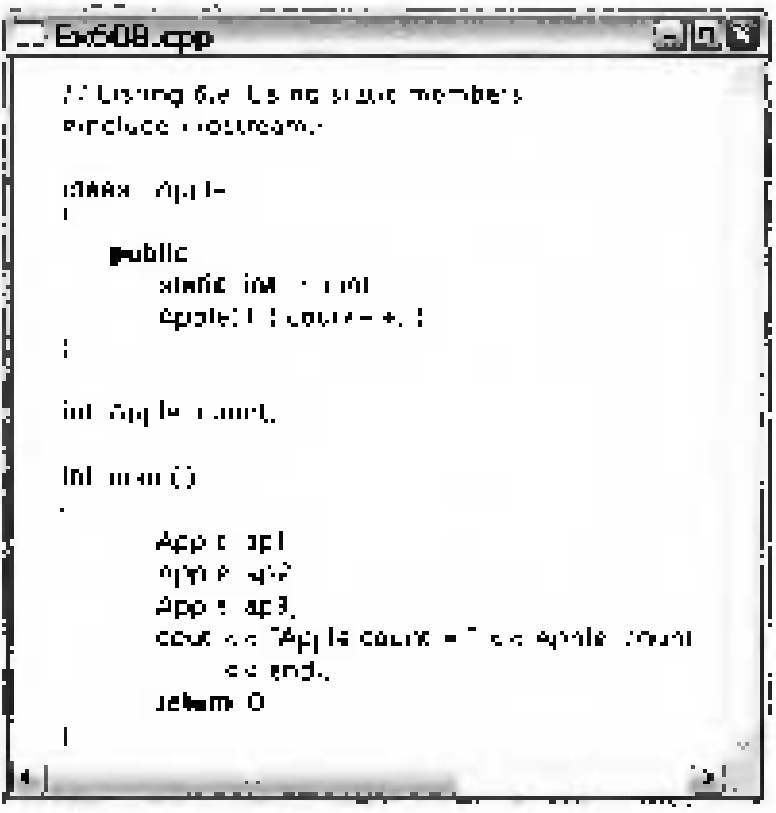
```
// Listing 6.7  Using const qualifier
#include <iostream.h>
const float PI=3.14159;

class circle
{
	float radius;
  public:
	circle(float r)
	{
		radius = r;
		cout << "\nIn Constructor \n"
			<< "\tRadius = " << radius << " m.\n\n";
	}

	void calArea() const
	{
		cout << "In MAIN \n"
			<< "\tArea of circle = "
			<< PI*radius*radius << " sq m\n";
	}
};

int main()
{
	const circle obj(5.0);

	obj.calArea();
	return 0;
```

ပုံ (၆-၅)

၆.၅ static Members

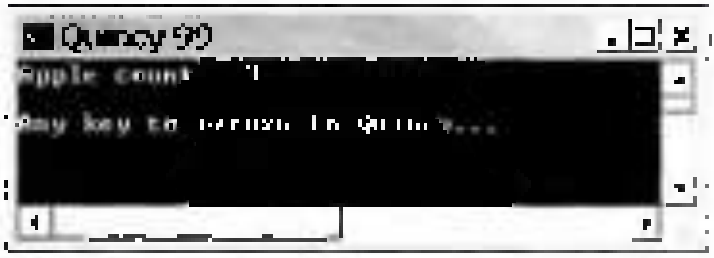
၁။ C++ program ၌ class member [illegible] static [illegible] data member

class instance တစ်ခုချင်းစီတွင် global value တစ်ခုဖြစ်သော static ကို class definition ပြု
လုပ်ခြင်းအားဖြင့် အဖြစ်အပျက်ကိုရှောင်ကြဉ်နိုင်ပုံကို ပြသရန် ဖန်တီးထားသော Ex608.cpp program ဖြင့်
static member အသုံးပြုပုံကို ကြည့်ရှုနိုင်ပါသည်။



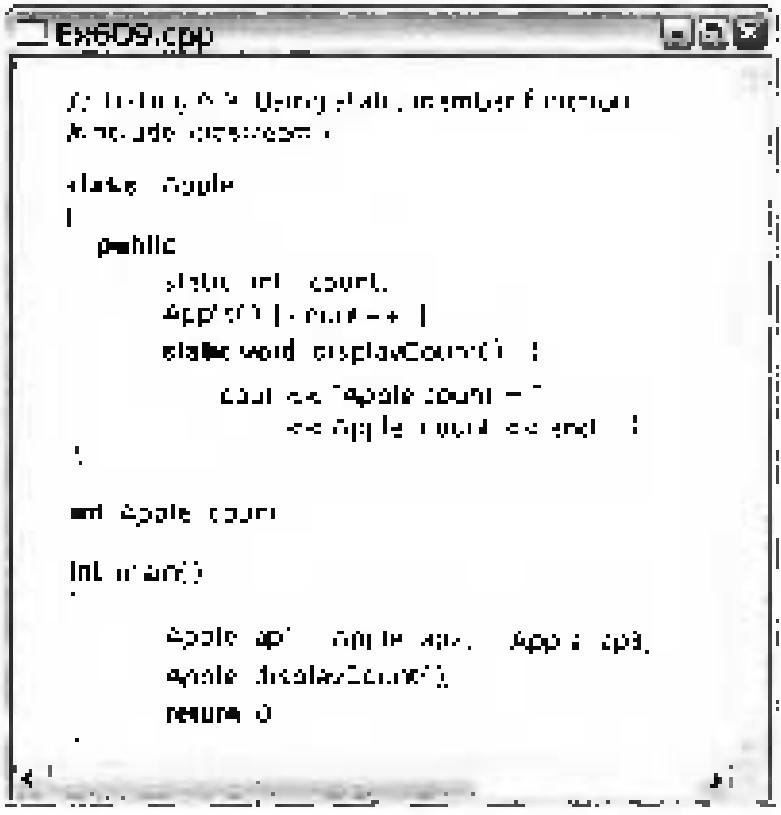
Ex608.cpp program ကို စမ်းသပ်ကြည့်ခြင်းဖြစ်ပုံ

- class Apple definition ထဲမှ static int count; ဖြင့်ရေးသားထားသော statement သည် static data member ကို declare ပြုလုပ်ထားသည်။ program စတင်သော main( ) တွင် Apple class instance (3) ခုကို define လုပ်ထားသဖြင့် constructor ဖြစ်သော count ကို တစ်ခုချင်းစီတွင် ခေါ်သွားမည်ဖြစ်ပြီး Apple::count ကို print ပြုလုပ်ခြင်း (3) ခြင်းသာဖြစ်သည်။ ဤသို့ဖြင့် Ex608.cpp program ကို run ခြင်းဖြစ်သည်။



static Member Functions

Ex608.cpp program ရဲ့ static member function အလုပ်လုပ်ပုံကိုနားလည်ဖို့ Ex609.cpp ကိုကြည့်ပါ။ class definition ထဲမှာ member function header ရှေ့မှာပြုလုပ်ထားတဲ့ static ကြေငြာချက်ကိုသတိပြုပါ။ static member function ကို call ခေါ်တဲ့အခါ class name နဲ့ခေါ်ရပါတယ်။



6.6 Overloaded Constructors

ဒု ယခုကိုယ်တွေ့ပုံစံက overloaded function တွေအကြောင်းကို ဆက်နွယ်ပြောဆိုခဲ့ကြပြီး ယခုလေ့လာမယ့် constructor သည် class member function တစ်ခုပဲဖြစ်တဲ့ အတွက်ကြောင့် function တိုင်း overload လုပ်နိုင်ပုံ ဖြစ်ပါတယ်။ ဥပမာအနေနဲ့ Ex6010.cpp program မှာ Box constructor ကို overload လုပ်ခြင်းအကြောင်း ဖြစ်ပါတယ်။ ဆက်လေ့လာကြည့်ပါ။

Ex6010.cpp

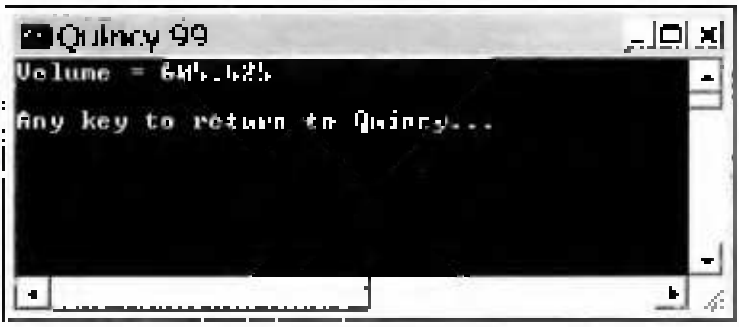
```
// Finding the Over loaded Constructors
#include <iostream.h>

class Box
{
        float height,width,depth;
    public:
        Box() {
        Box(float h, float w, float d);
          {
                height = h;
                width = w;
                depth = d;
          }
        float calVol()
          {
                return height*width*depth;
          }
};

int main()
{
        Box MyBox(7.5,8.5,9.5);
        Box *ptrBox;
        ptrBox = &MyBox;
        cout << "Volume = " << ptrBox->calVol()
             << endl;
        return 0;
}
```

Ex6010.cpp program ကိုစတင်လေ့လာခြင်း

- စတင်း main( ) ထဲမှာ parameter ပါတဲ့ Box constructor ကို define ပုံစံမျိုး class definition ထဲက Box constructor ကို call ခေါ်ပြီးသားဖြစ်ပေမယ့် constructor ထဲမှာပဲ parameter နဲ့တဲ့ constructor ထဲကိုဖျတ်ခေါ်ပေးနိုင်း height = h = 7.5 width = w = 8.5 နဲ့ depth = d = 9.5 ကို set လုပ်ပေးလိုက်သော် ပြီးစောင့်ခဲ့ main( ) အပြီးနဲ့ပေးပါ။

- နောက်ပါ main( ) ထဲမှာ otherBox ဆိုတဲ့ Box instance တစ်ခုတည်ကို define လုပ်ပါ တယ်။ ဒီတော့ class definition ထဲက Box( ) { }; constructor အတိုင်းတော့ အလုပ်လုပ်ပါ။ main( ) ကိုပြန်ပြောင်းပါတယ်။

- main( ) ထဲမှာ thisBox ရဲ့ value ကို other Box နဲ့ assign လုပ်ပေးခဲ့တဲ့အတွက် otherBox.calVol( ) ကို display လုပ်ခိုင်းထဲမှာ ဖြစ်လို့ float calVol( ) function ကို သွားမှာဖြစ်၊ height*width*depth = 7.5*8.5*9.5 = 605.25 ကိုရတာကြောင့် return လုပ်ပေးလိုက်ခြင်းဖြစ်။ ဖြစ်လာဒဲ့ဒီဥပမာကို Ex6010.cpp program ကို run ကြည့်လိုက်ရင်တော့ ဒီအတိုင်း ထွက်ပေါ်လာပါလိမ့်မယ်။ ပုံ (၆-၁၀) အတိုင်းပေါ်ပါ။



ပုံ (၆-၁၀)

More Overloaded Constructors

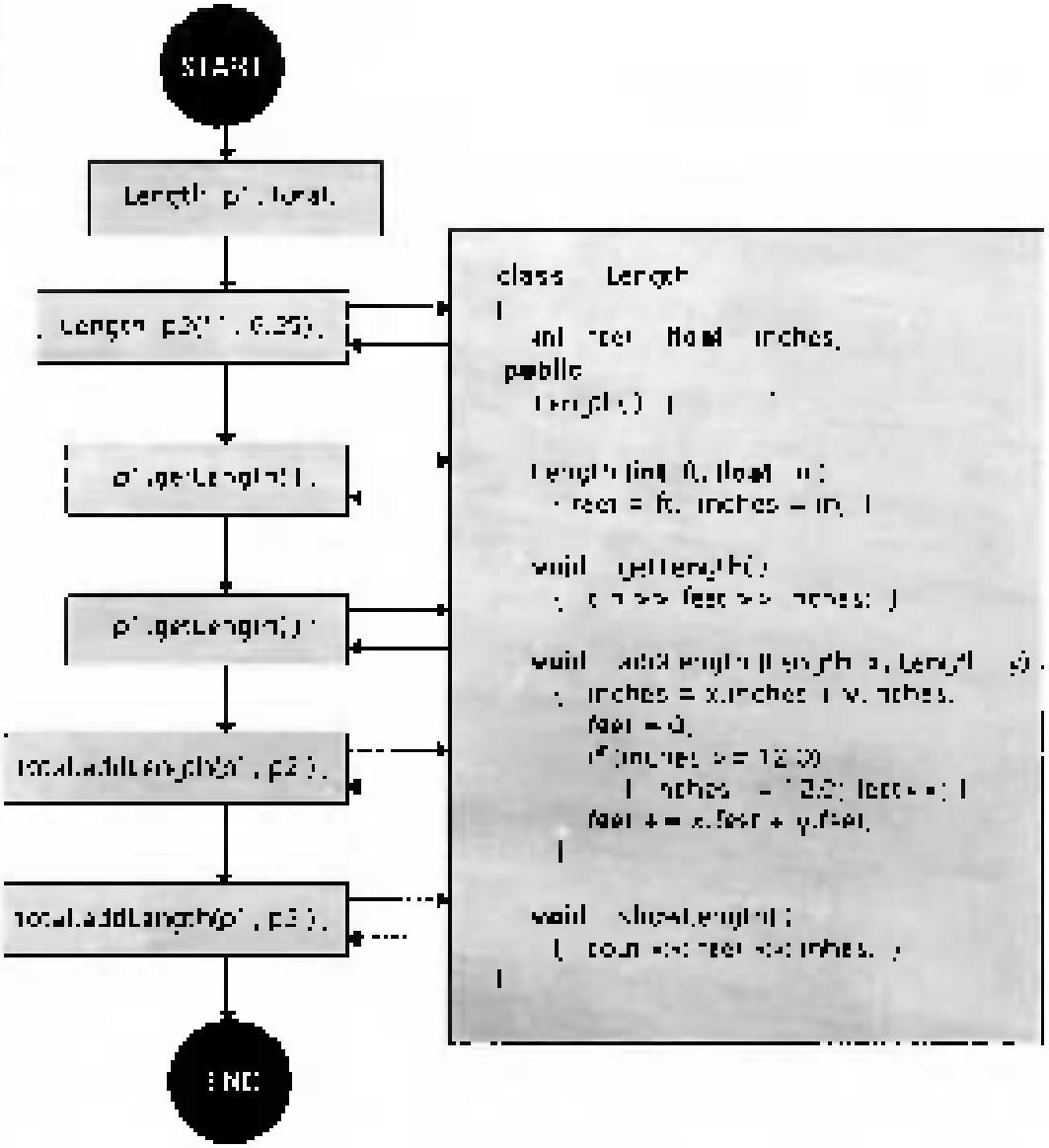
C++ program တွေမှာ overloaded constructor တွေကို ပိုပြီးကောင်းစေနိုင်တဲ့နည်းလမ်းတစ်ခုကိုဆိုပြီးရှင်းပြပါမယ်။ Ex6011.cpp program ကိုစတင်လေ့လာကြည့်ပါ။ ဒီ program မှာ class တစ်ခုတဲ့ ဆောက်လုပ်ရေးတွေမှာ nested လုပ်ပေးထားပါတယ်။

```
// Listing 6.11. More overloaded constructors
#include <iostream>

class Length
{
      int   feet;
      float  inches;

   public:
      Length( ) { }

      Length (int ft, float in)
      {
            feet = ft;
            inches = in;
      }

      void getLength( )
      {
            cout << "\nEnter feet    ";
            cin >> feet;
            cout << "Enter inches : ";
            cin >> inches;
      }

      void showLength( )
      {
            cout << feet << "'-" << inches << "\"" << endl;
      }

      void addLength (Length  x, Length  y)
      {
            inches = x.inches + y.inches;
            feet = 0;
            if (inches >= 12.0)
```

```
        {
            inches -= 12.0;
            feet++;
        }
        feet += x.feet + y.feet;
    }
};

int main( )
{
    Length p1, total;
    Length p2(11,6.25);

    p1.getLength( );
    total.addLength(p1,p2);
    cout << "\nPiece1 = ";

    p1.showLength( );
    cout << "\nPiece2 = ";
    p2.showLength( );

    cout << "\nTotal Length = ";
    total.showLength( );

    return 0;
}
```

Ex6011.cpp program အတွက် flow diagram ကို ဆွဲပြီး program လုပ်ငန်းစဉ်များကို Ex6011.cpp program အိုဘဂျက်ပရိုဂရမ်မင်းဖြင့်ရေးသားပုံကို

- ပထမဦးစွာ p1 နှင့် total ဆိုတဲ့ class Length object (2) ခုကို create လုပ်ပါတယ်။ ထပ်မံ ပြောင်းလဲ constructor Length( ) function တစ်ခုမှာပြီး feet နှင့် inches ဆိုတဲ့ private member variable တွေအတွက် define လုပ်ပေးထားပြီးတော့ member functions တွေကတော့ main( ) ထဲတွင်ပြန်ခေါ်သုံးပါတယ်။



ဖွံ့ဖြိုးရေး

- ဒုတိယ object p2 (11,6.25) ကို define ရယူခြင်းအတွက် overloaded constructor Length(int ft, float in) ထဲသို့ရောက်သွားပြီး feet = ft =11 နှင့် inches = in =6.25 ကို assign ရယူသွားမှာ ဖြစ်ပါသည်။
- p1 object အတွက် get_length( ) function ကို call ခေါ်လိုက်သော်လည်း function ထဲမှ feet နှင့် inches အတွက် data များကိုရယူခြင်းမပြုသေးသဖြင့် feet = 15 ကို ဆက်လက်ဆောင်ရွက်သွားပါသည်။

ဆိုပါဦး ဒါဆိုရင် p1.feet = 15 ဖြစ်သွားပါပြီ။ ဒီတော့ဆက်ပြီး inches အတွက် 11 ကိုပြန်လာတဲ့ ပေးပြီးနှိုင် p1.inches = 11 ဖြစ်သွားပါပြီ။ ဒုတိယ mark ; function ကိုပြန်ခေါ်ပြီး

- total object အတွက် add_length( ) function ကို call ခေါ်ပါတော့မယ်။ argument များ p1 ၊ p2 ဖြည့်ပေး object (?) ပြန်ပေး ဒီတော့ x = p1 ၊ y=p2 ကို argument များအဖြစ်သတ်မှတ်ခြင်းဖြစ်ပြီး လုပ်ဆောင်ပုံက inches = x.inches + y.inches − p1.inches + p2.inches = 11 + 6.25 = 17.25 ကို ကွက်ပေးပါတယ်။ ဒီနောက် work နဲ့ အလုပ်လုပ်ပြီး inches = 17.25 ဟာ 12 ထက် ကြီးသော ကြောင့် ဝင်ပေးလိုက်မယ်ဆိုပြီးတော့ <test expression> ဟာ true ဖြစ်ခဲ့တာကြောင့် inches = inches - 12 = 17.25 - 12 = 5.25 ကိုပြန်ယူသွားပြီး feet = feet + 1 = 0 + 1 = 1 ပေးလိုက်ပြီးတော့ feet = x.feet + y.feet. statement အရ feet = feet + x.feet + y.feet = 1 + 15 + 11 = 27 ကိုပြောပေးလိုက်ပြီး total object ထဲမှာ feet ကို 27 ပဲ ရှိနေတော့တာပေါ့ဗျာ။
- ဒါဆိုရင် total.showLength( ); statement အရ total object ရဲ့ feet နဲ့ inches တန်ဖိုးတွေကို print ထုတ်ပေးလိုက်ပြီးတော့ program ကို run ပုံက အောက်ပါအတိုင်း ဖြစ်ပါလိမ့်မယ်။ (ပုံ 6.11)

ပုံ (ပုံ. ၂၂)

Complex Overloaded Constructors

// Listing 6.12 More overloaded constructors

```
#include <iostream>

class Point
{
        int x,y;
    public:
        Point(int xp=0, int yp=0)
                {x = xp; y = yp; }
        void display( )
                {cout << x << "," << y << ")" ;}
};

class Lines
{
        Point start, stop;
    public:
        Lines(Point st, Point sp)
                {start = st; stop = sp; }

        Lines(int x1, int y1, int x2, int y2)
            {
                Point st(x1,y1);
                Point sp(x2,y2);
                start = st;
                stop = sp;
            }

        void Draw( )
            {
                cout << "\nFrom (";
                start.display( );
                cout << " to (";
                stop.display( );
            }
};
```

```
int main( )
{
        Point   p1(10,15);
        Point   p2(25,45);
        Lines   line1(p1,p2);
        Lines   line2(10,20,30,40);

        line1.Draw( );
        line2.Draw( );
        cout << endl;
        return 0;
}
```

ဥ Ex6d12.cpp program ကိုအကျဉ်းချုပ်ခြင်း

- အရင်း main( ) ထဲမှာ Point class instance တစ်ခုဖြစ်တဲ့ p1 ကို define လုပ်ခဲ့တယ်။ Point constructor သုံးခုရှိ data (2) ခု pass လုပ်ပေးထိုက်သောကြောင့် x = xp = 10 နဲ့ y = yp = 15 ဖြစ်ပြီး ဒီတော့ p1.x = 10 နဲ့ p1.y = 15 ကို main( ) ထဲမှာရှိတဲ့ ဒုတိယမြောက် instance p2 ကိုလည်း define လုပ်ခဲ့တယ်။ Point constructor သုံးခုရှိ x = xp = 20 နဲ့ y = yp = 45 ကိုပဲ pass လုပ်ပေးထိုက်သောကြောင့် ဒီတော့ p2.x = 20 နဲ့ p1.y = 45 ဖြစ်သွားတယ်။
- ဒုတိယအဆင့် main( ) ထဲမှာ Lines class instance တစ်ခုဖြစ်တဲ့ line1 ကို define လုပ်ခဲ့တယ်။ Lines constructor တစ်ခု object (2) ခု pass လုပ်ပေးထိုက်သောကြောင့် ဒီတော့ Lines (Point st, Point sp) ဆိုတဲ့ constructor သုံးခုထဲက ပထမတစ်ခုကို start = st = p1 နဲ့ stop = sp = p2 ကိုပဲ assign လုပ်ပေးသွားတာကြောင့် ဒီတော့ line1.start.x = 10 နဲ့ line1.start.y = 15 နဲ့ line1.stop.x = 25 နဲ့ line1.stop.y = 45 ဖြစ်သွားတယ်။
- Lines class instance တစ်ခုဖြစ်တဲ့ line2 ကို define လုပ်ခဲ့တယ်။ Lines constructor သုံးခုရှိ parameter (4) ခုကို pass လုပ်ပေးထိုက်သောကြောင့် Lines (int x1, int y1, int x2, int y2) ဆိုတဲ့ constructor ပထမတစ်ခုဖြစ်သွားတယ်။ ဒီတော့ x1 = 10 y1 = 20 နဲ့ x2 = 30 နဲ့ y2 = 40 ကိုပဲ assign လုပ်ပေးသွားပါလိမ့်မယ်။ ဒုတိယ constructor ကိုသုံးခဲ့ Point class instance st(x1,y1) ကို define လုပ်ပါသောကြောင့် ဒီတော့ x = xp = x1 = 10 နဲ့ y = yp = y1 = 20 ကို pass လုပ်ပေးသွားခဲ့တဲ့အတွက် line2 st.x = 10 နဲ့ line2.st.y=20 ဖြစ်သွားတယ်။ ဒုတိယတွဲလာတဲ့ line2.sp.x = 30 နဲ့ line2.sp.y = 40 ဖြစ်ပေါ်ခဲ့ပြီးတော့ အရင်က st ကို private member ဖြစ်လို့ start နဲ့ assign လုပ်ပေးလိုက်တဲ့အခါ line1.start.x =

line1.stop.x ကြိုတွေးပြီး ပုံတူကူး line1.spy = line1.start.y + line2.start.y - line2.stop.x - 30 နဲ့ line2.spy = line2.stop.y - 40 ပုံတူကူးခြင်းမပြုလုပ်ပါ။

- main( ) ထဲမှာ line1.Draw( ) ကို call ခေါ်ထားတဲ့အတွက် Draw( ) function ထဲတွင်ရှိသော start.display( ) ကို call ခေါ်ထားသောကြောင့် ထိုမှ display( ) function ဆင်းသွားပြီး x နှင့် y ကို display ပြသခြင်းဖြစ်ပါသည်။ ဒုတိယ line1.start.x နှင့် line1.start.y တို့ကို display ပြသခြင်းလုပ်ငန်းစဉ်အတွက် ခေါ်ယူရန် stop.display( ) သေချာပြီး line1.stop.x နှင့် line1.stop.y ကိုလည်း display ပြသပါသည်။
- line2.display; ဆက်လက်ဆောင်ရွက်ခြင်းအတွက် display ပြသခြင်းဖြစ်ပါသည်။ Ex6011.cpp program ကို run ကြည့်ခဲ့ပါက ပုံ (၂.၂) ကဲ့သို့ ပေါ်လာမည်ဖြစ်ပါသည်။

ပုံ (၂.၂)

Copy Constructors

C++ program တွင် a new object တစ်ခုကို class တစ်ခုတွင် existing object နှင့် initialize လုပ်ဆောင်မှုအတွက်အတွက် constructor ကို copy constructor ဟုခေါ်ပါသည်။ ၎င်းတွင် function တစ်ခု၊ new object တစ်ခုကို passing by value များ copy ကို pass ပြုလုပ်သည့် copy constructor ကို အဓိကထားသည်။ ထို့အတူ class object ကို by value အနေဖြင့် return ပြန်ပေးခြင်းတို့မှာလည်း copy constructor ကိုအသုံးပြုပါသည်။ အသုံးပြုထားသည့် Ex6012.cpp program ကိုလေ့လာကြည့်ပါ။

```
// Listing 6.13  Using copy constructor

#include <iostream>
#include <cstring>

class  Date
{
        int     mo, da, yr;
        char* month;

  public
        Date(int m = 0, int d = 0, int y = 0)      // constructor definition

        {
                static char* mos [ ] = {
                                        "January", "February", "March",
                                        "April", "May","June", "July",
                                        "August", "September", "October",
                                        "November", "December"
                };

                mo = m;
                da = d;
                yr = y;

                if (m != 0)
                {
                        month = new char[strlen(mos[m-1]) + 1];
                        strcpy(month, mos[m-1]);
                }
                else
                        month = 0;
        }

        Date(const Date&  dt)    // Copy constructor definition
        {
                mo = dt.mo;          da = dt.da;    yr = dt.yr;
```

```
        if (dt.month != 0)
        {
                month = new char [strlen(dt.month)+1];
                strcpy(month,dt.month);
        }
        else
                month = 0;
    }


    ~Date( )    // The destructor definition
    {
        delete [ ] month;
    }

    void  display( ) const
    {
        if (month != 0)
                cout << month << ' ' << da << ", " << yr << endl;
    }
};

int  main( )
{
    Date    birthday(10,25,1947);
    birthday.display( );

    Date newday = birthday;
    newday.display( );

    Date lastday(birthday);
    lastday.display( );

    return 0;
}
```

- ဒုတိယ main() ထဲတွင် class Date instance တစ်ခုဖြစ်သော birthday object ကို define လုပ်ပြီး 10, 25, 1947 တို့ကို pass ပေးပို့ခေါ်ယူခြင်း ဖြစ်ပါသည်။ Date constructor တွင်ပါသည့် m = 10 d = 25 နှင့် y = 1947 တို့ကို assign လုပ်ပေးသွားမည်။ ထို့နောက် mo = m = 10; da = d = 25 နှင့် yr = y = 1947 တို့အတိုင်း assign ပြုလုပ်ပေးသွားမည်။ if (m != 0) ဆိုသော expression က true ဖြစ်သောကြောင့် if block ထဲတွင်ပါသော month = new char [len(mas[m-1])+1] = new char [strlen(mas[4])+1] = new char ["October" + 1] = new size[8] အဆင့် memory size သတ်မှတ်ပေးခြင်း ဖြစ်ပြီး strcpy (month, mas[m-1]); statement အရ 'October' ကို month ထဲသို့ ကူးယူသွားခြင်း ဖြစ်ပါသည်။ ထို့နောက် main() လုပ်ငန်းဆောင်တာ ပြီးဆုံးသွားပါပြီ။ ထို့ကြောင့် main() လုပ်ငန်းဆောင်တာ ပြီးဆုံးသွားပါပြီ။
- birthday.display() ဟူသော call ခေါ်ယူခြင်းအားဖြင့် display member function လုပ်ငန်းဆောင်တာ ထဲသို့ month='October' da=25 နှင့် yr=1947 တို့ကိုလည်း ထည့်သွင်းပေးလိုက်ပါသည်။ display လုပ်ငန်းဆောင်တာ ရဲ့အဆုံး date ကို display လုပ်ပြီး ကောင်းစွာပြီးပါပြီ။
- ဆက်လက်၍ data အမြင်ပေး birthday object ကို new Data object ဖြစ်သည့် newday ကို assign လုပ်ပေးလိုက်သည်။ ဒါဆိုရင် birthday data တွေကို newday က သတ်မှတ်ရယူပြီး newday.display() ကို call ခေါ်ယူခြင်းအားဖြင့် ပေးပို့ခေါ်ယူလိုက်သည့်အတိုင်း ရလဒ်အနေဖြင့် date ကို display လုပ်ပေးသွားပါသည်။
- နောက်ဆုံးအနေဖြင့် birthday အမြင်ကို new object အဖြစ် create လုပ်ပြီး birthday object ကို argument အနေနဲ့ pass ပေးပို့လိုက်သည်။ ဒီတော့ copy constructor ထဲတွင်ပါသည့် dt = birthday ဖြစ်သွားခဲ့ပါသည်။ ထို့နောက် mo = dt.mo = birthday.mo = 10 ; da = dt.da = birthday.da=25 နှင့် yr = dt.yr = birthday.yr = 1947 ဖြစ်သွားပြီး ရလဒ်အနေဖြင့် ကျွန်တော်တို့ ဒုတိယ ရလဒ်ကို ရရှိခဲ့ပါသည်။ Ex6013.cpp program ကို run ကြည့်ပါမယ်။

ပုံ (၁၂)

6.7 Conversion Constructors

[illegible] parameter list [illegible] entry [illegible] constructor function [illegible] conversion constructor [illegible] parameter type [illegible] constructor class [illegible] time_t [illegible] cpp program [illegible] C standard time( ) function [illegible] return [illegible] time_t [illegible] Date class [illegible]

```
// Listing 6.11: Using conversion constructors
#include <iostream>
#include <ctime>
#include <stdio.h>

class Date
{
      int    month, day, year;
  public:
      Date(time_t  now)              // conversion constructor function
      {
            tm* lm = localtime(&now);
            day = lm->tm_mday;
            month = lm->tm_mon + 1;
            year = lm->tm_year;
            if (year >= 100)    year -= 100;
      }

      void   display( )
      {
            char yr[5];
            if (year < 10)   sprintf(yr, "0%d", year);
            else             sprintf(yr, "%d", year);
            cout << month << '/' << day << '/' << yr << endl;

      }
};
```

```
int main( )
{
        time_t now = time(0);   // get today's date and time
        cout << asctime(gmtime(&now)) << endl;
        Date dt(now);
        dt.display( );
        cout << endl;
        return 0;
}
```

Ex5b14.cpp ကို စမ်းသပ်ကြည့်ခြင်း

- main( ) ထဲမှ ပထမဆုံးအကြိမ် system ကနေတောင်းဆို time( ) function ကနေ time_t object အဖြစ်ရယူခြင်းဖြစ်သည်။ cout << asctime(gmtime(&now)) statement ၏ asctime( ) function မှာ std::tm structure ကို point လုပ်ထားတဲ့ now တွင်ရှိ date & time ကို string format နဲ့ display ပြပေးခြင်းဖြစ်သည်။
- Date dt(now); ဆိုတဲ့ statement မှာ conversion constructor ကိုခေါ်ယူခြင်းဖြစ်ကာ Date object တစ်ခုကို construct ပြုလုပ်ပြီး ထိုအဖြစ် conversion constructor function မှာ Date format ကိုဖောင်းခြင်းဖြစ်ပြီး dt.display( ); ကနေ display ပြပေးခြင်းဖြစ်သည်။ Ex5b14.cpp program ကို run ကြည့်မည်ဆိုလျှင် (၂-၃) ကဲ့သို့ ပေါ်လာမည် ဖြစ်ပါသည်။

ပုံ (၂-၃)

6.6 Member Conversion Functions

ကျွန်တော်တို့ class တစ်ခုမှာ define လုပ်ထားတဲ့ member conversion function တစ်ခုက class ရဲ့ object တစ်ခုကို data data type တစ်ခုသို့ ပြောင်းပေးမယ့် object တစ်ခုဖြစ်ပါတယ်။ ဒီ function ကို declare လုပ်တဲ့အခါမှာ class declaration ထဲမှာ အောက်ပါအတိုင်း ရေးပေးရပါတယ်။

```
operator  long( );
```

operator က C++ keyword တစ်ခုဖြစ်ပြီး long က converted ပြောင်းမယ့် data type တွေရဲ့ type specifier ဖြစ်ပါတယ်။ member conversion function ရဲ့ header က classname::operator long( ) ဖြစ်ပုံဖြစ်နည်းတော့ ကောင်းကောင်းသိရမှာပေါ့။ Ex6C15.cpp program မှာ member conversion function အတွက် ဥပမာပြထားပါတယ်။

```
// Listing 6.15: Using member conversion function
#include <iostream.h>

class  Date
{
      int   month, day, year;
  public
      Date (int m, int d, int y)
            { month = m; day = d, year = y; }
      operator long( );                    // member conversion function.
};


// The member conversion function.
Date::operator long( )
{
      static int   days[ ]={ 31,28,31,30,31,30,
                             31,31,30,31,30,31};
      long   x = year - 1900;
      x *= 365;
      x += year / 4;
```

```
        for (int i = 0; i < month-1; i++)
                x += days[i];
        x += day;
        return x;
}

int main( )
{
        Date   birthday(12, 25, 1997);
        long   since = birthday;
        cout << "Cummulative days since 1900 = " << since << endl;
        return 0;
}
```

Ex6015.cpp ကို trace လုပ်ကြည့်ပါအိုးဗျာ

- စနေ့: main( ) ထဲမှာ Date class object တစ်ခုဖြစ်တဲ့ birthday ကို create လုပ်လိုက်၊ Date constructor ထဲကို parameter (3) ခု pass လုပ်ပေးပါတယ်၊ constructor ထဲမှာ month = m = 10 day = d = 25 နဲ့ year = y = 2003 တို့ကို assign ပြီးတဲ့နောက် ပြီးရင် main( ) ထဲမှာရှိနေတဲ့ဒုတိယမြောက် long since = birthday; ဆိုတဲ့ statement ကနေ member conversion function ဆီသို့ရောက်သွားပြီး

  x = year - 1900 = 2003 -1900 = 103 years

  x *= 365 = 103*365 = 37595 days

  x += year/4 = 37595 + 2003/4 = 38095 days

  ဒီတွေးတော (4) နှစ်တစ်ဝက်မြောင် တစ်ခုတိုးတိုးလာတဲ့ တည့်မတ်ပြီးတာပါပဲ၊

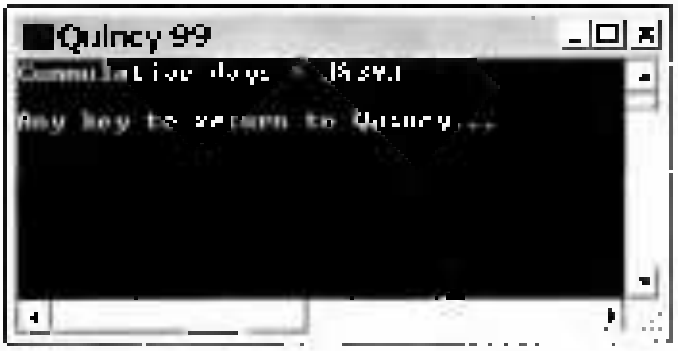
- for loop ထဲမှာ (9) ခေတာလုံး ရှင်းလင်းခဲ့ရတဲ့ပုံစံကို x ထဲတွေးတိုးကြည့်ပါဦးမယ်၊

  x += days[i] = 38095 + 31 + 28 + 31 + 30 + 30 + 31 + 31 + 30

  = 38368 days

  x += day = 38368 + 25 = 38393 days ဖြစ်သွားတာပေါ့၊

  ပါဆိုရင် birthday data type ကနေ long type ကိုရောက်သွားတာက ဘယ်လိုမဟုတ်သေးတာ ပါလဲဆိုတော့ member conversion function ကကြောင့်ပါပဲ၊ Ex6015.cpp program ကို run ထားကြည့်ပါဦး၊ (6, 25) အကြောင်းအတိုင်းဖြစ်ရမှာပါ၊



Improved Member Conversion Function

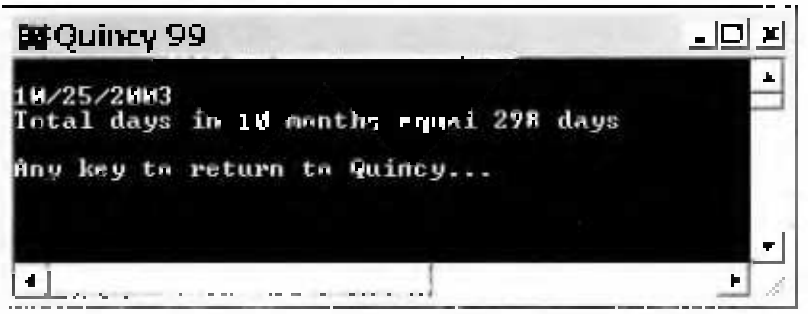
Ex6016.cpp program ဥာ Date::operator customDate( ) function ကို const ကို declare ပုံစံမျိုးမပေးထားဘဲ ဒီအတိုင်း customDate( ) လို့ပဲ Date object ရဲ့ data value ကို modify လုပ်ပြီး temporary object ကြွပဲ့ customDate ကိုပဲ return ပြန်ပေးတဲ့ data modify လုပ်ခွင့်ပေးထားတဲ့အတွက် main( ) function ဥာ single Date object dt ရဲ့ customDate object ကို ပုံမှန်အတိုင်း construct လုပ်ပြီး dt.display( ) နဲ့ assign လုပ်ပေးခြင်းအားဖြင့် Date data value ဟာ customDate data type ကို ပြောင်းလဲသွားပါတယ်။

```
// Listing 6.16: Improved member conversion function
#include <iostream>

class CustomDate
{
      int mo, da, yr , totalDays;
  public:
      CustomDate( ) {      }

      CustomDate (int m, int d, int y)
```

```
        { mo = m; da = d; yr = y;}
    void  display( ) const
        {
            cout << endl << mo << '/' << da << '/' << yr << end
                << "Total days in " << mo << " months equa "
                << totalDays << " days\n";
        }

    void  setDay(int d)
        { totalDays = d; }
};


class  Date
{
    int    mo, da, yr;

  public:
    Date (int m, int d, int y)
        { mo = m; da = d; yr = y; }

    operator CustomDate( ) const,          // conversion function
};


// Member conversion function (CustomDate <- Date).
Date ::operator  CustomDate( ) const
{
    static  int days[] = { 31,28,31,30,31,30,
                           31,31,30,31,30,31  };
    CustomDate   cd(mo,da,yr);
    int   day = da;
    for (int i = 0; i < mo-1; i++)
        day += days[i];
    cd.setDay(day);
    return  cd;
}
```

```
int  main()
{
        Date     dt(10,25,2003);
        CustomDate    cd;

         // Convert Date to CustomDate via assignment.
        cd = dt;
        cd.display( );
        return 0;
}
```

Ex6016.cpp program ကို run ကိုးကြည့်ရင် ပုံ (၆-၂) ကဲ့သို့ မြင်တေ့ရပါလိမ့်မယ်။



ပုံ (၆-၂)

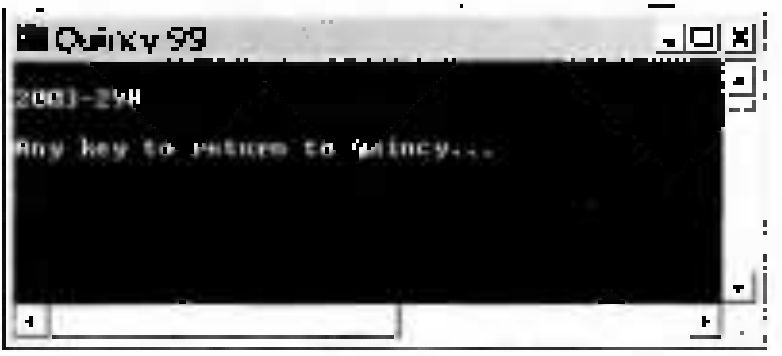
6.6 Using Friends

ယခင်အခန်းကဏ္ဍတွင်ကြည့်ခဲ့ကြသော class အပြင်ရှိ private member ဒေတာကို အခြား class member မဟုတ်သော function တစ်ခုမှ call ခေါ်ကာအသုံးပြုလိုပါကသော်လည်း အဆင်မပြေလုပ်ရန်အကြောင်း C++ တွင် friend function ကိုလာရောက်ကြည့်ရှုလေ့လာပါသည်။ friend function တစ်ခုကို define လုပ်နည်းမှာ non-member function တစ်ခုကိုလိုပင်ဖြစ်ပါသည်။ တစ်ခုသိသာသောကွာခြားချက်မှာ class declaration အတွင်း prototype function ရှေ့မှာနှစ်ခုတြိုး ထပ်ဆင့်ပြီး friend keyword ကိုထည့်သွင်းရေးသားခြင်းပါပဲ။ ယခုဇယားကဲ့သို့ Ex6017.cpp program ကိုကြည့်ပေးကြပါရန်တောင်းခံပါသည်။

```
// Listing 6.17: Using friends
#include <iostream>
class  Date;    // A forward reference

class  CustomDate
{
      int  da, yr;
  public:
      CustomDate (int  d = 0, int  y = 0)
            { mn = d; yr = y;}
      void  display( ) const
            { cout << end << yr << '-' << da;}

      friend  Date;
};


class  Date
{
      int  mo, da, yr;
  public
      Date (int mi, int d, int y)
            { mo = m; da = d; yr = y; }
      operator  CustomDate( );
};


Date  operator CustomDate( )
{
      static  int  days[] = {31,26,31,30,31,30,
                             31,31,30,31,30,31  },
      CustomDate  cd(0, yr);
      for (int i = 0; i < mo-1; i++)
            cd.da  += days[i];
      cd.da  += da;
      return cd;
}
```

```
int  main( )
{
        Date    dt(30,25,2003);
        CustomDate cd(dt);
        cd.display( );
        cout << endl;
        return 0;
}
```

Ex6b17.cpp program ကို run ကြည့်မယ်ဆိုရင် (ပုံ ၄၂) မှာပြထားတဲ့အတိုင်းတွေ့ရမှာပါ။



ပုံ (၄၂)

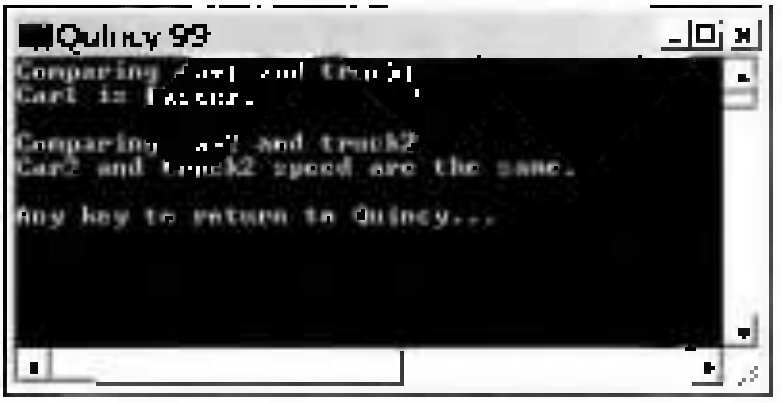
More on Using Friends

```
// Listing 6.18  More on using friends
#include <iostream.>
class  truck;    // A forward reference

class   car
{
        int   passenger, speed;
```

```
  public:
      car(int  p, int  s);
            { passenger = p; speed = s;}

      friend  int  spGreater (car  ca, truck  tr);
};


class  truck
{
      int  weight, speed;
  public:
      truck (int  w, int  s)
            { weight = w; speed = s; }
      friend  int   spGreater (car  ca, truck  tr);
};


int  spGreater (car  c, truck  t)
{
      return  c.speed - t.speed;
}


int  main( )
{
      car     car1(6,55), car2(2,75);
      truck   tr1(100,45), tr2(120,75);

      cout << "Comparing car1 and truck1\n";
      int  x = spGreater(car1,tr1);
      if (x < 0)
            cout << "Truck1 is faster\n";
      else if (x == 0)
            cout << "Car1 and truck1 speed are the same\n";
      else
            cout << "Car1 is faster\n";
```

```
    cout << "\nComparing car2 and truck2\n";
    x = spGreater(ca2,tr2);
    if (x < 0)
        cout << "Truck2 is faster.\n";
    else if (x == 0)
        cout << "Car2 and truck2 speed are the same.\n";
    else
        cout << "Car2 is faster.\n";

    return 0;
}
```

Ex6018.cpp ဖိုင်ထဲတွင်ပါရှိသော spGreater( ) function သည် class car နှင့်တွင် class truck တို့တစ်ခုမှတစ်ခုသို့ ပုံစံပြောင်း class ဖြစ်သော member များကိုအသုံးပြုနိုင်ရန် friend class ကိုကြေငြာထားခြင်း call ခေါ်ဆိုထားခြင်းတို့ ဖြစ်ပါသည်။ ယခု Ex6018.cpp program ကို run ကြည့်ပါမည်။



ပုံ (၆.၈)

၆.၀၀ Friend Functions

ယခင်ကဖော်ပြခဲ့သော class များတွင် သတ်မှတ်ထားသော class နှင့် friend ဆက်သွယ်ပေးနိုင်ခြင်း

[illegible] current class ရဲ့ member data [illegible] read/write လုပ်ခဲ့ [illegible] class [illegible] friend function [illegible] Exhibit 9 [illegible] program [illegible] friend function [illegible]

```
// Listing 6.19: Using friend functions
#include <iostream>

class  CustomDate;        // Forward reference

class   Date
{
        int   mo, da, yr;
  public:
        Date (const CustomDate&);    // conversion constructor

        void   display( ) const
               {cout << endl << mo << '/' << da << '/' << yr;}
};


class   CustomDate
{
        int   da, yr;
  public:
        CustomDate (int   d = 0, int  y = 0)
               { da = d, yr = y; }

        // Friend conversion function
        friend   Date::Date (const  CustomDate&);
};

// Conversion constructor: (Date <- CustomDate).
Date::Date(const CustomDate&   cd)
{
        static int   days[ ] = { 31,28,31,30,31,30,
                                 31,31,30,31,30,31   };
```

```
        yr = cd.yr;
        da = cd.da;

        for (mo = 0; mo < 11; mo++)
        {
                if (da > days[mo])
                        da  = days[mo];
                else
                        break;
        }
        mo++;
}


int main( )
{
        Date    dt (CustomDate(299, 2013));
        dt.display( );
        cout << endl;
        return 0;
}
```

Ex6019.cpp program ကို run ကြည့်မယ်ဆိုရင် ပုံ (ခ .၉) မှာကြည့်ပေးပါဗျာ

ပုံ (ခ .၉)

Chapter 7

multiple data item တွေကို group သုံးချင်တဲ့ structure လိုအပ်ချက်တွေအတွက် array ဟာ structure နဲ့တူပေမယ့် type မတူတဲ့ item တွေကို group လုပ်ခွင့်မပေးဘဲ structure ထဲမှာရဲ့ access လုပ်ချင်တဲ့ နှစ်သက်ရာကိုရွေးလုပ်ခွင့်ပေးပါတယ်။ array ဟာ integer index number တွေကိုသုံးပြီးရွေးနိုင်တဲ့ C++ ရဲ့အခြေခံဒေတာ array အမျိုးအစား int ဖြစ်ပြီး float ကိုကို simple type နဲ့ဖြစ်ပြီး group လုပ်ခွင့်ပေးနိုင်တဲ့ user-defined type အမျိုးအစား structure ဒါမှမဟုတ် class object တို့ဖြစ်ကြပါတယ်။ ဒေတာ variable နဲ့တူတဲ့ array သုံးခြင်းဥပမာတွေကို object တွေကိုလည်း array အနေနဲ့သုံးခြင်းဖြစ်စေ အကြီးစား သုံးသပ်ချက်အားလုံးကို အောက်ဆုံးအပိုင်းတွင်ကြည့်ရှုနိုင်ပါသည်။

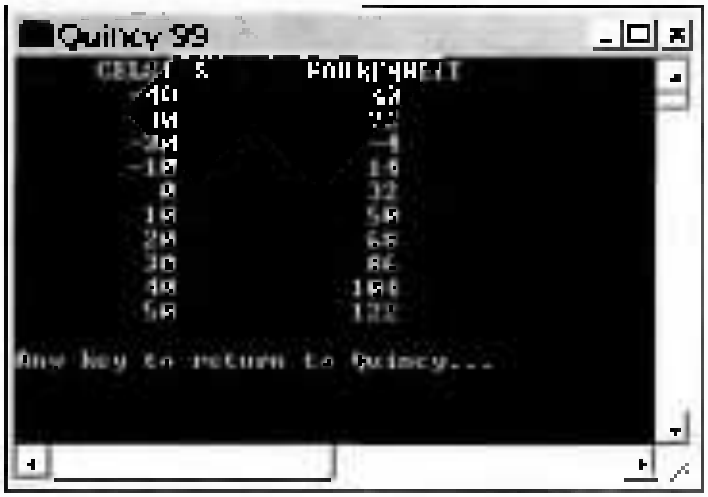
၇.၁ Array Basics

multiple data item တွေကို common name တစ်ခုတည်းနဲ့ကြေငြာသုံးစွဲနိုင်တဲ့ array ကို သုံးခြင်းဖြစ်ပြီး array ဆိုတာ တစ်ခုထက်ပိုတဲ့ variable တွေနဲ့ပြောင်းပြန်အားဖြင့် array name

size specification ဖြစ်ပေးရပါတယ်။ array ကိုကြေငြာတဲ့အခါ array name နောက်က square brackets [ ] ထဲမှာ non-negative integer (အနှုတ်တန်ဖိုးမဟုတ် သုညပါဝင်နိုင်ပါတယ်) ကိုရေးပေးရပါတယ်။ array ဖြစ်လာမယ်ဆိုရင် တစ်ခုချင်းစီ subscript တန်ဖိုးတွေပါတဲ့ one-dimensional array တစ်ခုကို ရည်ညွှန်းတဲ့အခါ square brackets ထဲမှာ subscript တန်ဖိုးတွေကိုထည့်ပေးရပါတယ်။ ဥပမာ myarray[10] ဆိုတာဟာ one-dimensional array တစ်ခုဖြစ်ပြီး array နာမည်က myarray ဖြစ်ကြောင်း subscript က (10) ခုရှိကြောင်း array element က (10) ခုမြောက် first element က myarray[0] ဖြစ်ပြီး last element က myarray[9] ဖြစ်ကြောင်းပါ။ two dimensional array တွေမှာ subscript (၂) ခုကို square brackets (2) နှစ်ခုတွဲသုံးပေးရပါတယ်။ ဥပမာ myarray[5][5] က two-dimensional array တစ်ခုဖြစ်ပြီး (၅×၅) အကွက်ရှိတယ်။ Ex10.1 .cpp program မှာတော့ array ကိုအသုံးပြုပြီး temperature conversion table တစ်ခုကို create လုပ်ပေးထားတာဖြစ်ပါတယ်။ ကျွန်တော်တို့

၇ (၁-၁)

1 program ကိုပေးလာတဲ့အခါမှာ ပထမဦးဆုံး fah ကို array ကိုကြေညာပေးပါတယ်။ ဒီ fah array ထဲက element (၉) ခုကိုဖြည့်ပါတယ်။ for loop ရဲ့ ပုံစံကတော့ cel = -40 ကနေ cel = 40 အထိ (၉) ခု ထိပါပဲ။ ဒီထဲမှာကြည့်မယ်ဆိုရင် fahrenheit ပြောင်းသည့်နည်းကိုပြောင်းပေးထားပါတယ်။ ဒါကြောင့် fahrenheit ထဲက ဂဏန်းတွေကို fah[0], fah[1], fah[2] အစရှိတာတွေထဲမှာ သိမ်းသွားတဲ့အတွက် ပရိုဂရမ်ကြီးပြီးသွားတဲ့အခါ cel ကသုညကနေ စတင်ပြီးတော့ cel = 40 ဖြစ်သွားတာကိုတွေ့ရမှာပါ။ Ex07-1.cpp program ကို ကွန်ပျူတာမှာ run ပြီးသွားပါပြီ။



ပုံ ၇-၂

၇.၂ Initializing an Array

C++ program တွေမှာ array တစ်ခုချင်းစီတန်ဖိုးတွေကို initialize လုပ်ပေးရင် array ပတ်ဝန်းကျင်က subscript အရေအတွက်အလိုက်၊ အမှားမဖြစ်အောင်ဖြစ်ပါတယ်။ (comma ခြားပြီး) ထည့်ပေးရပါတယ်။ data တွေကို brace တွေနဲ့ပိတ်ပေးရပါတယ်။

```
float   x [5] = { 0, 1, 0.25,  5, 1.0, 7.5 };
char    color [3] = { 'R', 'E', 'D' };
```

[illegible] initial assignment [illegible]

```
x [0] = 0          color [0] = 'R'
x [1] = 1.0        color [1] = 'E'
x [2] = 0.25       color [2] = 'D'
x [3] = -5.0
x [4] = 1.0
x [5] = 7.5
```

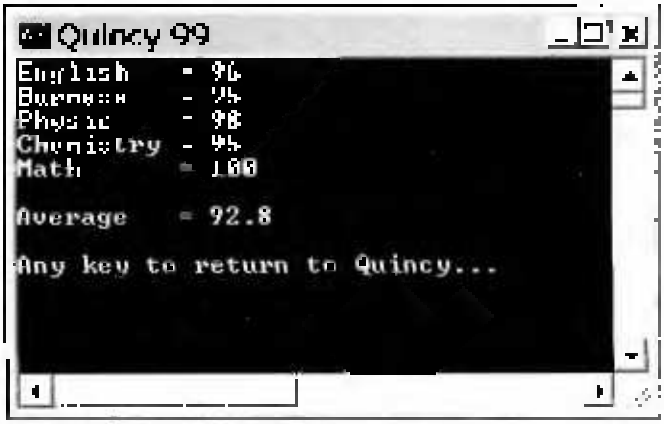
[illegible] Ex702.cpp program [illegible] array subject [illegible] initialize [illegible]

Ex702.cpp

```
// Array 7.2 This program averages a student's marks.
#include <iostream>

const int SIZE = 5;

int main()
{
    int    subj[SIZE] = {90.75,96,98.5,95};
    float total=0;

    cout << "English   = " << subj[0] << endl
         << "Burmese   = " << subj[1] << endl
         << "Physics   = " << subj[2] << endl
         << "Chemistry = " << subj[3] << endl
         << "Math      = " << subj[4] << endl;
    for (int i = 0; i < 5; i++)
        total += subj[i];
    float avg = total / SIZE;
    cout << "\nAverage = " << avg << endl;

    return 0;
}
```

[illegible]

Ex702.cpp program ကိုလေ့လာကြည့်ပါဆိုရင် sub[] ကို array ကိန်းဂဏန်းတွေကြိုပြီး initialize လုပ်ပေးထားတယ်။ array size ကို SIZE - 5 ဖြစ်ကြောင်းသတ်မှတ်ပေး၊ average ရှာပေးတဲ့ float avg = total/SIZE; statement ကလေးတွေကတော့ avg ကို display ထုတ်ပြီးတော့ အလုပ်လုပ်သွားမှာပါ။ ပုံ (၂-၄) မှာ Ex702.cpp ကို run ပြထားပါတယ်။



ပုံ (၂-၄)

Substituting Strings

ဒီတစ်ခါတော့ Ex703.cpp program မှာ char array တွေဖြစ်တဲ့ str1 နဲ့ str2 တို့ကို initialize ပေးရုံတင်မကဘဲ ပြောင်းလဲပေးတဲ့ display ပုံစံတွေကိုလေ့လာပါမယ်။

```
// Listing 7.3: This program shows how to print character
// strings that involve various kind of substitutions.

#include <iostream>

int  main( )
{
        char    str1[ ] = { "BASKET" };
        char    str2[ ] = { 'B', 'A', 'L', 'L', '\0' };
```

```
        cout << "The goal in " << str1 << " "
             << str2 << " is " << endl;
        cout << "to put the " << st-2 << " in the "
             << str1 << endl;
        return 0;
}
```

Ex703.cpp program ကို run လိုက်ရင်အဖြစ်မှာ ပုံ (၇-၇) မှာကြည့်ရသလိုဖြစ်ပါမယ်။

ပုံ (၇-၇)

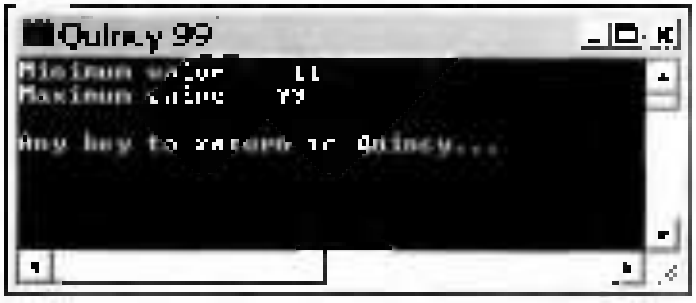
Finding the Maximum and Minimum Values

ဒီ ဥပမာတစ်ခုအတွက်ကြည့်ရအောင် Ex704.cpp program မှာ array တစ်ခုထဲမှာ minimum/maximum value နှစ်ခုကိုရှာဖြေထားပါတယ်။ ဆက်လေ့လာကြည့်ပါ။

```
// Listing 7.4: This program checks a set of numbers
// and reports the minimum and maximum values found.
#include <iostream>

int     MIN (int  a, int  b)  { return (a < b) ? a : b; }

int     MAX (int  a, int  b)  { return (a > b) ? a : b; }
```

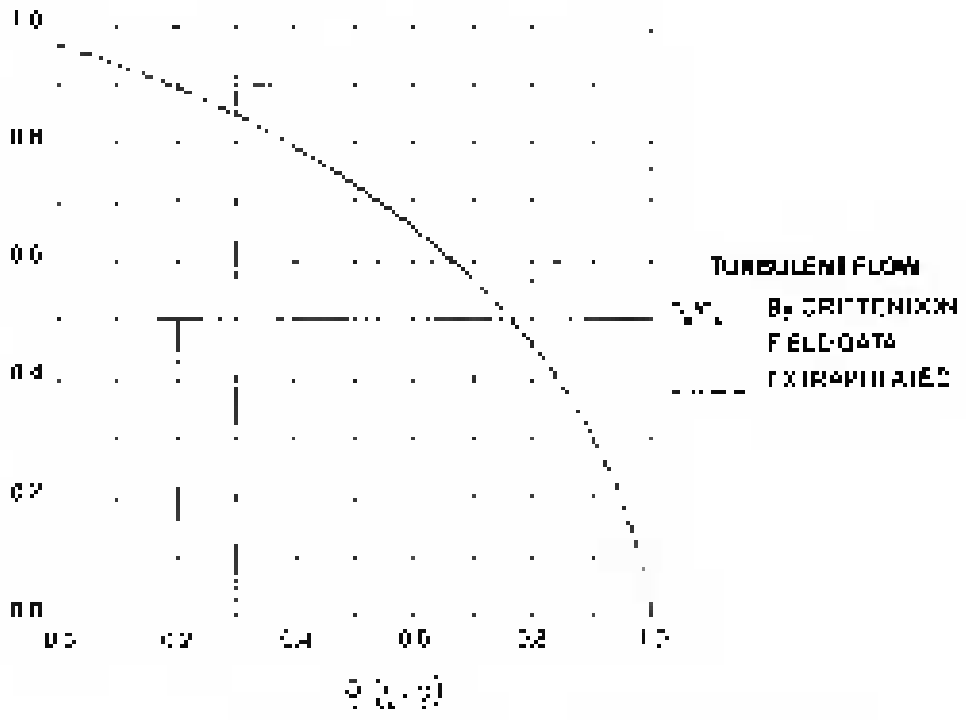
```
const int  N = 25;

int main( )
{
        int     minval, maxval;
        int     val[ ] = {
                        20,  11,  13,  41,  55,
                        75,  45,   2,  84,  -3,
                         7,  -9,  32,  16,  54,
                        82,  51,  99,  -3,  77,
                       -11,  71,  29,  33,  98   };

        minval = maxval = val[0];
        for ( int i = 1; i < N; ++i)
        {
                minval = MIN (minval, val[i]);
                maxval = MAX (maxval, val[i]);
        }
        cout << "Minimum value = " << minval << endl;
        cout << "Maximum value = " << maxval << endl;
        return 0;
}
```

.F E:704.cpp program ကို run ကြည့်မယ်ဆိုရင် ၃ (၃. ၆) ပုံကဲ့သို့ပေါ်လာမှာဖြစ်ပါတယ်။



ပုံ (၃. ၆)

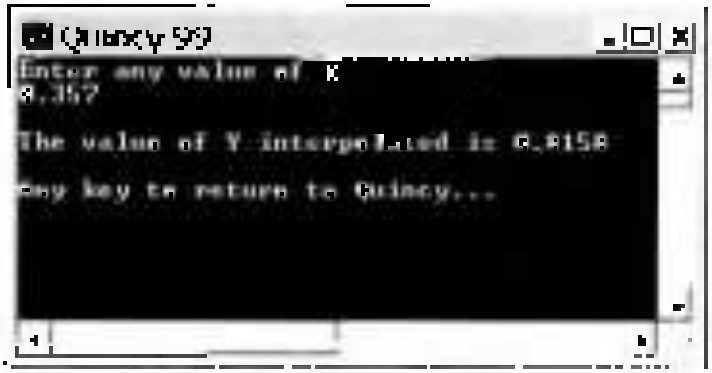
Processing an Array




```
// Listing 7.5:  This program interpolates the value of y for any value of x
// from a given data array read from the graph

#include <iostream>

int main( )
{
        float   x[ ]= { 0, 1..2,.3,.4, 5, 6, 7, 8..9,1.0 },
        float   y[ ]= { .97, 94,.89,.84,.79,.73..64..56,.46..3,0 };
        float  xnow, yval;
```

```
        cout << "Enter any value of X\n";
        cin >> anyx;
        if ( anyx <= x [0] )
                yval = y [0];
        else if( anyx >= x [10])
                yval = y [10];
        else
                for(int i= 1; i < 10; ++i )
                        if ( x [i] > anyx )
                        {
                                yval = y[i-1] + (anyx - x[i-1])* (y[i] - y[i-1])/
                                        (x[i] - x[i-1]);
                                break;
                        }
        cout << "\nThe value of Y interpolated is " << yval << endl;
        return 0;
}
```

Ex705.cpp program ရဲ့ output ကို ကြည့်ရအောင်။ graph ကို ကြည့်ချင်ရင် program အဆုံးမှာ X အဖြစ်ထားခဲ့တဲ့ data တွေကို ကွန်ပျူတာထဲကို Y ကိုဖြစ်စေဖို့ အမြန်ဆုံးနည်းလမ်းက graph ကိုကူးယူပြီး word processor ထဲမှာ paste လုပ်ပေးလိုက်ခြင်းပဲ ဖြစ်ပါတယ်။ program ထဲမှာ graphics ဆွဲတဲ့ code တွေ ထည့်ထားခြင်း မရှိသော်လည်း ကွန်ပျူတာပေါ်မှာ graph ကို ကြည့်ချင်ရင် ဒီလိုကူးယူနိုင်ပါတယ်။ ဥပမာအားဖြင့် linux လုပ်ငန်းမှာ Ex705.cpp program ကို run ပြီး output ကို (x, y) တန်ဖိုးတွေအတိုင်း ရိုက်ထည့်လိုက်ပါ။



ပုံ (၇-၁)

Generating a Pascal triangle

[illegible] Ex05.cpp program [illegible] array [illegible] Pascal triangle [illegible] program [illegible]

```
// Listing 7.6: This program generates a Pascal triangle of numbers.

#include <iostream>
#include <iomanip>

const int  MAXROW = 13;

int main( )
{
        unsigned int   lead_sp = 3*MAXROW;
        unsigned int   row[MAXROW];

        row[0] = 1;
        for (int n=1; n<MAXROW; n++)
                row[n] = 0;

        for (int row_no=1; ; row_no++)
        {
                for (int  i=1; i<= lead_sp; i++)
                        cout << " ";
                lead_sp -= 3;

                for (int j=0; j<row_no; j++)
                        cout << setw(5) << row[j];
                cout << endl;
                if (row_no == MAXROW)   break;
                for (int k=row_no; k>=1; --k)
                        row[k] += row[k-1];
        }
```

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

Ex706.cpp program ကို run ပြီးသွားရင် ၄ (၃. ၉) ကဲ့သို့ output ထွက်လာမှာဖြစ်ပါတယ်။ ထည့်သွင်းပေးလိုက်တဲ့ program ကို trace လုပ်ကြည့်ပါ။

၄ (၃. ၉)

၇.၄ Passing Array to a Function

ပေးပို့မယ့် function ထဲကို array တခု pass လုပ်ကြည့်ကြရအောင်။ အလွယ်ဆုံးနည်းလမ်းကတော့ function definition မှာ array name – [ ] ကို argument အဖြစ်ကြေညာပေးလိုက်ခြင်းပါပဲ။ ဒုတိယနည်းကတော့ argument ဖြစ်တဲ့ array name မှာ square bracket [ ] အလွတ်ကြီးချန်ထားခဲ့ပါ။ သုံးသပ်ချက်အရ ပထမနည်းလမ်းကို function call ခေါ်တဲ့အခါမှာတော့ argument ဘဲ့ array name ထဲမှာ ဘယ်လို square bracket [ ] မျိုးမှ မပေးရတာကိုတွေ့ရမှာပါ။ Ex707.cpp program ကိုကြည့်ကြည့်ပါ။

```
// Listing 7.7: This program passes a three-element integer array
// to a function where the array elements are altered.

# include <iostream>

const  int  SIZE = 3;

void  modify(int  x[ ])
{
      cout << "\nInside FUNCTION modify( )\n";
      for ( int i=0; i<SIZE; ++i)
      {
            x [i] =2* x [i];
            cout << "\tx[" << i+1 <<"] = " << x[i] << endl;
      }
}

int main( )
{
      int       i, x[SIZE];
      void     modify (int   x[ ]);

      cout << "From MAIN before calling the function\n";
      for ( i= 0; i < SIZE; ++i)
      {
            x [i] = i+1;
            cout << "\tx[" << i+1 <<"] = " << x[i] << endl;
      }
      modify(x);
      cout << "\nFrom MAIN after the function is called\n";
      for ( i= 0; i < SIZE; ++i)
      {
            x[i] = i+1;
            cout << "\tx[" << i+1 <<"] = " << x[i] << endl;
      }
      return 0;
}
```

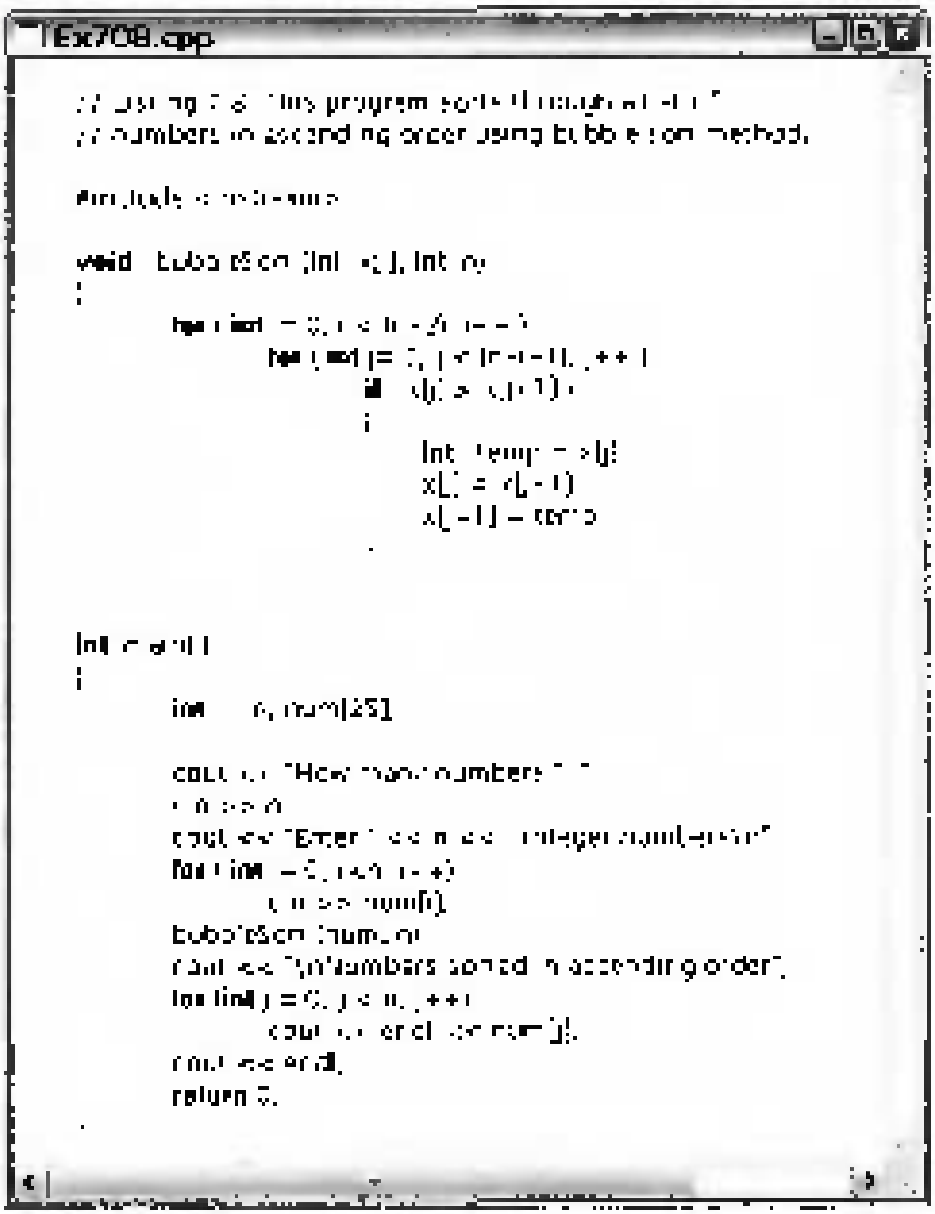
ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင် main( ) function ထဲကလည်း function modify ထဲကိုရောက်သွားတဲ့ array element တွေဟာ modify( ) function ထဲမှာ အပြောင်းအလဲဖြစ်ခဲ့ပြီးတာနဲ့ main( ) ထဲကိုပြန်ရောက်လာတဲ့အခါမှာ array element တွေရဲ့တန်ဖိုးတွေက အသစ်ပြောင်းသွားတာကိုတွေ့ရမှာပါ။ ဒါကတော့ array ကိုသုံးပြီးတော့ function တစ်ခုကနေ တစ်ခုကို data pass ပေးဝေနည်းပါပဲ။ Ex707.cpp program ကို run ကြည့်မယ်ဆိုရင် ဒုံ (၂) ခု မျှပြသပေးမှာဖြစ်ပါတယ်။

ဒုံ (၂) ခု

Sorting

ဒုံ (၂.၁) မှာတင်ပြခဲ့တဲ့ Ex708.cpp program ကတော့ array အတွက်ပြုလုပ်ထားတဲ့ bubble sorting လုပ်နည်းကိုရေးထားတဲ့ program ပါပဲ။ လေ့လာကြည့်ပါ။ ဒီ program ရဲ့ရည်ရွယ်ချက်ကတော့

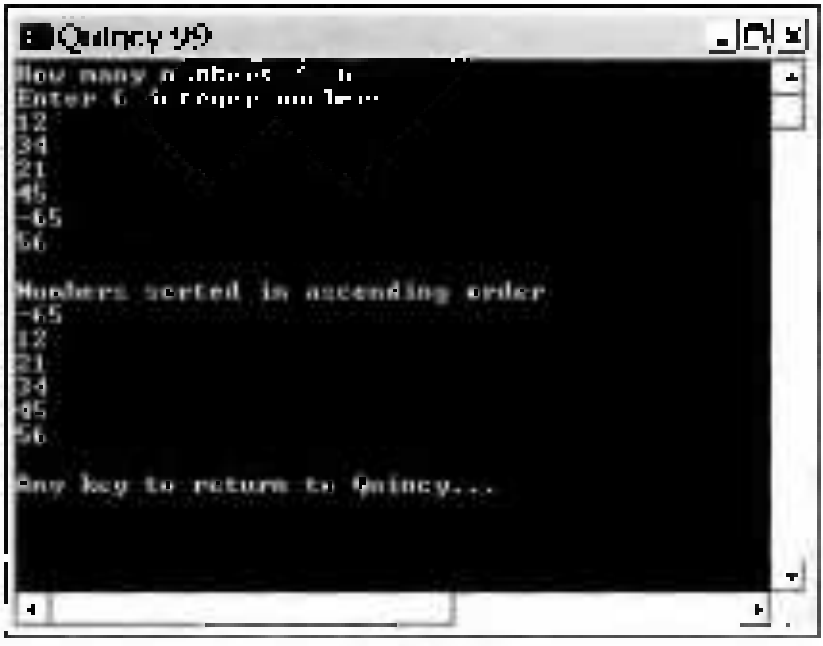
- ဥပမာအနေနဲ့ ပေးဝေနိုင်တဲ့နံပါတ်စီးရီးတွေအနေနဲ့ပြောရမယ်ဆိုရင် ကော်ဇောတစ်ထပ်မှာ (25) ပဲကျိုးမယ်ဆိုပါစို့။ ဒီအခါမှာ ကွန်ပျူတာက How many numbers ? ဆိုပြီး prompt ပေးလာမယ်ဆိုရင် (6) ကို ရိုက်ထည့်လိုက်မယ်ဆိုရင် n = 6 ဖြစ်သွားပါပြီ။ n ဟာ sort လုပ်ဖို့ ကျန်းမာရေးအလွှာဦးရေပေါ့။



ပုံ (၇-၁၀)

■ ဒုတိယအနေဖြင့် (5) ခုပါဝင်သော data တော်တော်များများ 12 34 21 45 65 56 ကဲ့သို့သော ကိန်းဂဏန်းများကို ascending order ဖြင့် sort လုပ်ပေးနိုင်ရန် BubbleSort( ) function ကို call ခေါ်ပေးသော function argument များသော num နှင့် n ဖြစ်သော num[ ] ကို ရေးချရန်လိုအပ်ပေးပါမည်။

- bubbleSort( ) function ကူးယူသုံးစွဲသော argument ပေးပုံ (int x[ ], int n) သတိပြုရန်
  ဖြစ်သော array name (x) အတွက် dummy argument ကို int x[ ] ဖြင့်ဖော်ပြ
  ကြောင်းနှင့် function body တွင်ပါဝင်သော numerical method ကို
  မှတ်သားထားရန်။ Ex708.cpp program ကို run ကြည့်ခြင်းဖြင့် ရလဒ်ကို
  သေချာစွာလေ့လာပါ။



(Ex708)

Counting Frequencies

- Ex708.cpp program တစ်ခုလုံးကို လေ့လာပြီးနောက် keyboard မှရိုက်သွင်းလိုက်သော character
  များတွင် digit ဘယ်နှစ်လုံး ၊ white space ဘယ်နှစ် ၊ non-white space ဘယ်နှစ်ခုပါသည်ကို
  ရေတွက်ပေးလိုပါသည်။

```
// Listing 7.6. This program keeps count of all white spaces,
// non-white spaces and the frequencies of digits 1 through 9.

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

const int   ESC = 27;
const int   N=10;

int main( )
{
        int   i, wh_spcs = 0, ch;
        int   nonwh_spcs = 0;
        int   digit[N];

        for  (i=0; i<N; i++)  digit[i] = 0;

        cout << "Enter a line of characters\n";
        while ((ch=getche( )) != ESC)
        {
                if (ch >= '0'  &&  ch <= '9')
                        digit[ ch -'0' ]++;
                else if (ch == ' ')
                        wh_spcs++;
                else
                        nonwh_spcs++;
        }
        cout << endl << endl;

        for ( i = 0; i<N; i++ )
                cout << "digit[" << i << "] = " << digit[i] << endl;
        cout << "\nWhite spaces = " << wh_spcs
             << "\nNon-white spaces = " << nonwh_spcs << endl;

        return 0;
}
```

- ပထမဆုံး main( ) function ထဲမှာ digit[10] နဲ့ nonDigit[0] ဆိုတဲ့ array variable ၂ ခုကို ကြေငြာပြီး initialize လုပ်ပေးထားပါတယ်။ ပြီးတော့ std C function getche( ) ကို အသုံးပြုပြီး character variable ch ထဲကို စာလုံးတစ်လုံးချင်း ဖတ်ယူထားပါတယ်။ ch ထဲကို ဖတ်ယူလိုက်တဲ့ စာလုံးက ESC (27) မဟုတ်သရွေ့ while loop ကနေ ဆက်လက်ပြီး ဖတ်ယူနေမှာဖြစ်ပါတယ်။ ဒီ while loop ထဲမှာ ရိုက်ထည့်လိုက်တဲ့ စာလုံးက 0 နဲ့ 9 ကြားမှာ ရှိမရှိကို ဦးစွာ while loop ထဲမှာ ဆုံးဖြတ်ပြီး ရှိခဲ့ရင် ch ထဲကနေ ဂဏန်းကို ပြောင်းလဲယူပြီး digit[] array ထဲကို ထည့်သွင်းပေးပါတယ်။ ဥပမာ ch ထဲကို '1' လို့ ရိုက်ထည့်လိုက်ရင် digit[1] = 1 + 1 = 2 ဖြစ်သွားမှာပါ။ ဒီနောက်မှာ ထပ်ဆင့်ပြီး ch ဟာ white space တစ်ခုခုဖြစ်ခဲ့ရင် wh spa variable ကို တစ်တိုးမြှင့်ပေးပြီး ch ဟာ digit မဟုတ်သလို white space လည်း မဟုတ်ခဲ့ရင် nonwh spa variable ကို တစ်တိုးမြှင့်ပေးထားပါတယ်။
- ကွန်ပျူတာကို run လိုက်တဲ့အခါ ကီးဘုတ်ကနေ Escape key ကိုနှိပ်လိုက်မှသာ Ex708.cpp program ကို run လုပ်ထားတာကို ရပ်စဲပေးနိုင်မှာဖြစ်ပါတယ်။

ပုံ (၃-၁)

Multidimensional Arrays

multidimensional array ဆိုတာကတော့ one-dimensional array တစ်ခုကို ကျွန်တော်တို့ ပြောခဲ့ကတဲ့ dimension တွေတွေနဲ့ square bracket [ ] တွေ ပေါင်းထည့်ပေးရုံပါပဲ။ two-dimensional array ဆိုရင် square bracket [2] နဲ့ three-dimensional array ဆိုရင် [3] ပေါ့။ ဒီလိုမျိုး ကျွန်တော်တို့ လိုသလောက် multidimensional array တွေကိုလည်း ကြေညာနိုင်ပါတယ်။

```
float     table [50] [50];
char      page [25] [80];
double    elements [L] [M] [N];
```

multidimensional array တစ်ခုရဲ့ element တွေကို initialize လုပ်တဲ့အခါမှာ အောက်ပါအတိုင်းလုပ်နိုင်ပါတယ်။

```
int     x [3] [4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

ဒီဥပမာအရဆိုရင် x လို့ array ရဲ့ name ဟာ [3] rows နဲ့ [4] columns ပါတဲ့ two-dimensional array တစ်ခုကို ဆိုလိုတာပါ။ ဒီ array element တွေကို တွန်းကောက်ကနေ assign လုပ်သွားတဲ့အတိုင်းပါပဲ။

```
x [0][0] = 1      x [0][1] = 2      x [0][2] = 3      x [0][3] = 4
x [1][0] = 5      x [1][1] = 6      x [1][2] = 7      x [1][3] = 8
x [2][0] = 9      x [2][1] = 10     x [2][2] = 11     x [2][3] = 12
```

ဒီ array ကို အောက်ပါအတိုင်းလည်း ကြေညာနိုင်ပါတယ်။ အဲဒီမှာတော့ row နဲ့ column တွေကို ခွဲခြားပေးထားတယ်။

```
int     x [3][4] = {
                    { 1, 2, 3, 4 },
                    { 5, 6, 7, 8 },
                    { 9, 10, 11, 12 },      };
```

Ex705.cpp

```
// Listing 7.9  This program displays students
// marks (their margins vary)

#include <iostream>
#include <iomanip>

const int  STUDENTS = 4;      // Array dimensions
const int  SUBJECTS = 5;

int main()
{
      int  marks[STUDENTS][SUBJECTS] =
                                    {100.75,85,70,85},
                                    { 75,91,56.50,70},
                                    { 45,50,55,75.50},
                                    { 20,40,36.48,45}
      };

      cout << "      ENG  BUS  PHY  CHEM  MATH"
      for(int i=0; i<STUDENTS; i++)
      {
            cout << "STUDENT" << i+1
            for(int j=0; j<SUBJECTS; j++)
                  cout << setw(5) << marks[i][j]
      }
      cout << endl;
      return 0
}
```

ပုံ (၇-၁၄)

ပုံ (၇-၁၄) မှာပြထားတဲ့ Ex705.cpp program ကိုရေးသားထည့်သွင်းကြည့်ပါ။

- ဥပမာ- marks ဟုခေါ်တဲ့ two dimensional array အစုအဝေးကို initialize လုပ်ပါတယ်။ row နဲ့ column dimension သတ်မှတ်ပေးထားပါတယ်။ ပြီးတော့ heading ကို display လုပ်ဖို့အတွက် nested for loop ကိုသုံး row (4) နဲ့ column (5) ကိုဖတ်ဖို့ ပြီးတော့ program ကိုရေးသားပါ။

■ Ex709.cpp program ကို run လိုက်ရင်အခြေခံ ၄ ခု (၃၀ ဦး) ဘွဲ့သင်တန်းတက်ရောက်သူတို့ရဲ့ရလဒ်ကို

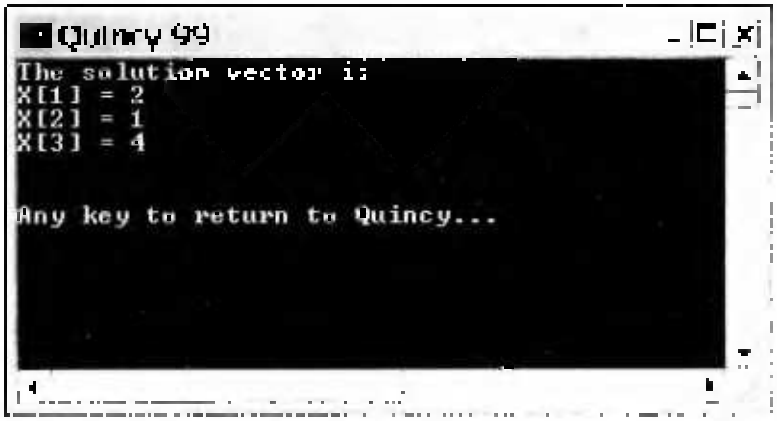
ပုံ (၇-၁၀)

Solving Simultaneous Equations

```
// Listing 7.10: This program solves the following
// simultaneous equations

//                    6x + 2y + 3z = 30
//                    x  - 9y + 2z = 1
//                    2x + 3y + 8z = 31

// using the Gauss Seidel iterative method

#include <iostream>
#include <cmath>

int main()
{
      int  i, j, n=3, m;
```

```
// This is coefficient matrix
double   c[][3] = {
                        { 9, 2, 3 },
                        { 1, -9, 2 },
                        { 2, 3, 6 },
};

// Right-hand side vector
double     r[ ] = { 31, 1, 21 },

// Assume solution vector
double   x[ ] = { 1, 1, 1 },

double  temp,

do {
        m = 0;
        for ( i=0; i<n; i++ )
        {
                temp = r[i];
                for ( j=0; j<n; ++j )
                        if ( i != j )
                                temp -= x[j] * c[i][j];
                temp /= c[i][i];
                if (fabs(temp - x[i]) > 1e-7 ) m++;
                x[i] = temp;
        }
} while (m != 0);

// Print output result
cout << "The solution vector is\n";
for ( i=0; i<n; ++i )
        cout << "x[" << i << "] = "
             << x[i] << endl;
cout << endl;
return 0;
}
```

■ Ex7010.cpp program ကို run လိုက်လျှင်အောက်ပါ ပုံ (၃.၆) အတိုင်း ရရှိမည်ဖြစ်ပါသည်။



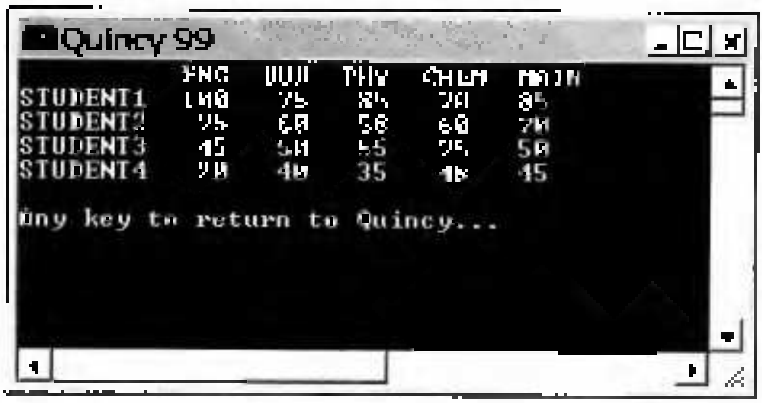
ပုံ (၃.၆)

Passing Multidimensional Arrays to Functions

```
// Listing 7.11: This program displays students vs
// their marks chart.

#include <iostream>
#include <iomanip>

const int  STUDENTS = 4;
const int  SUBJECTS = 5;

void  display (int  marks[STUDENTS][SUBJECTS])
{
      cout << "        ENG  BUR  PHY  CHEM  MATH";
      for (int  i = 0; i < STUDENTS; i++)
      {
            cout << "\nSTUDENT" << i+1;
```

```
            for (int  j =0; j < SUBJECTS; j++)
                  cout << setw(5) << marks[i][j];
      }
}

int main( )
{
      int    marks [STUDENTS][SUBJECTS] =
            {
                  {100,75,85,70,85},
                  { 75,60,58,60,70},
                  { 45,50,55,75,50},
                  { 20,40,35,48,45}
            };
      display(marks);
      cout << endl;
      return 0;
}
```

■ Ex7011.cpp program ကို run လိုက်ပေးခြင်းဖြင့် ပုံ (၇-၈) ကဲ့သို့ ရရှိကြောင်းတွေ့ရှိနိုင်ပါသည်။



ပုံ (၇-၈)

Arrays of Objects

ဒီနည်း data type တွေလို array ဟာလည်း multiple data တွေကနေ store လုပ်ဖို့အတွက် object တွေကိုလည်း array အဖြစ် သိမ်းဆည်းနိုင်ပါတယ်။ ဥပမာအားဖြင့် Listing 7.12 မှာ array ကိုအသုံးပြုပြီး object (15) ခုတွေနဲ့ Celsius ကနေ Fahrenheit ကို temperature conversion လုပ်ပေးထားပါတယ်။

```
// Listing 7.12: Arrays of objects

# include <iostream>
const int  N=15;

class  temps
{
      int  cel;
  public:
      int  setCel(int  n)
            {
                  cel = 40 + n*10;
                  return cel;
            }

      float getFah( )
            { return  1.8*cel+32; }
};


int  main( )
{
      temps  obj[N];

      cout << "  ----------      \n";
      cout << "    CELSIUS  FAHRENHEIT\n";
      cout << "  ----------  ----\n";
```

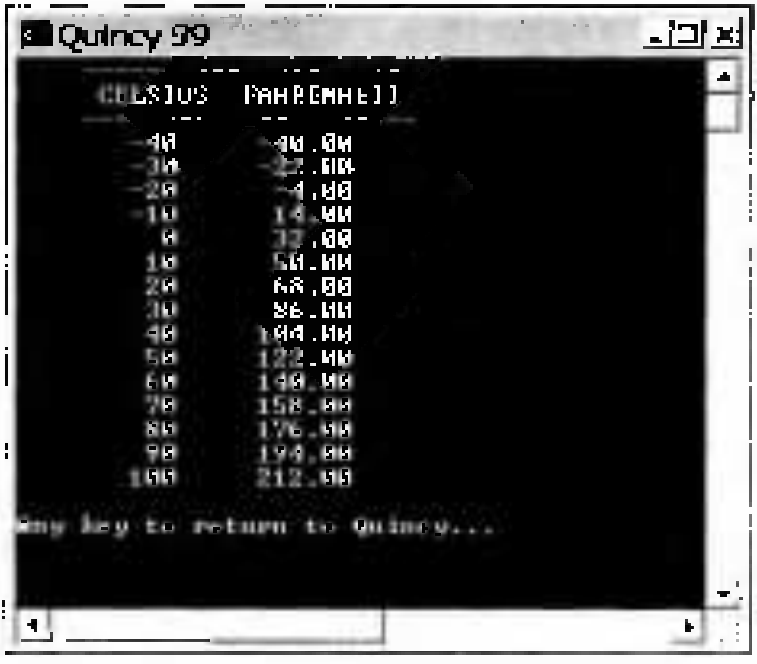
```
    for (int  i=0; i<N; i++)
    {
            cout.setf(ios::fixed);
            cout.width(10);
            cout.precision(2);
            cout << obj[i].setCel();
            cout.width(11);
            cout << obj[i].getFah( );
            cout << endl;
    }
    return 0;
}
```

Ex7012.cpp program ကုတ်ကို ရှင်းပြခြင်း

- ပထမဦးစွာ template class object တွေကို array object [N] နဲ့ကို construct လုပ်ပြီး ထပ်ပြီး ပြောင်းတဲ့ for loop ထဲတော့ obj[i] ဆိုတဲ့ celsius အနိမ့်ဆုံးကို setCel( ) call ပြီးမှ set လုပ်ထားတဲ့အတွက် ပထမဆုံး i=0 အတွက် cel = -40+ i*10 = -40+0*10 = -40 ဖြစ်သွားတဲ့အတွက် main( ) function ကိုပြန်ရောက်တော့ obj[0].setCel(0) = -40 ဖြစ်သွားပါ
- obj [0] ရဲ့ celsius အနိမ့်ဆုံးကို fahrenheit ပြောင်းပေးဖို့တွက် getFah ( ) function ကို call သောအခါ ကျွန်တော်တို့ fah = 1.8*cel+32 = 1.8* (-40) + 32 = -40 ဖြစ်သွားပြန်ပါတယ် ပြီးမှ main ( ) function ဆီကို obj[0].getFah ( ) = -40 ကို return ပြန်ပေးလိုက်ပါတယ် for loop ကို (15) ကြိမ်ပတ်ပြီးတဲ့အခါ celsius vs Fahrenheit table ကို ထုတ်ပေးသွားမှာဖြစ်ပါတယ် Ex7012.cpp program ကို run ကြည့်ပြီး ရှင်းပြပုံ အပြည့်အစုံကို ကိုယ်တိုင်ပြန်လေ့လာကြည့်ပါဦးနော်

More on Arrays of Objects

```
// Listing 7.13: More on arrays of objects
#include <iostream>
```

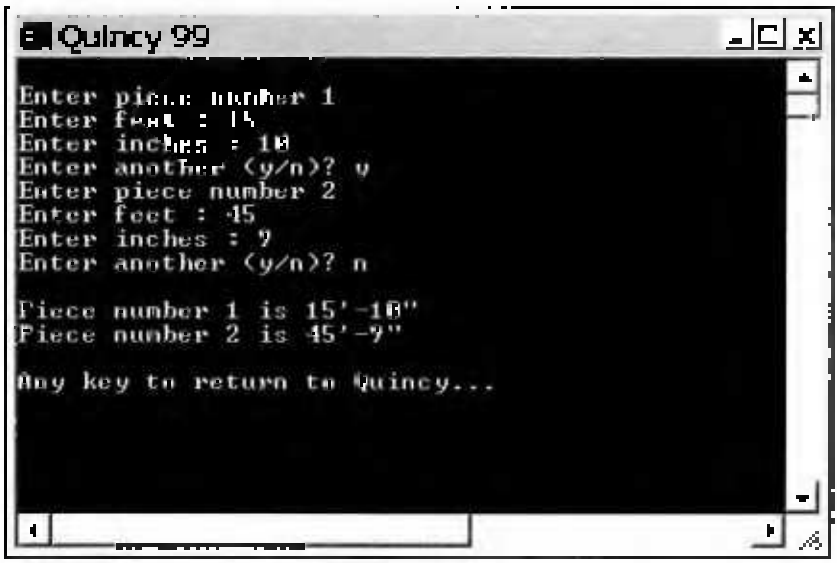


((2, aa)

```
const int  MAX = 50;

class  Length
{
      int     feet;
      float   inches;
  public:
      void  getLength( )
            {
                  cout << "\nEnter feet : ";   cin >> feet;
                  cout << "Enter inches : ";   cin >> inches;
            }
      void  showLength( )
            { cout << feet << "'-" << inches << '\"';  }
};
```

```
int main( )
{
        Length   piece[MAX];
        int      count = 0;
        char     code;
        cout << endl;
        do {
                cout << "Enter piece number " << count+1;
                piece[count++].getLength( );
                cout << "Enter another (y/n)? ";    cin >> code;
        } while (code != 'n');

        for (int  i=0; i<count; i++) {
                cout << "\nPiece number " << i+1 << " is ";
                piece[i].showLength( );
        }
        cout << endl;
        return 0;
}
```

Ex7013.cpp program ကို run ခ်ိန္ၿပီးဆိုရင္ ပံု (၇-၇၉) ကဲ့သို့ေတြ့ရမွာျဖစ္ပါတယ္။



ပံု (၇-၇၉)

Initializing Arrays of Objects

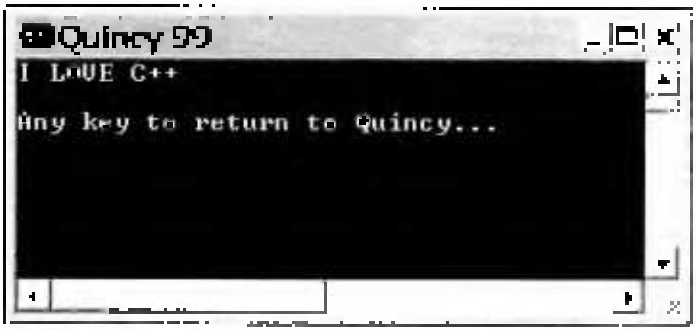
ဒုတိယ ဥပမာ ဖြစ်တဲ့ Ex7014.cpp program မှာ type class letters ဖြစ်တဲ့ array object တစ်ခုကို initialize လုပ်ထားတဲ့ character တွေကို uppercase letter ပြောင်းလဲပေးနိုင်တဲ့ program ဖြစ်ပါတယ်။ object obj[ ] မှာ I love C++ ဆိုတဲ့ character (10) ခုကို ch နဲ့ initialize လုပ်ထားသေးပါတယ်။ ပါရာကြွင်း obj[i].get_ch( ) ကို display လုပ်နိုင်ဖတ်နဲ့ array object တစ်ခုချင်းစီကနေတစ်ဆင့် ch ပစ်နှင့်ကို uppercase ပြောင်းပြီးပြန်ပေးပါမယ်။ toupper( ) function ကိုသုံးသပ်ပြုပြုနေအောင် <cctype> header ဖိုင်ကို program မှာ include လုပ်ထားပေးပါမယ်။

Ex7014.cpp

```
// Listing 7.14: This program changes a line
// of characters to uppercase.

#include <iostream>
#include <cctype>

class  Letters
{
        char  ch;
public
        Letters (char  c)
                { ch = c; }
        char  get_ch( )
                { return   toupper(ch); }
};

int  main( )
{
        Letters  obj[10]    { 'I', ' ', 'l', 'o', 'v',
                              'e', ' ', 'C', '+', '+' }
                            };
        for (int i=0; i<10; i++)
                cout << obj[i].get_ch( );
        cout << endl;
        return 0;
}
```

ဒုတိယ ဥပမာ

၂။ Ex7014.cpp program ကို run ထိုက်ပေးခြင်းဖြင့် ပုံ (၃-၂၀) မျက်ကားပြတ်ကားကိုမြင်ရမှာဖြစ်ပါတယ်။



ပုံ (၃-၂၀)

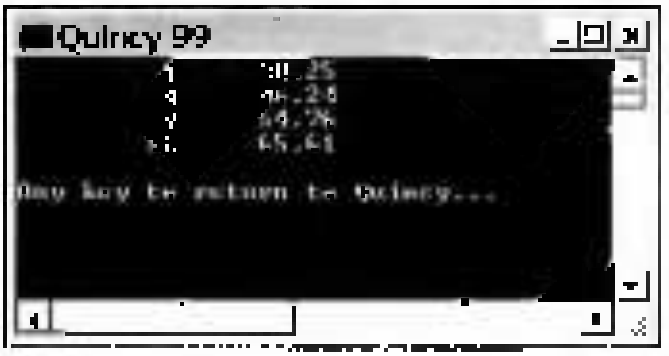
Initializing Multidimensional Arrays of Objects

၁။ Ex7015.cpp program: သည် two-dimensional array object obj[ ] အစုမှ initialize လုပ်ထားတဲ့ element တစ်ခုချင်းစီနှစ်ခေါက်စီးပေးတဲ့ program ပါပဲ။ getSquare( ) function ကိုကသုံးပြီးတော့ obj[0][0] နဲ့ obj[0][1] သိုမဟုတ် (1, 5) element (2) ခုကိုနှစ်ထပ်ကိန်းကင်းမြှင့် display လုပ်ပြပါလိမ့်မယ်။ for loop ကို (၂) ကြိမ်ပတ်ခိုင်းထားတဲ့အတွက် ကျနော်တို့ object (8) ခုမှာ element တွေကိုလည်း square လုပ်ပေးမှာပါ။ စလုပ်ကြည့်ပါ။

```
// Listing 7.15: Initializing the multidimensional arrays of objects
#include <iostream>
#include <iomanip>

const int   ROW=4;
const int   COL=2;

class   Asqr
{
        float   a;
```

```
public:
        Asqr (float  x)
                { a = x; }
        float getSquare( )
                { return  a*a; }
};

int main( )
{
        Asqr  obj[ROW][COL]= {
                                {1, 5.5},
                                {2, 6.6},
                                {3, 7.4},
                                {4, 8.1}
                             };

        for (int i=0; i<ROW; i++)
                cout << setw(10) << obj[i][0].getSquare( )
                     << setw(10) << obj[i][1].getSquare( )
                     << endl;
        return 0;
}
```

Ex2N15.cpp program run



Averaging an Array of Objects

```
// Listing 7.16. This program averages an array of
// Length objects that is typed in by user

#include <iostream>

const int   MAX = 50;
class  Length
{
      int    feet;
      float  inches;
 public:
      Length( )
            { feet=0; inches=0; }

      Length (int ft, float  in)
            { feet=ft; inches=in; }

      void  getLength( )
            {
                  cout << "\nEnter feet : ";
                  cin >> feet;
                  cout << "Enter inches : ";
                  cin >> inches;
            }

      void  showLength( )
            { cout << feet << "\'-" << inches << "\""; }

      void  addLength (Length  x, Length  y)
            {
                  inches=x.inches + y.inches;
                  feet = 0;
                  if (inches >= 12)
                  {
```



```
                        >>;
```

```
                inches -= 12;
                feet++;
            }
            feet += x.feet + y.feet;
        }

    void getAvg (Length x, int divisor)
        {
            float y = x.feet + x.inches/12;
            y /= divisor;
            feet = int(y);
            inches = (y-feet)*12;
        }
};

int main( )
{
    Length obj[MAX];
    Length total, ans;
    int count=0;
    char ch;

    do {
        cout << "Enter a length";
        obj[count++].getLength( );
        cout << "\nDo you want another (y/n) ? ";
        cin >> ch;
    } while (ch != 'n');

    for (int i=0; i<count; i++)
        total.addLength(total, obj[i]);
    ans.getAvg (total, count);
    cout << "\nThe average is ";
    ans.showLength( );
    cout << endl;
    return 0;
}
```

Ex7016.cpp program ကိုစမ်းသပ်ခြင်း

- Ex7016.cpp program သည် type class Length ဖြစ်သည့် array object (50) ခုအား store လုပ်သော ft-inch data များကို ဖတ်ပြီး၊ average တွက်သည့် program ဖြစ်သည်။ 50 သော maximum array element အထိသာ လက်ခံနိုင်ပြီး object များအတွက် data များကို ထည့်သွင်းရာတွင် do-while loop ကိုအသုံးပြု၍ feet, inches data များအား ထည့်သွင်းပြီးတိုင်း count သို့မဟုတ် ယခုလက်ခံသော data ကို ပြောင်းလဲစစ်ဆေးရန် Do you want another (y/n) ? prompt ထုတ်ပေးပြီး y or N ခလုတ်ကိုနှိပ်ရန် ကျန်ရှိသည်။ n ကိုနှိပ်လိုက်ပါက average တွက်ချက်ခြင်းကို လုပ်ဆောင်ကာ display လုပ်ပြသည်။
- ဤ program မှ Length constructor function ကို ပြင်ဆင် ထပ်မံအသုံးပြုထားသည်။ Ex7016.cpp program ကို run ကြည့်ခဲ့ပြီးပါက ပုံ (၁) ကဲ့သို့ တွေ့ရမည်ဖြစ်ပါသည်။

ပုံ (၁)

Chapter 8

operator overloading ဆိုတာ C++ နဲ့ object-oriented programming (OOP) မှာ စိတ်ဝင်စားဖွယ် အဓိကအင်္ဂါရပ်များထဲက တစ်ခုဖြစ်ပြီး ကျွန်တော်တို့ကတော့ C++ program statement မှာတွေ့ရတဲ့ operator overloading ဥပမာတွေ ဖြစ်တဲ့ integer ဒေတာအမျိုးအစား floating-point variable တွေအကြောင်းတွေကို C++ ရဲ့ ပုံမှန် operator အတိုင်း ' + ' operator ကို အသုံးပြုပြီး c= a+b; ဒီနည်းပဲဖြစ်ဖြစ် a နဲ့ b ဟာ ဘယ်လို class type object အသစ် ဖြစ်ဖြစ် သတ်မှတ်ပြုလုပ်ပေးနိုင်တဲ့ အကြောင်းကိုလည်းကောင်း အဲဒီလိုလုပ်ရတဲ့ အကြောင်းရင်းတွေကိုလည်းကောင်း ရှင်းပြသွားမှာပါ

```
r.addObjs (a,b);
```

ဒီလောက်ဆိုရင် ကောင်းကောင်းနားလည်သွားပြီလို့ ကျွန်တော်ထင်ပါတယ် ' + ' operator ဟာ ပုံမှန် ဒေတာ type တွေကိုပဲ ပြုလုပ်ခဲ့တဲ့ object တွေကိုလည်း သတ်မှတ်ပေး ထားပြီးသားမို့လို့ အခြားအခြားသော ရှုပ်ထွေးတဲ့ ပုံစံတွေအတွက်ပါ ကျွန်တော်တို့ကိုယ်တိုင် ရေးသားပေးရမှာပါ ဥပမာ c=a+b, ဆိုတာမျိုးဆိုရင်လည်း တစ်ခုသတိထားရမှာက ' - ' operator ကို overload လုပ်မယ်ဆိုရင် နှုတ်လိုက်တဲ့ ရလဒ်ကို ဘယ်လိုသိမ်းဆည်းမလဲ ဆိုတာမျိုးတွေ ဆုံးဖြတ်ပေးရမှာ ဖြစ်ပါတယ် ဒီလိုမျိုး drive လုပ်ခဲ့တဲ့ meaning တွေအတိုင်း operator overloading ဟာ C++ language ကိုပိုပြီးတော့ ပြည့်စုံအောင် ပြုလုပ်ပေးထားတာပါ

9.3 Overloaded ++ Operator

ကျွန်တော်တို့ operator တစ်ခုကို overload လုပ်ခြင်းဟာ တကယ်တော့ operator function တစ်ခုကို create လုပ်တာပဲဖြစ်တယ်။ operator overloading ဆိုတာ function overloading နည်းပညာကိုအသုံးပြုတာကြောင့် Ex901.cpp program မှာ counter ဆိုတဲ့ class တစ်ခုကို create ပုံဖော်ပြီး objects နှစ်ခု မှာ ပုံတူ variable count ကိုတိုးပေး increment လုပ်ပေးမယ့် member function ကို အတွဲလိုက် call ခေါ်ပြထားတယ်။

```
a.incCount( );
printCount( );
```

အထက်ပါ statement (2) ခုကို အစား ++a; a++; ဒီနှစ်ခုထဲကတစ်ခုခု ပြောင်းလဲရေးသားလိုက်ရင် '++' operator ကို overload လုပ်ပြီးသားမို့ ချက်ချင်း compiler ဟာ ++ operator ကို တွေ့ပြီး ပုံတူတစ်ခုခုမှာ member function ကို call ခေါ်ပြီး operand ကို increment လုပ်ပေးမယ့် အစားထိုးပြီးသား operand ဖြစ်တဲ့ a နဲ့ b ကိုတွေး 'counter' class type ဖြစ်ပြီးတော့ a နဲ့ b ကိုတော့ integer variable တွေ ဖြစ်နေမယ်ဆိုရင် compiler က C++ built-in '++' operator ကို အသုံးပြုပြီး increment လုပ်ပေးမှာဖြစ်ပြီး '++' operator (2) ခုကို compiler အဆိုပြုပြောင်းလဲပေးခြင်းကြောင့် Ex901.cpp program ကိုအဆင်ပြေပြေ run နိုင်ပါတယ်။

```
// Listing 9.1  Incrementing a counter variable
#include <iostream>

class counter
{
      unsigned int  count;
  public:
      counter( )
            { count = 100; }

      int getCount( )
            { return count; }

      void  incrCount( )
```

```
        { count++; }
};

int main( )
{
        Counter x,y;            // x=100, y=100

        cout << "\nBefore incrementing\n"
             << "\tx = " << x.getCount( )
             << "\n\ty = " << y.getCount( );
        x.incrCount( );
        y.incrCount( );
        y.incrCount( );
        cout << "\n\nAfter incrementing\n"
             << "\tx = " << x.getCount( )
             << "\n\ty = " << y.getCount( ) << endl;
        return 0;
}
```

Ex801.cpp program ကို run ကြည့်ခဲ့ရင် ဒုတိယပုံအတိုင်းမြင်ရမှာဖြစ်ပါတယ်

ပုံ (၈.၁)

overloaded ++ operator ကိုသုံးထားပြီး Ex801.cpp program ကိုအနည်းငယ်ပြောင်းလဲထားတဲ့
(သို့) ပေါင်းထားတဲ့ Ex802.cpp program ကိုစမ်းသပ်ကြည့်ပါ

Ex802.cpp

```
// Listing 8.2  Using overloaded ++ operator
#include <iostream>

class counter
{
        unsigned int  count;
public:
        counter() : count(100) {}

        int getCount() { return count; }

        void operator++(int) // for x++ and y++
                { count++; }

        void operator++()   // for ++x and ++y
                { count++; }
};


int main()
{
        counter x, y;

        cout << "\nBefore incrementation:"
             << "\nx = " << x.getCount()
             << "\ny = " << y.getCount();
        x++;
        ++y;
        cout << "\n\nAfter incrementation:"
             << "\nx = " << x.getCount()
             << "\ny = " << y.getCount() << endl;
        return 0;
}
```

(သို့)

Ex802.cpp program ကိုပေးထားကြည့်မယ်ဆိုရင်

- main( ) function ထဲမှာ counter class object (2) ခုကြေငြာပြီး x နဲ့ y ဆိုတာကို define လုပ်လိုက်ပေးနဲ့ counter( ) constructor function ကအလုပ်လုပ်သွားပြီး x.count=100 နဲ့ y.count=100 ကို initialize လုပ်ပေးလိုက်တယ်။ ဒီတော့ x.getCount( ) နဲ့ y.getCount( ) ကို call ခေါ်လိုက်တာနဲ့ object x နဲ့ y ရဲ့တန်ဖိုး count တန်ဖိုးတွေဟာ 100 ဆိုတာကို return ပြန်ပေးပါလိမ့်မယ်။ getCount( ) function က count တန်ဖိုးတွေကို ထုတ်ပြန်ပေးခြင်းဖြစ်ပေတာကိုပဲ
- ဒုတိယအနေနဲ့ object x နဲ့ y ကို '++' operator နဲ့ overload လုပ်လိုက်တဲ့အတွက် x နဲ့ y ရဲ့ count data တန်ဖိုးကို increment လုပ်သွားတယ်။ ဒီတစ်ကြောင်းမှာ void operator ++ (int) ဆိုတဲ့ function member ကို call ခေါ်ထဲကိုဝင် x++ နဲ့ y++ ကိုရေးပြန်ပေးတာပေါ့
- ++y ဆို့ operator overloading လုပ်မယ်ဆိုရင် void operator ++( ) ဆိုတဲ့ function member ကို call ခေါ်သုံးရမှာ ++y ကိုရေးပြန်ပေးရပါလိမ့်မယ်။ Ex802.cpp program ကို run မယ်ဆိုရင် ဒုံ (ဇး ဘဲ) မျက်ထဲတွေ့အပြဲခရုခန့်ခြံစံပါတယ်။

၈.၂ Overloaded + Operator

ဒီ အခုတစ်ခါကတော့ Ex803.cpp program မှာ overloaded + operator အလုပ်လုပ်ပုံကို ဆက်ပြီး အလေ့လာရအောင် လေ့လာကြည့်ကြပါ။

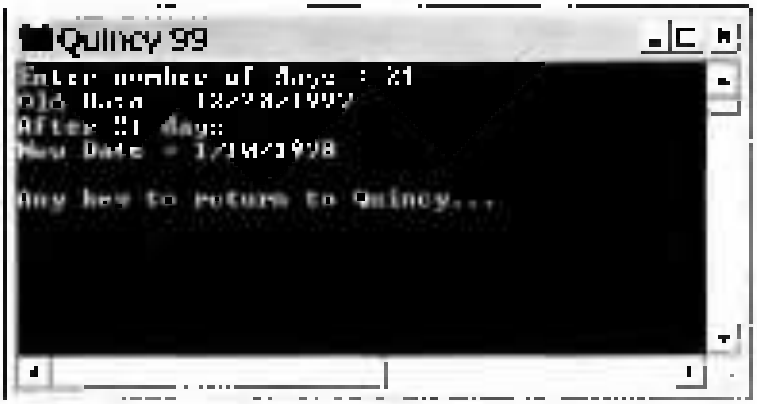
```
// Listing 8.3: Using overloaded – operator
#include <iostream>

class  Date
{
        int    month, day, year;
        static  int days[];

  public:
        Date (int m=0, int d=0, int y=0)
                { month = m; day = d; year = y; }
```

```
        void    display( ) const
                { cout << month << '/' << day << '/' << year << endl; }

        // Overloaded + operator
        Date    operator+(int) const;
};

int   Date::days[]={31,28,31,30,31,30,31,31,30,31,30,31};

// Overloaded + operator definition
Date  Date::operator+(int n) const
{
        Date  x = *this;
        n += x.day;                         // n = 21 + 20 = 41

        while (n > days[x.month-1])
        {
                n -= days[x.month-1];       // n = 41 - 31 = 10
                if (++x.month == 13)
                {
                        x.month = 1;        // month = 1
                        x.year++;           // year = 1998
                }
        }
        x.day = n;
        return x;
}


int  main( )
{
        Date   oldDate(12,20,1997);
        Date   newDate;
        int    x;

        cout << "Enter number of days : ";
        cin >> x;
```

```
    newDate = oldDate + x;              // three weeks hence
    cout << "Old Date = ";
    oldDate.display();
    cout << "after " << x << " days\n";
    cout << "New Date = ";
    newDate.display();

    return 0;
}
```

Ex803.cpp program ကိုအလုပ်လုပ်ပုံခြင်းဆက်

- ပထမဦးဆုံး main() function မှာ Date object (2) ခုကြေငြာပြီး oldDate နဲ့ newDate တို့ကို constructor သုံးပြီးသတ်မှတ် oldDate ကတော့ oldDate.month = 12 oldDate.day = 20 နဲ့ oldDate.year = 1997 တို့ကို assign လုပ်ပေးမှာပါ။
- Enter number of days : ဆိုတဲ့ prompt ထုတ်ပေးတဲ့အခါမှာ x = 21 ဖြစ်သွားမယ်ဆိုရင် newDate = oldDate + x; ဒီလို statement ကြီးကိုကွဲသွားပြီး newDate = oldDate.operator + (21); ဖြစ်သွားမှာပေါ့။ ဒီတော့ overloaded + operator function ဆီကို n = 21 တောင့်နဲ့ pass လုပ်ပေးလိုက်မှာပေါ့။
- overloaded + operator function ထဲရောက်လာတဲ့အခါ n += x.day = 21 + x.day = 21 + oldDate.day = 21 + 20 = 41 ဖြစ်သွားပါတယ်။ while (n > days[x.month-1] ဆိုတာ while (41 > days[12-1] ဖြစ်လာပြီး while (41 > 31) ဆိုပြီးဖြစ်သွားမယ်။ ဒီတော့ while loop ထဲဝင်တော့မှာ n -= days[x.month-1] = 41 - 31 = 10 ဖြစ်သွားမှာပေါ့။
- ဒီနောက်တော့ if (++x.month == 13); ဆိုတာကိုဖတ်မယ်။ ဒါကြောင့်လည်း if (12+1 == 13) ဖြစ်သွားတဲ့အတွက် ဒီနောက် if block statement ကိုဝင်သွားပြီး x.month = 1 နဲ့ x.year++ = 1997 + 1 = 1998 ဖြစ်သွားမှာပါပဲ။
- နောက်ဆုံးတော့ while loop ကိုတွေ့တော့ x.day = n - 10 ကို assign ပြန်ပေးမှာဖြစ်ပြီး Date object x ကို main() function ဆီကို return ပြန်ပေးလိုက်တဲ့အခါ main() ထဲမှာရှိတဲ့အတိုင်း newDate = (1,10,1998) ဖြစ်သွားမှာပါ။ Ex803.cpp program ကို run လိုက်ရင်အဲဒီလိုမျိုး ရလဒ်ထွက်လာတာကိုတွေ့ရမှာပါ။



More on Overloaded + and ++ Operators

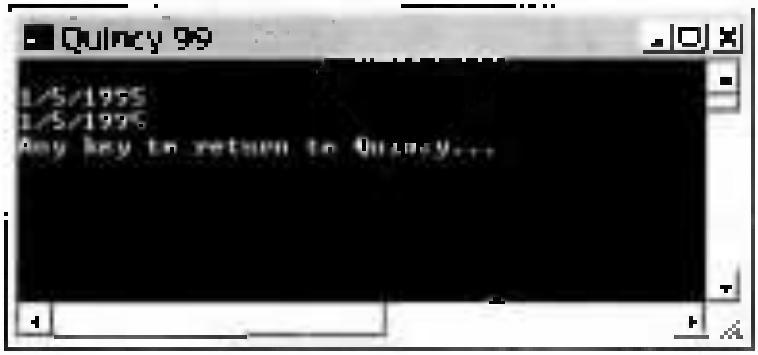
... Ex8b4.cpp program ...ဖြင့် overloaded +/++ operator ...ကြောင့် program ...ပြောင်းလဲသွားပုံ ... program ...ကိုကြည့်ပါ။ ...

```
// Listing 8.4: More on overloaded + and ++ operators
#include <iostream>

class   Date
{
        int     month, day, year;
        static int  days[];

  public:
        Date (int m=1, int d=1, int y=1)
                { month = m, day = d, year = y; }

        void    display( ) const
                { cout << endl << month << '/' << day << '/' << year; }
```

```
    // Overloaded + operator.
    Date  operator +(int n) const
          {
                Date x = *this;
                n += x.day;
                while (n > days[x.month-1])
                {
                      n -= days[x.month-1];
                      if (++x.month == 13)
                      {
                            x.month = 1;
                            x.year++;
                      }
                }
                x.day = n;
                return x;
                }

    // Overloaded prefix ++ operator.
    Date  operator++( )
          {
                *this = *this + 1;
                return *this;
          }

    // Overloaded postfix ++ operator.
    Date  operator++(int)
          {
                Date x=*this;
                *this=*this+1;
                return x;
          }
};

int   Date::days[]={31,28,31,30,31,30,31,31,30,31,30,31};
```

```
int main( )
{
        Date   newDate, oldDate(1,9,1995);

        newDate = oldDate++;
        oldDate.display( );
        ++newDate;
        newDate.display();

        return 0;
}
```

Ex0804.cpp program ကို run လိုက်လျှင်ရရှိသော output (ပုံ ၄-၄) ကိုကြည့်ပါ။



ပုံ (၄-၄)

Overloading Arithmetic + and - Operators

Ex0805.cpp program တွင် Length object နှစ်ခုဖြစ်သော L1 နှင့် L2 ကိုကြေငြာ define လုပ်ထားပြီး Length object L3 နှင့် L4 တို့ကို overloaded +/- operator တွင်နှင့် အသုံးပြုကြေငြာထားကြောင်းတွေ့ရမည်ဖြစ်သည်။ program ကိုစတင်လုပ်ဆောင်ပုံကို trace လုပ်ကြည့်ရန်ဖြစ်သည်။

```
// Listing 8.5: Overloading arithmetic + and - operators
#include <iostream>

class  Length
{
      int   feet;
      float inches;
  public
      Length( )
            { feet = 0; inches = 0; }

      Length(int ft, float in)
            { feet = ft; inches = in; }

      void  getLength( )
            {
              cout << "Enter feet   : ";
              cin >> feet;
              cout << "Enter inches : ";
              cin >> inches;
            }

      void  showLength( )
            {
              cout << feet << "\'-" << inches << "\"";
            }

      Length operator +(Length x)
            {
                  int   f = feet + x.feet;
                  float i = inches + x.inches;

                  if (i >= 12.0)
                  {       i -= 12.0;      f++;   }
                  return Length(f,i);
            };
```

```
        Length operator -(Length x)
                {
                        int   f = feet - x.feet;
                        float i = inches - x.inches;

                        if (i < 0)
                        {       i += 12.0;      f--;    }
                        return Length(f,i);
                }
};


int main( )
{
        Length   L1,L3,L4;

        cout << "Enter length for L1\n";
        L1.getLength( );

        Length   L2(11,6.25);
        cout << "For L2\nL2, length = ";
        L2.showLength( );
        cout << endl;
        L3 = L1 - L2;
        L4 = L1 + L2 + L3;

        cout << "\nLength1 = ";
        L1.showLength( );
        cout << "\nLength2 = ";
        L2.showLength( );
        cout << "\nLength3 = ";
        L3.showLength( );
        cout << "\nLength4 = ";
        L4.showLength( );
        cout << endl;
        return 0;
}
```

Ex805.cpp program ကို run ပြီးပါပြီး output ကို ဖော်ပြထားပါသည်။

(ပုံ ၈-၈)

Adding Vectors

ယခု vector sum နှင့်တွဲလျက် vector subtract ပြုလုပ်နိုင်ရန်အတွက် operator overloading ကို အသုံးပြု 2D ကိုဩဒိနိတ်စနစ်ကို ကိုယ်စားပြုတဲ့ class တစ်ခုကို ဖန်တီးကြပါစို့။ 2D plane တစ်ခုပေါ်ရှိ point တစ်ခုကို (x,y) rectangle coordinates များဖြင့် သို့မဟုတ် polar coordinates (radius, angle) ပုံစံဖြင့် ဖော်ပြနိုင်ပါသည်။ ကျွန်တော်တို့ program တွင် အရင်ဆုံး vector တစ်ခုကို rectangular coordinates အသုံးပြုပြီး program ရေးကြည့်ပြသထားပါသည်။ vector တစ်ခုမှာ (x=2, y=3) နှင့် vector တစ်ခုမှာ (x=5, y=2) ဖြစ်သည်ဆိုပါစို့။ ၎င်း vector (2) ခုကိုပေါင်းကြည့်ပါမည်။

```
// Listing 8.6  Adding vectors

#include <iostream.h>
class  coord
{
      int    xco, yco;
```

```
public:
    coord( )            { xco = 0; yco = 0;}

    coord (int i, int j)  { xco = i; yco = j;}

    void  getXy (int &i, int &j);
          {i = xco ; j = yco ;}

    coord  operator + (coord  obj)
          {return  coord (xco+obj.xco, yco+obj.yco);}

    coord  operator - (coord obj)
          {return  coord (xco-obj.xco, yco-obj.yco);}
};

int main( )
{
    coord   v1(2,4), v2(5,2), vr;
    int     x, y;

    cout << "First vector for v1 is\n";(2, 3)\n\n",
    cout << "Second vector for v2 is\n";(5, 2)\n\n";
    vr = v1 + v2;

    vr.getXy (x, y);
    cout << "Thus New vector for v1 + v2 is\n"
         << "i\" << x << ", " << y << "\n\n";

    vr = v2 - v1;
    vr.getXy (x, y);
    cout << "New vector for v2 - v1 is\n"
         << "i\" << x << ", " << y << "\n\n";
    return 0;
}
```

Ex606.cpp program ကိုစမ်းသပ်ကြည့်ပါမယ်။

- ဒေတြာ main( ) function ထဲက class coord object တွေဖြစ်တဲ့ v1 ၊ v2 နဲ့ vr ကို define လုပ်ပါတယ်။ ဒီလောက်ဆိုရင် coord constructor အလုပ်လုပ်သွားပြီး (v1.xco = i = 2 ၊ v1.yco = j = 3)၊ (v2.xco = _ = 5 ၊ v2.yco = j = 2) နဲ့ (vr.xco = 0 ၊ vr.yco = 0) ကို initialize လုပ်ပေးပါတယ်။ vr = v1 + v2 ဆိုတာ vr = v1.operator+(v2) ဖြစ်ပါတယ်။ vr = v2 - v1 ဆိုတာ vr = v2.operator (v2) ဖြစ်ပါတယ်။
- Ex606.cpp program ကို run လိုက်ရင်အဆင့်ဆင့် (၇) ခု ဖော်ပြသွားမှာဖြစ်ပါတယ်။

ပုံ - ၇

Overloaded + Operators with Polar Coordinates

ဒီ ဥပမာမှာတော့ rectangular coordinates အစား polar coordinates ကို အသုံးပြုထားပါတယ်။ အဲဒီလို အသုံးပြုနိုင်ဖို့အတွက် အောက်ပါအတိုင်း ဆက်စပ်ပြောင်းလဲပေးရပါတယ်။

x = radius * cos (angle)

y = radius * sin (angle); where angle is in radians

```
        radius =        sqrt( x*x + y*y )
        angle = atan (y / x)
```

```
// Listing 8.7  Overloaded + operators with polar coordinates

#include <iostream>
#include <cmath>

class  polar
{
   private:
        double    radius;
        double    angle;
        double    getx( )
                { return   radius*cos(angle); }

        double    gety( )
                { return   radius*sin(angle); }

   public:
        polar( )
                { radius=0.0; angle=0.0; }

        polar (float r, float  a)
                {       radius = r;
                        angle = a;     }

        void   display( )
                {
                        cout << '(' << radius
                             << ',' << (int)(angle*180/3.141593) << ')';
                }
```

```
        polar   operator + (polar   p)
        {
                double          x = getx( ) + p.getx( );
                double          y = gety( ) + p.gety( );
                double          r = sqrt(x*x+y*y);
                double                  a = acos(y/x);
                return polar(r, a);
        }
};

int main( )
{
        polar   p1(5.0);
        polar   p2(5.1, 3.1415926535);
        polar   p3;

        p3 = p1 + p2;
        cout << "The sum of two polar vectors are:\n";
        cout << "\tP1 = ";
        p1.display( );
        cout << "\tp2 = ";
        p2.display( );
        cout << endl;
        cout << "\n\nSum of the two polar vectors is\n";
        cout << "\tp3 = ";
        p3.display( );
        cout << endl;
        return 0;
}
```

Ex8b7.cpp program output

- The main( ) function creates polar class type object (3) and defines objects p1(5.0), p2(5.1, 3.1415926535) & p3. p1.radius = r = 5, p1.angle = a = 0), p2.radius = r = 5, p2.angle = a = π/2) & (p3.radius = 0, p3.angle = 0);

return type ကြေညာထားတဲ့အတွက်ကြောင့် p3 = p1 + p2; ဆိုတဲ့ statement ကြောင့်
ဖြစ်ပေါ်လာမယ့် အဖြေတွေကတော့

```
x = getx( ) + p2.getx( )
  = p1.getx( ) + p2.getx( )
  = p1.radius*cos(p1.angle) + p2.radius*cos(p2.angle)
  = 5*cos(0) + 5*cos(π/2)
  = 5 + 0 = 5

y = gety( ) + p2.gety( )
  = p1.gety( ) + p2.gety( )
  = p1.radius*sin(p1.angle) + p2.radius*sin(p2.angle)
  = 5*sin(0) + 5*sin(π/2)
  = 0 + 5 = 5

r = sqrt(x*x + y*y)
  = sqrt(5*5 + 5*5)
  = sqrt(50)
  = 7.07107

a = atan (y / x)
  = atan(5/5)
  = atan(1)
  = 0.7854
```

- ပြီးရင် return polar(r,a); ဆိုတဲ့ statement ကနေ တဲ့အဲဒါကို main( ) ကို ပြန်ပြီးခေါ်ပေးတဲ့အခါ
ကြေးမုံပြင်ပေါ်မှာ object p3 ထဲမှာ p3.radius = 7.07107 နဲ့ p3.angle = 0.7854 ဆိုတဲ့
တန်ဖိုးတွေကို ထိုးသွင်းပေးတဲ့ object p3 ထဲက radius နဲ့ angle တန်ဖိုးတွေကို display လုပ်ဖို့
အတွက် p3.display( ) function ကို call ခေါ်ပြီး အလုပ်လုပ်ခိုင်းလိုက်ပါတယ်။

p1.radius = 5
angle (in degrees) = p1.angle*180/3.141593
= 0 * 180/3.141593 = 0

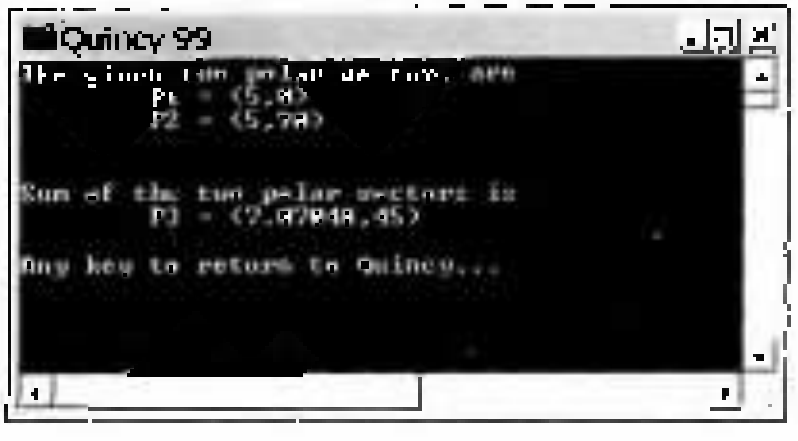
- ဒုတိယအနေဖြင့် object p2 တွင် radius နှင့် angle တို့ကိုထည့်သွင်းပေးလိုက်သည်။

p2.radius = 5
angle (in degrees) = p2.angle*180/3.141593
= 1.570796325 * 180/3.141593 = 90

- object p3 တွင် radius နှင့် angle ထည့်သွင်းပြီးနောက် p3.display( ) function ခေါ်ယူခြင်းဖြင့် ပေါင်းလဒ်ဖြစ်သော ပိုလာကိန်းကို display ပြသခိုင်းလိုက်သည်။

p3.radius = 7.07107
angle (in degrees) = p3.angle*180/3.141593
= 0.7854 * 180/3.141593 = 45

- ဥပမာ ၁.၁၃ program ကို run လုပ်ပေးလိုက်သည့်အခါ ပုံ (၁.၇) ကဲ့သို့ ရလဒ်ထွက်ပေါ်လာမည်ဖြစ်သည်။



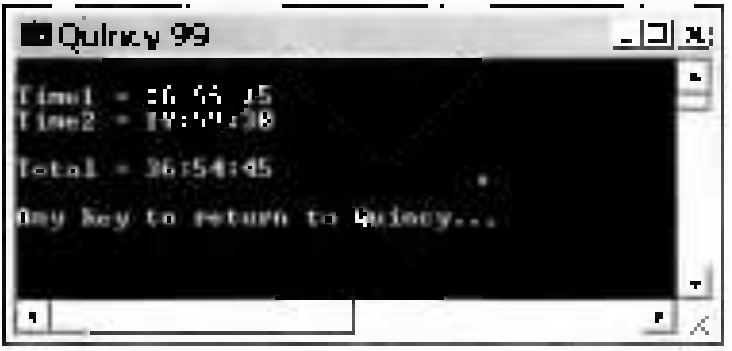
ပုံ (၁.၇)

Using Overloaded + Operator to Add Time

```
//Listing 8.8.  Using overloaded '+' operator to add times
#include <iostream>

class mytime
{
      int     hrs, mins, secs;
   public:
      mytime( )
            {hrs = mins = secs = 0;}

      mytime (int h, int m, int s)
            { hrs = h; mins = m; secs = s; }

      void  display( )
            { cout << hrs << ':' << mins << ':' << secs; }

      mytime    operator +(mytime  mt)
            {
                  secs += mt.secs;
                  if (secs>59)
                  {
                        secs -=60;
                        mins++;
                  }
                  mins += mt.mins;
                  if (mins>59)
                  {
                        mins -= 60;
                        hrs++;
                  }
                  hrs += mt.hrs;
                  return  mytime (hrs,mins,secs);
            }
};
```

```
int main( )
{
        mytime      t1(16, 55, 15);
        mytime      t2(19, 55, 30);

        cout << "\nTime1 = ";        t1.display( );
        cout << "\nTime2 = ";        t2.display( );
        mytime total = t1 + t2;
        cout << "\n\nTotal = ";      total.display( );
        cout << endl;
        return 0;
}
```

Ex808.cpp program ကို run ကြည့်လိုက်ရင် ပုံ (၈.၈) အတိုင်းတွေ့ရမှာပါ။



ပုံ (၈.၈)

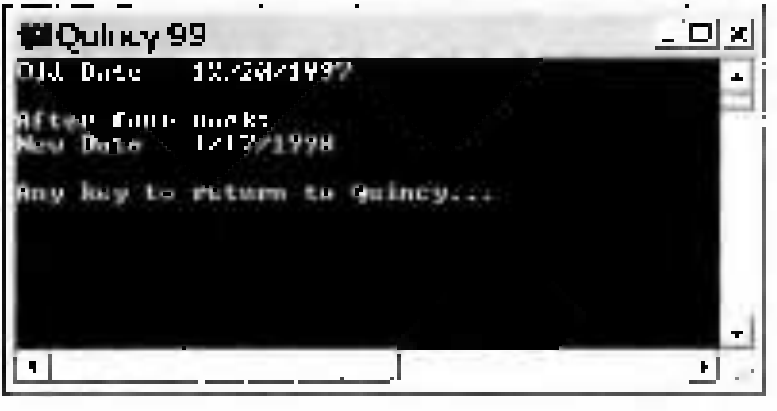
၈.၇ Overloading + with a Nonmember Function

```
//Listing 8.9:  Overloading + with a nonmember function
#include <iostream>
```

```
class Date
{
        int   month, day, year;
        static int  days[];

  public:
        Date (int m=0, int d=0, int y=0)
                { month = m; day = d; year = y; }

        void  display () const
                { cout << month << '/' << day << '/' << year; }

        // Overloaded + operator
        Date  operator +(int) const;
};

int  Date::days[ ]={31,28,31,30,31,30,31, 31,30,31,30, 31};

// Overloaded + operator: Date + int.
Date Date::operator +(int  n) const
{
        Date x = *this;
        n += x.day;
        while  (n > days[x.month-1])
        {
                n -= days[x.month ];
                if (++x.month == 13)
                {
                     x.month = 1;
                     x.year++;
                }
        }
        x.day = n;

        return  x;
}
```

```
Date operator + (int n, Date& x)
{
        return  x + n;
}

int main( )
{
        Date    oldDate(12,20,1997);
        cout << "Old Date = ";
        oldDate.display( );
        cout << endl;

        Date  newDate = 15 + oldDate + 13;       // four weeks hence
        cout << "\nAfter four weeks\n" << "New Date = ";
        newDate.display();
        cout << endl;
        return 0;
}
```

Ex80.cpp program ကို run လိုက်ပေးခြင်းဖြင့် ၄ (က) ပုံအတိုင်းတွေ့ရမည်ဖြစ်သည်။



၄ (က)

Overloading the Assignment += Operator

... assignment operator ... += ... ^= ...
... += operator ... Ex8010.cpp program ...

```
//Listing 8.10: Overloading the assignment += operators
#include <iostream>

class Date
{
      int  month, day, year;
      static int days[ ];

  public:
      Date (int m=0, int d=0, int y=0)
            { month = m; day = d; year = y; }

      void  display( ) const
            { cout << month << '/' << day << '/' << year; }

      // Overloaded + operator.
      Date operator + (int) const;

      // Overloaded += operator.
      Date operator += (int n)
            {
                  *this = *this + n;
                  return *this;
            }
};
int  Date::days[ ]={31,28,31,30,31,30,31,31,30,31,30,31};
```

```
// Overloaded + operator definition.
Date Date::operator + (int n) const
{
        Date x = *this;
        n += x.day;
        while (n > days[x.month-1])
        {
                n -= days[x.month-1];
                if (++x.month == 13)
                {
                        x.month = 1;
                        x.year++;
                }
        }
        x.day = n;
        return x;
}

int main( )
{
        Date oldDate(1,4,1995);
        cout << "Old Date = ";
        oldDate.display( );
        cout << endl;

        oldDate += 365;
        cout << "\nAfter 365 days:\n"
             << "New Date = ";
        oldDate.display( );
        cout << endl;

        return 0;
}
```

ExDate.cpp program output

Concatenating Strings with Overloaded += Operators

ဥပမာ Listing 9.12.cpp program မှာ object နှစ်ခုအကြား string ကို object တစ်ခုက
ကျန်တဲ့ string နဲ့ဆက်ပြီးထည့်သွင်းသွားတဲ့ assign လုပ်ပုံကိုပြထားပါတယ်။ ဒီမှာ s1 ကို s2 နဲ့ဆက်ပြီး assign
လုပ်ထားတဲ့အတွက် s3 ကို display လုပ်တဲ့အခါမှာ ပုံ (၈.၈) အတိုင်းတွေ့မြင်ရမှာဖြစ်ပါတယ်။

```
// Listing 9.12: Concatenating the strings

# include <iostream>
# include <cstring>

const int    MAX = 80;

class myString
{
        char  str[MAX];

   public:
        myString( )                 {strcpy(str," ");}
        myString(char s[ ])         {strcpy(str, s);}
        void  display( );           {cout << str;}
```

```
        myString  operator += (myString  ms)
            {
                    if (strlen(str) + strlen(ms.str) < MAX)
                    {
                            strcat(str,ms.str);
                            return str;
                    }
                    else   cout << "\nString overflow.";
            }
};

int main( )
{
        myString  s1 = "\n\tYear 2004!  ";
        myString  s2 = "Happy New Year!";
        myString  s3 = "\n\t<Blank>";

        s1.display( );          s2.display( );          s3.display( );

        s3 = s1 += s2;
        s3.display( );
        cout << endl;
        return 0;
}
```

Ex8011.cpp program ကို run လိုက်ပေးခြင်းဖြင့် ၄ (၈၀ ၁၀) ကဲ့သို့ output ကိုရရှိနိုင်ပါသည်။

၄ (၈၀ ၁၀)

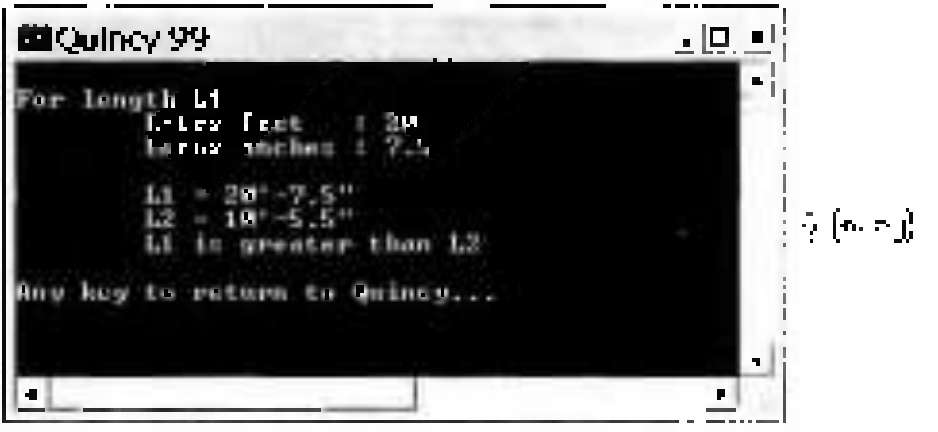
8.9 Overloaded Relational Operators

Listing 8.12 is the Ex8012.cpp program. The less than '<' operator is overloaded to compare length class type objects. The overloaded operator function is

```
// Listing 8.12: Using the overloaded relational < operator

#include <iostream>

bool   TRUE = 1, FALSE = 0;
class  Length
{
      int    feet;
      float inches;

   public:
      Length( )
            {feet=0,  inches=0; }

      Length (int ft, float in)
            {feet=ft; inches=in; }

      void   getLength( )
            {
                  cout << "\n\tEnter feet   : ";   cin >> feet;
                  cout << "\tEnter inches : ",    cin >> inches;
            }

      void  showLength( )
            {
                  cout << feet << "\' " << inches << "\"";
            }
```

```
        bool  operator < (Length l)
              {
                    float    f1=feet + inches/12;
                    float    f2=l.feet + l.inches/12;
                    return   (f1<f2) ? TRUE:FALSE;
              }
};

int  main( )
{
      Length L1;
      cout << "\n\nFor length L1";          L1.getLength( );

      Length  L2(10,5.5);
      cout << "\n\nL1 = ";                  L1.showLength( );
      cout << "\n\nL2 = ";                  L2.showLength( );
      if (L1 < L2)
            cout << "\n\nL1  is less than L2\n";
      else
            cout << "\n\nL1 is greater than L2\n";
      return 0;
}
```

Ex8012.cpp program ကို run လိုက်လျှင် ဖော်ပြပေးမှုမျိုးရရှိမည်ဖြစ်သည်။



More on Overloaded Relational Operators

```
// Listing 8.13: Move on overloaded relational operators
#include <iostream>

class  Date
{
       int    month, day, year;

    public:
       Date (int m=0, int d=0, int y=0)
              { month = m; day = d; year = y; }
       void   display( ) const
              { cout << "\n\t" << month << '/' << day
                     << '/' << year << endl; }

       // Overloaded operators.
       int   operator == (Date& dt) const;
       int   operator < (Date&) const;
};

// Overloaded equality operator definition
int  Date::operator == (Date& x)  const
{
       return  (this->month == x.month &&
       this->day   == x.day &&
       this->year  == x.year);
}

// Overloaded less-than operator definition.
int  Date::operator < (Date& x)  const
{
       if (this->year == x.year)
       {
              if (this->month < x.month)
                     return   this->day < x.day;
```

```
        return  this->month < x.month;
    }
    return  this->year < x.year;
}

int  main( )
{
    Date  date1(10,25,1947),
    date2(11,10,2003),
    date3(10,25,1947);
    if (date1 < date2)  {
        date1.display( );
        cout << "\t  is LESS than ";      date2.display( );
    }
    cout << endl;

    if (date1 == date3)  {
        date1.display( );
        cout << "\t  is EQUAL to ";       date3.display( );
    }
    return 0;
}
```

Ex8013.cpp program ကို run ကြည့်ခဲ့ရင် ဒုတိယ [illegible]

[illegible]

9.6 Overloading == Operators

Listing 9.14 ကို ပေါ်ကြည့်ပါ။ ExAbt14.cpp program သည် == operator ကို overload လုပ်ခြင်းဖြင့် myString class type ဖြစ်သည့် string (2) ခုကို နှိုင်းယှဉ်ကြည့်ခြင်းဖြင့် ပရိုဂရမ်ကို ရေးသားထားပါသည်။

```
// Listing 9.14: Overloading == operator

#include <iostream>
#include <cstring>

const int    MAX = 80;
bool   TRUE = 1, FALSE = 0;

class myString
{
      char  str[MAX];
   public:
      myString( )
            {  strcpy (str,""); }

      myString (char s[ ])
            {  strcpy (str,s);  }

      void  getSt( )
            {
                  cout << ":";
                  cin.get (str,MAX);
            }

      bool  operator == (myString ms)
            {
                  return   strcmp(str,ms.str) ? FALSE:TRUE;
            }
};
```

```
int main( )
{
        myString   s1("yes");
        myString   s2;

        cout << "\n\nEnter 'yes' or 'no'" << endl;
        s2.getStr( );
        if (s2 == s1)
                cout << "\n\nYou typed yes.\n";
        else
                cout << "\n\nYou typed no.\n";
        return 0;
}
```

Ex8014.cpp program ကို run လိုက်သည့်အခါ ဤကဲ့သို့ ပေါ်လာမည်ဖြစ်ပါသည်။

ပုံ (က-၁၄)

Overloading - Operators

Listing 8.15 ဖော်ပြထားသော Ex8015.cpp program တွင်ပါ unary minus - operator ကို overload လုပ်ခြင်း အတွက် (2) ခုသော နည်းလမ်းများကို ရှင်းလင်းပြသထားပါသည်။ ပထမတစ်ခုမှာ

```
// Listing 6.15: Overloading unary minus - operator

#include <iostream>
#include <cstring>

class  amount
{
        int     x;
        char    ch[25];

    public:
        amount (int i, char *d)
                { x = i; strcpy(ch, d); }

        void  display( ) const
                {
                        cout << "\n\t" << ch << ": " << x
                                << " convert to";
                }

        int  operator - ( ) const
                {  return -x;  }
};


int  main( )
{
        amount     obj1(1000,"APPLES");
        amount     obj2(-50000,"ACCOUNT");

        obj1.display( );
        cout << "\n\t" << -obj1 << endl;

        obj2.display( );
        cout << "\n\t" << -obj2 << endl;
        return 0;
}
```

ပုံ (ဂ-၇)

၈.၆ Overloading [ ] Operators

Listing 8.16 ဟာ ဖော်ပြထားတဲ့ Ex8D16.cpp program ထဲက subscript [ ] operator အသုံးပြုပုံကို ဖော်ပြထားပါတယ်။ ဒီ program ဟာ string တစ်ခုကို store လုပ်ထားတဲ့ myStr class ထဲမှာ subscript operator ကို overload လုပ်ပြီး string value ထဲကနေပြန်ပဲ character position ကို access လုပ်နိုင်မှုအားကို တင်ပြထားပါတယ်။ ပါးစပ်က constant string အတွက်တော့ modify လုပ်ခိုင်းမှာ မဟုတ်ပါ။

```
// Listing 8.16: Using subscript operator
#include <iostream>
#include <cstring>

class  myStr
{
      char*  sp;

public:
      myStr (char*  s = 0);

      ~myStr( )  { delete sp; }
```

```
        void display( )
                { cout << sp << endl; }
        // Overloaded [ ] operator.
        char&  operator[ ] (int  n)
                { return  *(sp + n); }

        const char&  operator[ ] (int n) const
                { return *(sp + n); }
};

// The String class constructor.
myStr::myStr (char*  s)
{
        if (s)
        {
                sp = new  char[strlen(s)+1];
                strcpy(sp, s);
        }
        else   sp = 0;
}


int  main( )
{
        myStr  str1("Complete C++");
        str1.display();

        // Change some string characters.
        str1[4] = 'P';
        str1[5] = 'L';
        str1[6] = 'T';
        str1[7] = 'E';
        str1.display( );

        // Change a substring.
        strncpy(&str1[0], "COMPLETE", 8);
        str1.display( );
```

```
    const myStr str2("\nVERY COMPLETE C++");

    for (int i= 0; i<18 ; i++)
        cout << str2[i];
    // const string, cannot be modified.
    // strcpy(&str2[0], "COMPLETE", 8);
    // str2.display( );
    cout << endl;

    return 0;
}
```

Ex8016.cpp program ကို run ထဲ့ရင်ရလဒ် (ပြသမှု) ကိုအောက်ပါအတိုင်းတွေ့ရမှာပါ

ပုံ (၈.၄)

၈.၆ Overloading -> Operators

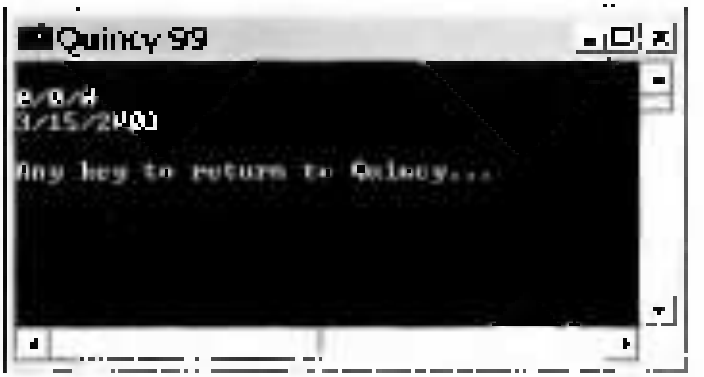
Listing 8.17 ဖော်ပြထားသော Ex8017.cpp program တွင်ကျွန်ုပ် pointer-to-member -> operator ကို overload လုပ်ထားခြင်းဖြစ်ပါသည်။ Date class object တွေ့ pointer တစ်ခုကိုဖွဲ့ပြီး data value assign ပြုလုပ်ရာတွင် အသုံးပြုခဲ့ပါသည်။ ထူးခြားချက်မှာ date pointer ကို data assign

ပုံသေးသေးလေးကို DatePtr constructor function က ပေး data မှာကိန်းဆောက်လုပ်ပြီး ဒီတော့ DatePtr object ဖြစ်သွား dp ဟာ zero value ကို point ပြန်လုပ်ပါဘိမယ်။ dp->display( ); ဟာ call ခိုက်ခဲ့ရင်ဟာ ဟာ Date* operator ->( ) function ထဲဝင်သွားပြီး if (dp == 0) နဲ့စစ်ကြည့်ခဲ့ရင် null Date object ရဲ့ address (&nulldate) ကို return ပြန်ပေးလိုက်၊ ဒီနှစ်ခု nulldate(0,0,0) က parameter တွေကို Date (int m=0, int d=0, int y=0) constructor function မှာ month = 0 day = 0 နဲ့ year = 0 ကို initialize လုပ်ပေးလိုက်မှာပဲ ဖါးဒို့ month ၊ day နဲ့ year ကိုရရှိတဲ့နှစ်ခုတွေကို display( ) function မှာ display ပြုလုပ်ခဲ့ရမှာပြီး date object မှာ parameter ပေး assign ထားတဲ့ကြိုးကိုးတွေကို ဟာအဲဒီလိုသဘောကို trace လုပ်ကြည့်ပေးခဲ့ပါဘယ်။

```
// Listing 8.17: Overloaded pointer-to-member -> operator
#include <iostream>

class  Date
{
        int   month, day, year;
    public:
        Date (int m=0, int d=0, int y=0)  { month = m; day = d; year = y; }

        void  display( )
              { cout << endl << month << '/' << day << '/' << year; }
};

class  DatePtr
{
       Date* dp;
    public:
       DatePtr (Date* d = 0)      { dp = d; }

       Date* operator ->( )
             {
                    static Date nulldate(0,0,0);
                    if (dp == 0)  return &nulldate;
                     return dp;
              }
};
```

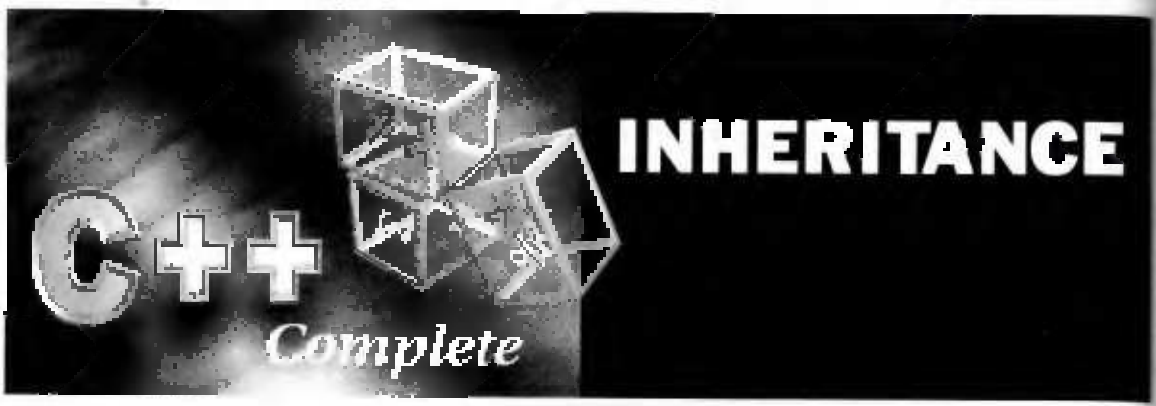
```
int main( )
{
        // Date pointer with nothing in it.
        Date* dp;

        // Use it to call display function.
        dp->display( );

        Date dt(3,15,2003);

        // Put address of date in pointer.
        dp = &dt;

        // Display date through the pointer.
        dp->display();
        cout << endl;

        return 0;
}
```

ex8017.cpp program-ကို run ပြုလုပ်ခြင်းဖြင့် ရလာမည့် output ကို ကြည့်ရှုနိုင်ပါသည်။

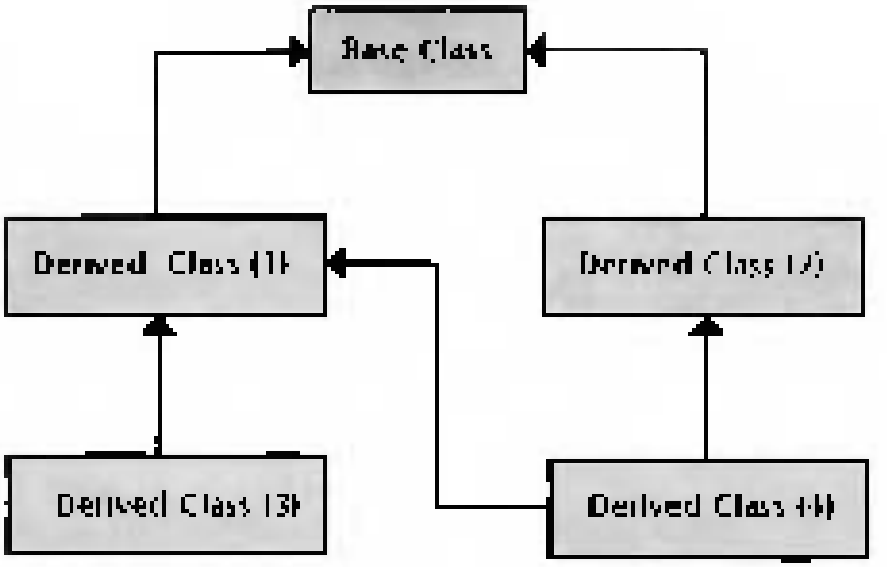


ပုံ (၈-၁၂)

Chapter 9



class ပြုလုပ်ရာတွင် C++ ပရောဂရမ်းမင်းဘာသာစကား၏ feature တစ်ခုဖြစ်သော inheritance ဟုခေါ်သော inheritance ဟာ ရှိပြီးသား base class တစ်ခုခုကို အခြေခံပြီး အသစ်သော class တစ်ခုကို ဖန်တီးနိုင်ခြင်းဖြစ်ပါသည်။ ဤသို့ derive လုပ်ခံရသော class ကို base class ဟုခေါ်ပြီး derive လုပ်ယူသော class ကို derived class ဟုခေါ်ပါသည်။ base class ရဲ့ အင်္ဂါလက္ခဏာများသည် ၎င်း class ကိုအခြေခံ၍ ဆင်းသက်လာသော derived classes တို့တွင်လည်း ပါဝင်သွားပါသည်။ derived class တစ်ခုတည်းတွင်ပင် ဆင်းသက်လာသော class များစွာရှိနိုင်ပါသည်။ ဤသို့ class များကို ပြန်လည်အသုံးပြုခြင်းဖြင့် ပရိုဂရမ်းမင်းလုပ်ငန်းစဉ်ကို ပိုမိုလွယ်ကူစေပါသည်။ ထို့အပြင် ပရိုဂရမ်းမင်းအမှားများကိုလည်း ပိုမိုလျော့နည်းစေပါသည်။ ဥပမာအားဖြင့် class တစ်ခုတွင် ပရိုဂရမ်းမင်းအမှားများ ရှိနေခဲ့ပါက ထို class ကို အခြေခံသော derived class အသစ်များတွင်လည်း ထိုအမှားများ ပါဝင်နေတတ်ပါသည်။ ထို့ကြောင့် ပရိုဂရမ်းမင်းအမှားများကို ရှာဖွေပြင်ဆင်ရန် base class တွင်သာ အာရုံစိုက်ရန် လိုအပ်ပါသည်။ ထို့အတူ ပရိုဂရမ်းမင်းအမှားများကို ပြုပြင်ပြီးပါက derived class များတွင်လည်း အလိုအလျောက် ပြုပြင်ပြီးသားဖြစ်သွားပါသည်။ ဤသို့ base class တွင်သာ ပရိုဂရမ်းမင်းအမှားများကို ပြုပြင်ရသောကြောင့် ပရိုဂရမ်းမင်းလုပ်ငန်းစဉ်ကို အလွန်ပိုမိုလွယ်ကူစေပါသည်။ ဤသို့ ပရိုဂရမ်းမင်းလုပ်ငန်းစဉ်ကို လွယ်ကူစေခြင်းကို reusability (ပြန်လည်အသုံးပြုနိုင်မှု) ဟုခေါ်ပါသည်။ ထို့ကြောင့် inheritance သည် reusability ကို ရရှိစေသော ပရိုဂရမ်းမင်းဘာသာစကားတစ်ခု၏ အဓိကအင်္ဂါရုပ်တစ်ခုဖြစ်ပါသည်။ ဤအခန်းတွင် inheritance နှင့်ပတ်သက်သော အကြောင်းအရာများကို အသေးစိတ်ဖော်ပြသွားမည်ဖြစ်ပါသည်။ inheritance ကို အသုံးပြုရာတွင် derived class ကနေ base class ထဲကို ဝင်ရောက်ကြည့်ရှုခွင့်ရှိ၊ မရှိ ဆိုသည်မှာ အရေးကြီးသော အချက်တစ်ခုဖြစ်ပါသည်။ ဤအကြောင်းအရာကိုလည်း ဤအခန်းတွင် ဆွေးနွေးသွားမည်ဖြစ်ပါသည်။ ထို့အပြင် inheritance ကို အသုံးပြုရာတွင် ရှောင်ကြဉ်သင့်သော အချက်အချို့ကိုလည်း ဤအခန်းတွင် ဆွေးနွေးသွားမည်ဖြစ်ပါသည်။



- derived class (1) နှင့် (2) သည် base class မှ derived ခြင်းဖြစ်သောကြောင့် သူတို့သည် base class ရဲ့ ဂုဏ်သတ္တိများနှင့် အပြုအမူများကို အမွေဆက်ခံရရှိသည်။ ထို့ပြင် property အသစ်များကိုလည်း ထပ်ဆောင်းနိုင်သည်။
- derived class (3) သည်ကတော့ base class နဲ့ derived class (1) တို့ကို ပေါင်းစပ်ထားသော အမွေဆက်ခံမှုအမျိုးအစားကို ပုံဖော်ထားပြီး property အသစ်များကိုလည်း ထပ်ဆောင်းထားသည်။
- derived class (4) ကတော့ base class တွင် derived class (1) နှင့် (2) တို့မှ ပုံစံတူ အပြုအမူများကို ပြန်လည်အသုံးပြုခြင်းအပြင် ကိုယ်ပိုင် property အသစ်များကိုလည်း ထပ်ဆောင်းထားသည်။ ဒီလို inheritance မျိုးကို multiple inheritance လို့ ခေါ်သည်။

Create a Derived Class

ကျွန်တော်တို့ ကျင့်သုံးတဲ့ derived class ကို create လုပ်ရန် base class တစ်ခုကို အရင် create လုပ်ပြီးမှသာ ဖြစ်ပါမည်။ ဥပမာအနေနဲ့ rectangle တစ်ခုရဲ့ area ကို ရှာတဲ့ program တစ်ခုမှာ

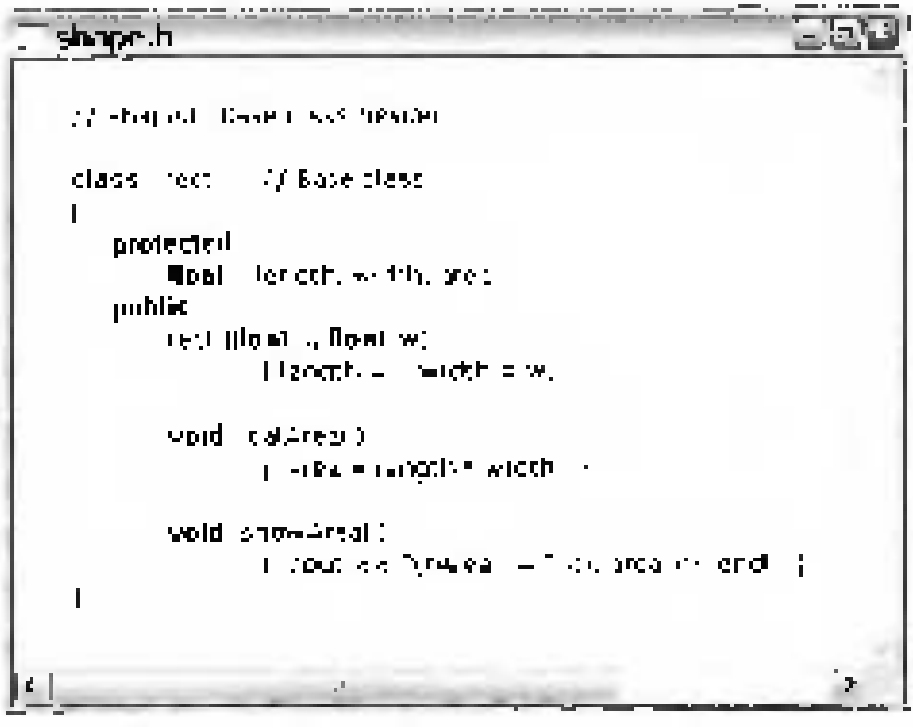
ပုံ (၂၀.၂) တွင်ပြထားသော Ex901.cpp program ကိုကြည့်ပါ။ ဒီ program ဟာ base class တစ်ခုကို အသုံးပြုပြီး
ဧရိယာရှာတဲ့ program ကို run လာရင် enter length နဲ့ Enter width ဆိုပြီး prompt လုပ်ပေးမယ်။
10 နဲ့ 20 ကိုထည့်လိုက်ရင်တော့ area = 200 ဆိုပြီး ပြသပေးမှာဖြစ်ပါတယ်။ ဒီ program ကတော့
ဧရိယာရှာတဲ့ class ဆောက်ထားတဲ့ class ကို base class လုပ်ထားတာပါ။

Ex901.cpp

```
// Inheriting a Class to Create Derived Class
#include <iostream.h>

class rect                    // Base Class
{
        float length, width, area;
    public:
        rect(float length, float width);

        void getarea();

        void showarea()
                { cout<<"\n\nArea = "<<area<<"\n\n"; }
};

int main()
{
        float x,y;

        cout<<"\n\nEnter length ";
        cin>>x;
        cout<<"\nEnter width ";
        cin>>y;

        rect r(x,y);
        r.getarea();
        r.showarea();
        return 0;
}
```

ပုံ (၂၀.၂)

၃။ class declaration တွင် private အတွက် protected ကြေငြာထားမှုကို class rect ၏ shape.h ခေါင်းစီးဖိုင်တွင် header ဖိုင်၌ save ပြင်ပါက private အား protected ဖြင့် ချိတ်ဆက်ထားသော base class ၏ data member များကို derived class များမှ အသုံးပြုနိုင်ရန် ဖြစ်သည်။ base class နှင့် derived class ဆက်သွယ်မှုအပေါ် မူတည်ပြီး shape header ဖိုင်အတွက် listing ကို ပုံ (၃-၇) တွင်ကြည့်ပါ။

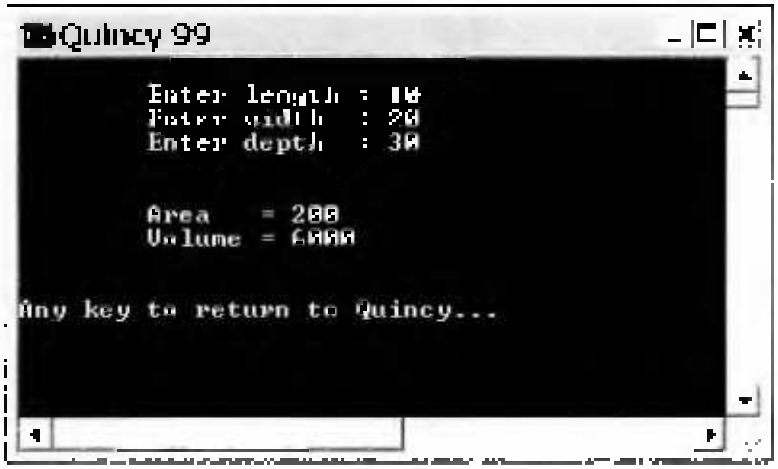


ပုံ (၃-၇)

၄။ class declaration ပါဝင်သော ကျွန်တော်တို့၏ root base class များတွင်ပြုလုပ်ထားသော volume နှင့် derived class အတွက် create လုပ်ဆောင်ချက် Base class ၏ shape.h ခေါင်းစီးဖိုင်တွင် save ထားသောအတိုင်းပါပဲ။ base class ထဲတွင်ပါသော length နှင့် width ကိန်းရှင်များကို derived class များမှ depth data ကိန်းရှင်များနှင့် ဆက်သွယ်ပြီး volume ပြုလုပ်မည့် function ကို call ခေါ်ဆိုသော ပုံ (၃-၈) တွင်ကြည့်ရှုနိုင်သော Ex302.cpp ဖိုင်ကို derived class ဖိုင်တွင် main( ) function ကိုခေါ်ယူကြည့်ပါမည်။ ကျေးဇူးတင်ပါသည်။

```
// Listing 9.2: Using shape.h header
#include <iostream>
#include "shape.h"

class  cube: public rect      // Derived class
{
        float  depth, vol;
    public:
        cube (float  l, float  w, float  d): rect (l, w)
                { depth = d;  }

        void calVol( )
                { vol = length * width * depth;  }

        void showVol( )
                {  cout << "\tVolume = " << vol << endl;  }
};


int  main( )
{
        float  x,y,z;
        cout << "\n\tEnter length : ";        cin >> x;
        cout << "\tEnter width  : ";          cin >> y;
        cout << "\tEnter depth  : ";          cin >> z;
        cout << endl << endl;

        rect   rectan(x,y);
        rectan.calArea( );
        rectan.showArea( );

        cube   box(x,y,z);
        box.calVol( );
        box.showVol( );
        cout << endl;
        return 0;
}
```

Ex902.cpp program ကို run လိုက်ပေးခဲ့ရင် ပုံ (၉-၇) အတိုင်းမြင်ရမှာပါ။



ပုံ (၉-၇)

Ex902.cpp program ကိုစေ့စေ့ကြည့်ပေးပါဦးနော်

- စတုဒ်ဘုံအနေထားတဲ့ #include "shape.h" ရဲ့အခိုင်လိမ်ဘားဆိုရင် Ex902.cpp program ကို compile ဝင်တဲ့အခါမှာ shape.h ဖိုင်ထဲမှာရှိတဲ့ cube အတွက်ဝင်ပြောင်းရောင်း compile လုပ်ပေးသွားမှာဖြစ်ပါတယ်။ x = 10 y = 20 နဲ့ z= 30 ဆိုတဲ့တန်ဖိုးတွေကိုလည်းပေးလိုက်ပါတဲ့။ ဒါကြောင့် derived class cube ကြောင့် object box ကို construct လုပ်ပြီး parameter (3) ကို pass လုပ်ပေးလိုက်တယ်။
- ဒီတော့ rect class နဲ့ derived class ကြောင့် cube ကတို့ x = 10 y = 20 နဲ့ depth = z = 30 ကို အရင်ကလို့ပေးလိုက်တယ်။ ပြီးတော့ rect (x,y) constructor ကပေးကဲ့သို့ shape.h ဖိုင်ထဲက length = l = x = 10 နဲ့ width = w = y = 20 ကို assign လုပ်ပေး လိုက်ပါတယ်။
- box.Calcvol( ) ကို call ပေါ်တဲ့အခါ volume = length * height * depth = 10 * 20 * 30 = 6000 ဖြစ်လာမှာပါပေါ့။ ဘာကြောင့်ဒီလိုလဲဆိုတော့ derived class က base class ထဲက protected data member ဖြစ်တဲ့ length နဲ့ width ကိုတိုက်ရိုက်ကြေညာ အသုံးပြုခွင့်ရပါတယ်။ private ဖိုင်ကြေညာထားရင် အခုလိုဆင်ခြင်ရမှာမဟုတ်ပါဘူး။

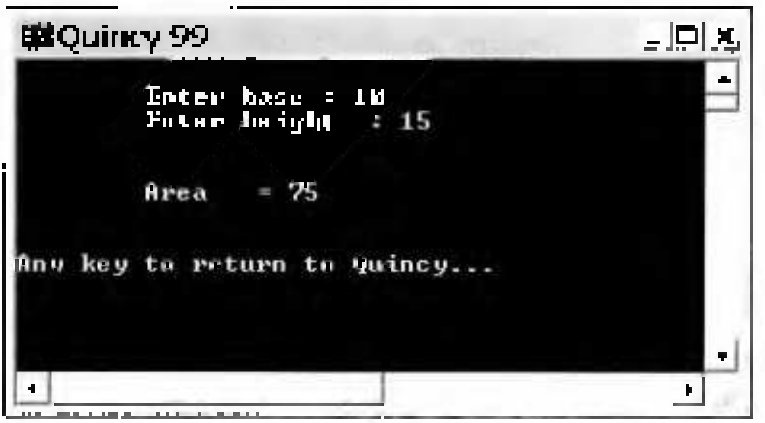
■ Parallel box.ShowVol( ) ကို call ခေါ်ခြင်း volume = 6000 ကို display ပြသခြင်းဖြစ်သော program ကိုရေးသားရန်အတွက် rect နှင့် base class အဖြစ် cube ကို derived class အသစ်တစ်ခုကို ရယူသတ်မှတ်ခြင်းဖြင့် create လုပ်ခြင်း volume ကို ရှာဖွေတွက်ချက်ရန် class inheritance ပုံစံအသုံးပြုခြင်း

Create a Derived Class 'triangle'

➔ Listing 9.3 ကဖော်ပြထားသော program တွင် base class rect ကို အသုံးပြုခြင်းဖြင့် derive လုပ်ထားသော derived class triangle ဖွဲ့စည်းပုံအသစ်ကို အခြေခံ၍ area ကိုရှာဖွေပုံဖြစ်သည်။

```
// Listing 9.3  Creating a derived class 'triangle'
#include <iostream>
#include "shape.h"

class   triangle: public  rect
{
   public
       triangle (float  x=0, float  y=0) : rect (x, y)       {    }

       void   calArea( )
              { area = 0.5* length* width;   }
};

int  main( )
{
       float    x,y;

       cout << "\n\nEnter base . "; cin >> x;
       cout << "\tEnter height  : ";          cin >> y;
       cout << endl << endl;
       triangle  tri(x, y);     tri.calArea( ), tri.showArea( );
       cout << endl;
       return 0;
}
```

Ex003.cpp program ကို run ကြည့်လိုက်ရင် ပုံ (၉-၅) အတိုင်းမြင်ရမှာပါ။



ပုံ (၉-၅)

Inheritance with counter class

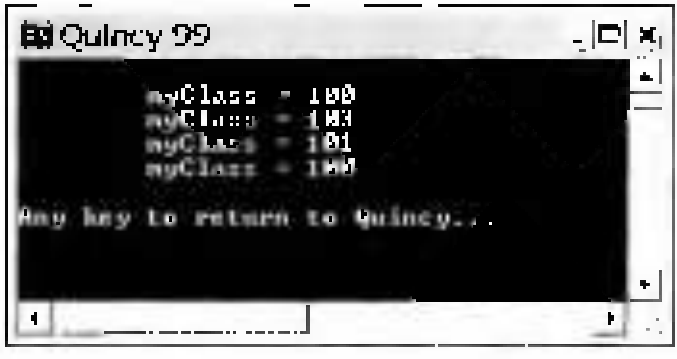
```
// Listing 9.4: Inheritance with counter class
#include <iostream>

class  counter      // Base class
{
   protected:
      unsigned int   count;

   public:
      counter( )              {count = 100;}

      counter(int c)          {count = c;}

      int     getCount( )
              {return count;}
```

```
        counter    operator ++(int )
                {
                    count++;
                    return  counter(count);
                }
};

class countDemo : public counter // Derived class
{
    public:
        counter   operator --(int)
                {
                    count--;
                    return  counter(count);
                }

        counter operator   ( )
                {
                    --count;
                    return  counter(count);
                }
};

int main( )
{
        countDemo    myClass;              // myClass = 100
        cout << "\n\nmyClass = "
             << myClass.get count( );

        myClass++;                         // myClass = 101
        myClass++;                         // myClass = 102
        myClass++;                         // myClass = 103
        cout << "\n\nmyClass = "
             << myClass.getCount( );

        myClass--;                         // myClass = 102
        myClass--;                         // myClass = 101
```

```
    cout << "\n\tmyClass = " << myClass.getCount( );

    --myClass;                          // myClass = 100
    cout << "\n\tmyClass = "
         << myClass.getCount( );

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

Ex904.cpp program ကိုစမ်းသပ်လုပ်ဆောင်ပုံဖော်ပြခြင်း

- ဆောင်ရွက်ချက်: countDeriv class object တစ်ခုဖြစ်တဲ့ myClass ကို define လုပ်လိုက်တဲ့အခါ base class ထဲကအလုပ်လုပ်ဆောင်ချက်တွေ counter( ) constructor က အလုပ်လုပ်သွားပါတယ်။ ဒါပေမဲ့ myClass.count = 100 ကို initialize လုပ်ပေးတဲ့အပိုင်းမှာ myCass.count ကို display လုပ်ပြတဲ့အခါမှာတော့ myClass.getCount( ) function ကို call ခေါ်ပါတယ်။ ဒီ function က base class ထဲက function member တစ်ခုပါ။ count ကို protected လုပ်ထားတဲ့အတွက် base class ကနေတစ်ဆင့် ဝင်ရောက်ပေးပါတယ်။
- myClass++; statement ပြုလုပ်ခဲ့တဲ့ base class ထဲကအလုပ်လုပ်ချက်တွေ counter operator ++( ) function ကနေ myClass.count ကို 1 တိုးပေးသွားပါတယ်။ နိဂုံးအနေနဲ့ (1) အရ ပြောင်းလဲသွားတဲ့အတွက် ရရှိလာမဲ့ myClass.count က 101 ဖြစ်သွားပါပြီ။ myClass ကို decrement လည်းလုပ်ခိုင်းထားပါသေးတယ်။ စဉ်းစားကြည့်ပါ။
- Ex904.cpp program ကို run လိုက်မယ်ဆိုရင် ဒုတိယ (၃) အကြောင်းအတိုင်းဖြစ်ပေါ်မယ်။



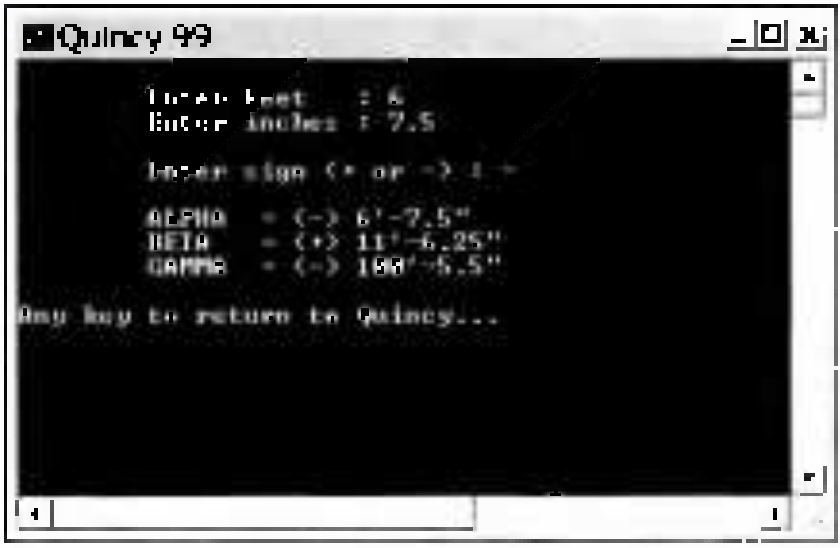
ဒုတိယ (၃)

9.3 A Complex Inheritance

```
// Listing 9.5: A complex example of inheritance
#include <iostream>

class  Length
{
   protected
      int  feet;    float inches;
   public
      Length( )
            { feet= 0; inches= 0;  }

      Length (int ft, float in)
            { feet= ft; inches= in;  }

      void  getLength( )
            {
                  cout << "\n\tEnter feet  : ";       cin >> feet;
                  cout << "\tEnter inches : ";       cin >> inches;
            }

            void  showLength( )
            {
                  cout << feet << "'-" << inches << "\";
            }
};

enum  sign {pos, neg};

class LengthSign : public  Length
{
      char  sign;
public:
      LengthSign( ) : Length( )    { sign = pos; }
```

```
        LengthSign (int ft, float in);
                { feet = ft; inches = in; sign = pos;  }

        LengthSign (int ft, float in, char pos) : Length (ft, in)
                { sign = pos; }

        void  getLength( )
                {
                        Length::getdata( );
                        char  ch;
                        cout << "\n\tEnter sign (+ or -) : ";        cin >> ch;
                        sign = (ch == '+') ? pos  neg;
                }

        void   showLength( );
                {
                        cout << ((sign == pos) ? "(+) " : "(-) ");
                        Length::showLength( );
                }
};

int main( )
{
        LengthSign  alpha;
        alpha.getLength( );
        cout << "\n\tALPHA  = ";              alpha.showLength( );

        LengthSign   beta(11,6.25);
        cout << "\n\tBETA   = ";              beta.showLength( );

        LengthSign   gamma(100,5.5,neg);
        cout << "\n\tGAMMA  = ";              gamma.showLength( );
        cout << endl;
        return 0;
}
```

■ Ex905.cpp program ကို run ထဲတဲ့အခါမှာ ဒုတိယ output ကို ရရှိမှာဖြစ်ပါတယ်။

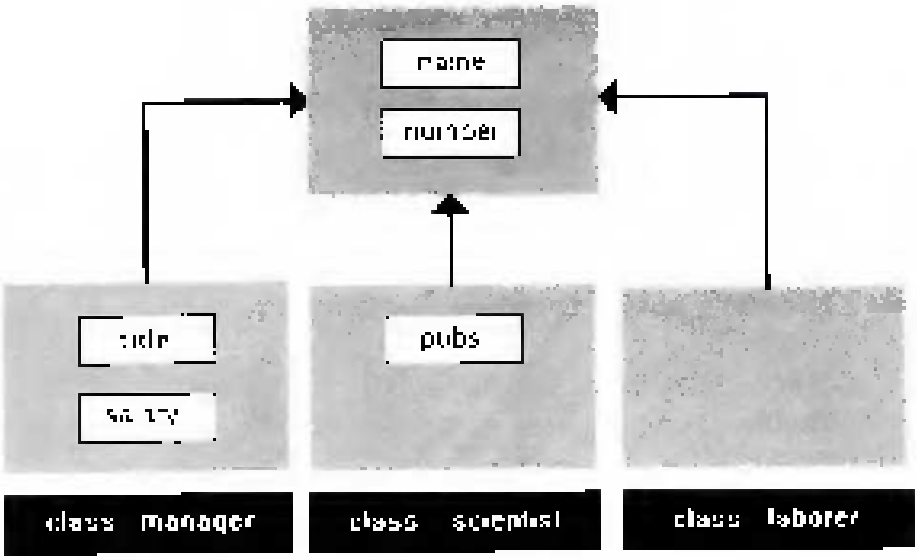


ပုံ (၉-၇)

၉.၇ Class Heirarchies

ဒီ derived class တစ်ခုက base class တစ်ခုထက်ပိုပြီးတော့လည်း ဆင်းသက်လာနိုင်ပါတယ်။ ဒီလို multiple base class တစ်ခုထဲမှာ ဆင်းသက်လာတဲ့အခါမှာ ပိုပြီးရှုပ်ထွေးသွားပါတယ်။ ဥပမာ ၉.၇ ကတော့ employee database တစ်ခုအတွက် အသုံးပြုနိုင်မယ့် class (3) ခုအပါအဝင် class များစွာကို ဖော်ပြထားတာပါ။ base class ဖြစ်တဲ့ employee ထဲမှာ name, identification number စတဲ့ data တွေကို store လုပ်ထားပေးပါတယ်။ employee ရဲ့ derived class ဖြစ်တဲ့ manager class မှာ title နဲ့ salary ဆိုတဲ့ data (2) ခုကိုပဲ store လုပ်ထားပေးပါတယ်။ manager category ရဲ့ data (4) ခုကို ထိန်းသိမ်းပေးထားတဲ့ scientist category ဖြစ်ပြီး pubs ဆိုတဲ့ number of publications စတဲ့ data တစ်ခုကိုပဲ store လုပ်ထားပေးပါတယ်။ laborer category ဖြစ်တဲ့ data တွေအတွက်ကတော့ ဒီအတွက် program ထဲမှာ ထပ်ဖြည့်ပေးထားခြင်း မရှိပါဘူး။

Inheritance ကို



9 (w.r)

Employee Database Model Using Inheritance

```
// Listing 9.6: Employee database model using inheritance
#include <iostream>

const int  LEN = 80;

class employee   // Base class
{
      char   name[LEN];
      int    num;
   public:
      void   getData( )
             {
                   cout << "\n\tEnter name   : ";     cin >> name;
                   cout << "\tEnter number  : ";     cin >> num;
             }
```

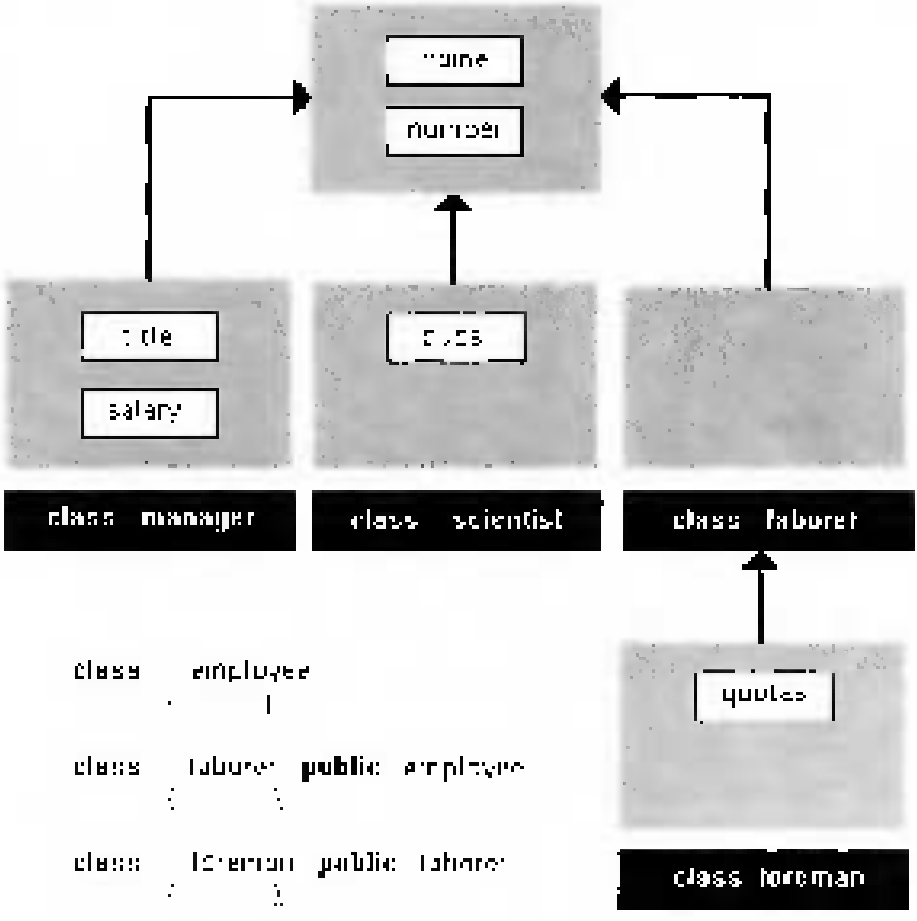
```
        void   putData( )
               {
                    cout << "\n\tName     : " << name;
                    cout << "\n\tNumber   : " << num;
               }
};

class  manager : public employee    // Derived class
{
        char   title[LEN];
        double salary;
    public:
        void  getData( )
               {
                    employee::getData( );
                    cout << "\tEnter title  : ";          cin >> title;
                    cout << "\tEnter salary  ";          cin >> salary;
               }

        void  putData( )
               {
                    employee::putData( );
                    cout << "\n\tTitle    : " << title;
                    cout << "\n\tSalary   : " << salary;
               }
};

class  scientist : public  employee   // Derived class
{
        int   pubs;
    public:
        void   getData( )
               {
                    employee::getData( );
                    cout << "\tEnter number of publications : "; cin >> pubs;
               }
```

```
        void    putData( )
                {
                        employee::putData( );
                        cout << "\n\tNumber of publications : " << pubs;
                }
};

class  laborer  public employee     // Derived class
        {       };


int main( )
{
        manager    ml,
        scientist    sl;
        laborer     ll,

        cout << "\n\nEnter data for manager1",
        ml.getData( );
        cout << "\n\nEnter data for scientist1",
        sl.getData( );
        cout << "\n\nEnter data for laborer1";
        ll.getData( );

        cout << "\n\n\tData on manager1",
        ml.putData( );
        cout << "\n\n\tData on scientist1";
        sl.putData( );
        cout << "\n\n\tData on laborer1";
        ll.putData( );
        cout << endl;

        return 0;
}
```

■ Ex506.cpp program ကိုကြည့်ခြင်းအားဖြင့် employee ကိုပြုလုပ်ထား
သော base class ကိုသုံးပြီးတော့ manager
scientist နှင့် laborer တို့သော derived class object တို့ကို ကြေငြာခြင်းတို့ကို
သော employee နှင့် laborer ကြားမှ ပူးပေါင်းသော class hierarchies ပုံ derived class
များက base class ထဲမှ ပယ်ဖျက်ပစ်သောအချက်များကို ဖော်ပြထားပါသည်
Ex506.cpp program ကို run ကြည့်ခြင်းဖြင့် ပုံ (၇-၉) ကဲ့သို့ ရလဒ်ကိုတွေ့ရမည်ဖြစ်ပါသည်

ပုံ (၇-၉)

၉.၄ Multiple Levels of Inheritance



ပုံ (၉.၁၀)

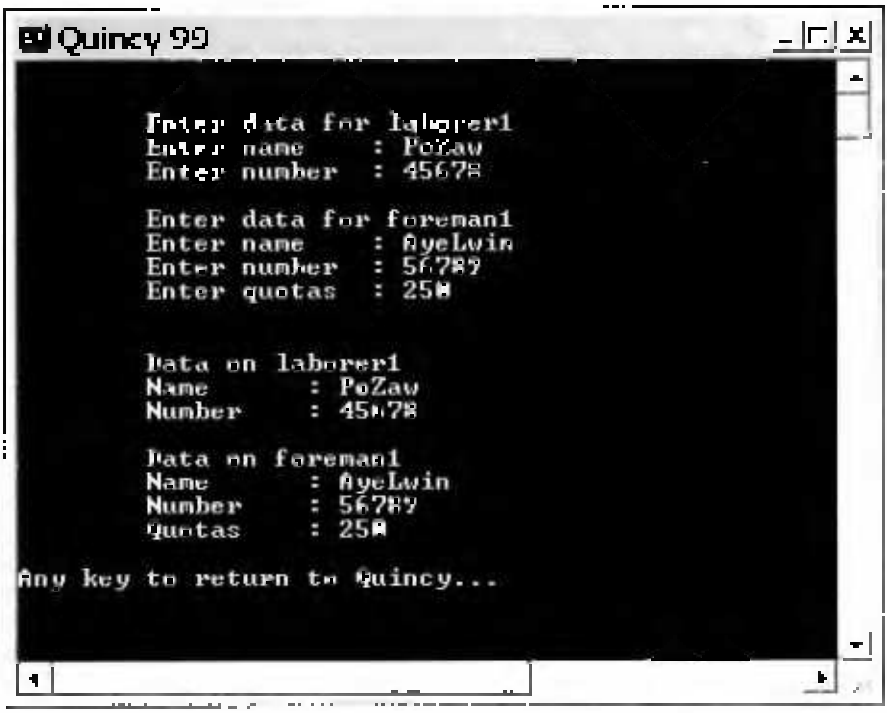
ပုံ (၉.၁၀) တွင်ဖော်ပြထားသည့်အတိုင်း class laborer သည် class employee ၏ derived class တစ်ခုဖြစ်ပြီး foreman class သည် laborer class မှ derive ဆင်းသက်လာသော်လည်း laborer class တွင် data အချို့မပါဝင် foreman class တွင် quotas ဆိုသည့် data ရှိခဲ့သော် employee laborer နှင့် foreman class တို့အတွက် သက်ဆိုင်ရာဖြည့်စွက်ရန် Ex907.cpp program ပေးထားလိုက်ပါသည်။

```
// Listing 9.7  Multiple levels of inheritance
#include <iostream>
const int  LEN = 80;

class   employee   // Base class
{
        char   name[LEN];
        int    num;
   public:
        void   getData( )    {
                    cout << "\n\tEnter name   : ";      cin >> name;
                    cout << "\tEnter number  : ";       cin >> num;
               }

        void   putData( )    {
                    cout << "\n\tName      : " << name;
                    cout << "\n\tNumber    : " << num;
               }
};

class  manager : public  employee   // Derived class
{
        char     title[LEN];
        double   salary;
   public:
        void   getData( )    {
                    employee::getData( );
                    cout << "\tEnter title   : ";          cin >> title;
                    cout << "\tEnter salary    ";         cin >> salary;
               }

        void   putData( )    {
                    employee::putData( );
                    cout << "\n\tTitle     : " << title;
                    cout << "\n\tSalary    : " << salary;
               }
};
```

```
class  scientist  public  employee    // Derived class
{
        int  pubs;
    public:
        void  getData( )      {
                    employee::getData( );
                    cout << "\tEnter number of publications  ";  cin >> pubs;
                }

        void  putData( )     {
                    employee::putData( );
                    cout << "\nNumber of publications : " << pubs;
                }
};

class  laborer  public  employee     // Derived class
        {       };

class  foreman : public  laborer     // Derived class
{
        int  quotas;
    public:
        void  getData( );     {
                    employee::getData( );
                    cout << "\tEnter quotas  : ";        cin >> quotas;
                }

        void  putData( )     {
                    employee::putData( );
                    cout << "\nQuotas       : " << quotas;
                }
};

int  main( )
{
        laborer  l1;
        foreman  f1;
```

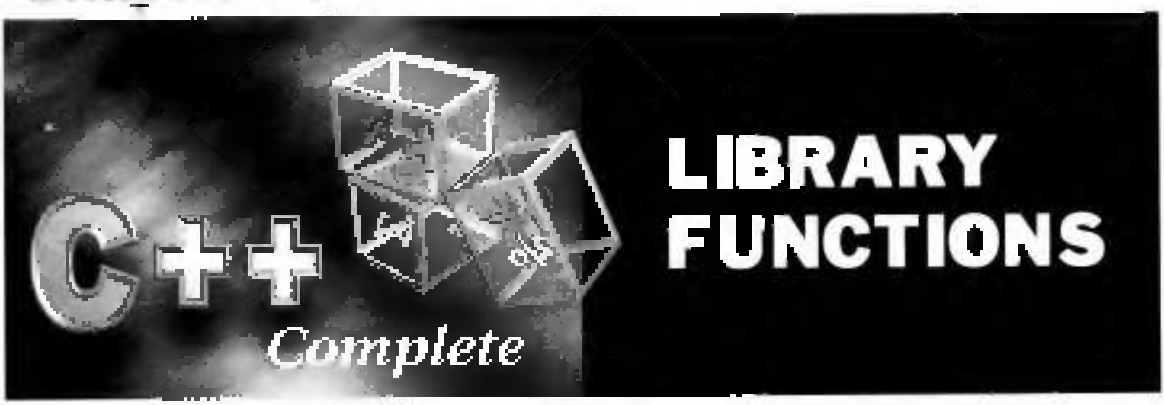
```
    cout << "\n\n\tEnter data for laborer1";
    l1.getData( );
    cout << "\n\tEnter data for foreman1";
    f1.getData( );

    cout << "\n\n\tData on laborer1";
    l1.putData( );
    cout << "\n\n\tData on foreman1";
    f1.putData( );
    cout << endl;
    return 0;
}
```

■ Ex007.cpp program ကို run တဲ့အခါမှာ ပုံ (၉-၁၁) အတိုင်းတွေ့ရမှာပါ။

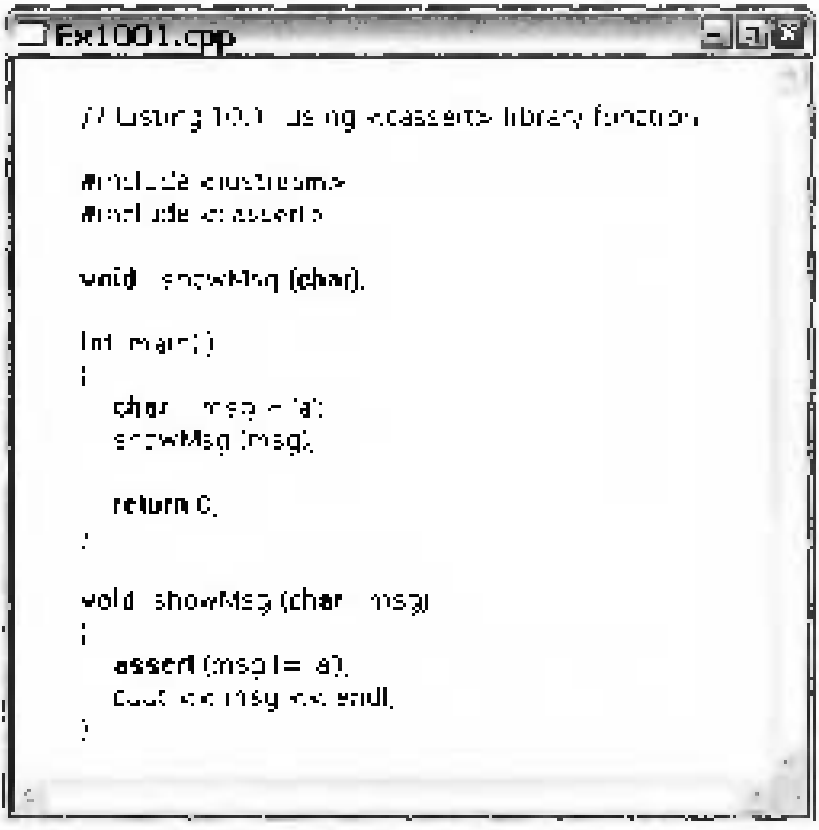


ပုံ (၉-၁၁)

Chapter 10

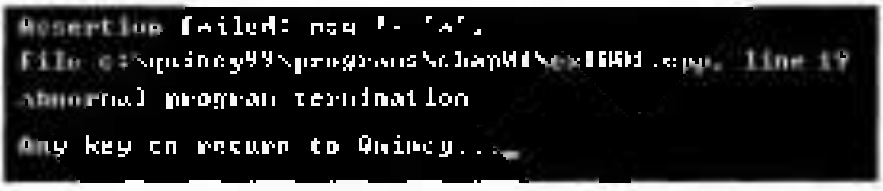


Standard C function တွေဟာ Standard C++ library မှာပါဝင်ပါပေမဲ့ ရှေ့ပိုင်းမှာတုန်းက လျှံခဲ့ကော်တို့နဲ့ program တွေမှာ standard C function တစ်ချို့ကိုအသုံးပြုခဲ့ပါတယ်၊ လျှံခဲ့ကော်ကဖန်တီးခဲ့တဲ့ Standard C++ မှာပါဝင်တဲ့ library function ကတော့လုံးဝကိုတင်ပြဖို့ရည်ရွယ်ချက်မပါပေမဲ့ အထိုးပုံတဲ့ library function တွေတိုးတင်ပြဖို့ကိုပါပေမယ်။ ဥပမာ လျှံခဲ့ကော်ကိုဖန်တီးတဲ့ C++ program တွေမှာ debugging code တိုးထည့်ပေးဖို့နှင့် <cassert> header ဖိုင်ထည့်ပေးခြင်းအားဖြင့် assert( ) function ကိုအသုံးပြုနိုင်ပါမယ်၊ main( ) function ဘက်ဆီ pass လုပ်ပို့လိုက်တဲ့ data ကို assert( ) function မှာစစ်ဆေးလိုက်လို့ false ဖြစ်ရင်ဒဲ့ argument ကို display လုပ်ပြီးပြီး program ကအလိုအလျောက်ရပ်စဲသွားပါလိမ့်မယ်။ program မှာအသုံးပြုပြီးတဲ့ assert macro တွေကို အသုံးမလိုတော့ပြီဆိုရင်နောက်ဖွင့်ခဲ့ ဘဲဆွဲခဲ့လိုက်ပြီးကော်ဆရာလိုပါဘူး NDEBUG macro ကို program မှာတည်ပြီး compile လုပ်ပေးရင် assert macro တွေ disable ဖြစ်သွားပါပြီး ပုံ (၁၀.၁) မှာ ဒေါ်ပြထားတဲ့ Ex1001.cpp program မှာ <cassert> header ဖိုင်အသုံးပြုထားပါတယ်။ ဆက်လေ့ကြည့်ပါ။



ပုံ (၁၀.၁)

Ex1001.cpp program ကိုရေးထားကြည့်ရင် main( ) ထဲက showMsg( ) function ထဲကနေ pass လုပ်လိုက်တဲ့ data ဟာ character 'a' ဖြစ်ပါတယ်၊ ဒီ 'a' ကို assert( ) function ထဲမှာ (msg != 'a') လား လို့စစ်ဆေးလိုက်တဲ့အခါမှာ true ဆိုရင် message prompt မလုပ်ပါဘူး၊ false ဆိုရင် assert( ) function ရဲ့ argument ကို prompt လုပ်ပြပြီး program stop ပြီးသွားမှာပါ။ အခုနှ Ex1001.cpp program ကို run ကြည့်ပါဆိုတဲ့ ပုံ (၁၀.၂) မှာပြထားသလိုဘဲပေါ်လာမှာဖြစ်ပါတယ်



ပုံ (၁၀.၂)

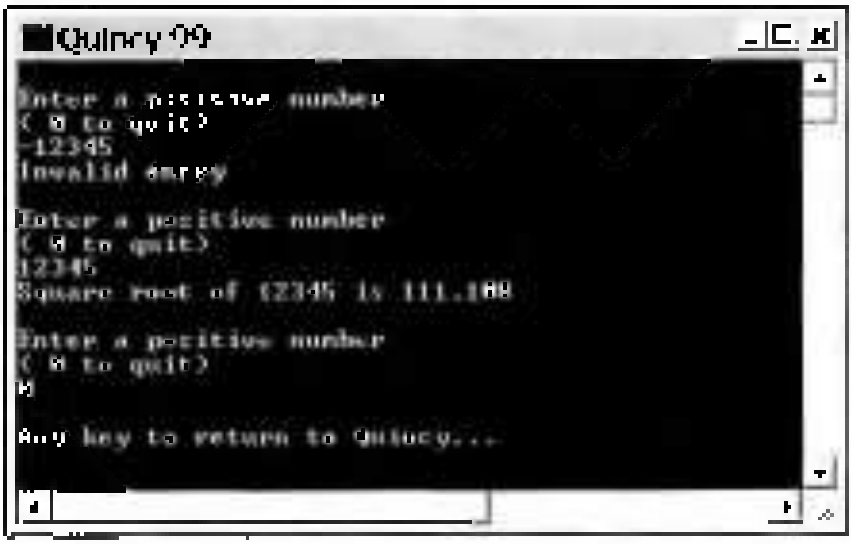
၁၀.၁ <cerrno>

[illegible] <cerrno> header [illegible] program [illegible] error [illegible] errno variable [illegible] define [illegible] errno value [illegible] assign [illegible] 0 [illegible] assign [illegible] Ex1002.cpp program [illegible] run [illegible]

```
Ex1002.cpp

// Listing 10.2 Using <cerrno> - sqrt function
#include <iostream>
#include <cmath>
#include <cerrno>

int main()
{
    double num;
    do {
            errno = 0;
            cout << "\nEnter a positive number\n"
                 << "0 to quit\n";
            cin >> num;
            if (num != 0)
            {
                double s = sqrt(num);

                if (errno == 0)
                        cout << "Square root of " << num
                             << " is " << s << endl;
                else
                        cout << "Invalid entry" << endl;
            }
        } while (num != 0);
    return 0;
}
```

ပုံ (၁၀.၂)

ဥပမာတစ် statement ကိုအသုံးမပြုထားပါဘူး။ num value အားသုတ်သင်ဖြုတ်ခြင်းများ ပြီးမှသာ ဆုံး
အဆိုးဝေးတော့ program stop ဖြစ်သွားမှာပါ။ အများ error variable ကို မ ဖြည့်ခင် အပေါ်ပါလို
ယုံကြည်အားထားရ error check ပုံစံတော်ကြည့်အတွက်ပါဖြစ်ပါသည်။ ပုံ (၁၀.၄) မှ program ကို run ခြင်းအကြောင်း
program stop လုပ်ခြင်းကို ပြထားပါသည်။



ပုံ (၁၀.၄)

၁၀.၂ <cmath>

၁။ Ex10b2.cpp program ကဲ့သို့ <cmath> header ဖိုင်မှ ပြဋ္ဌာန်းထားသည့် sqrt()
function ကိုအသုံးပြုနိုင်ပါသည်။ အခြားသော math header များကို အောက်ပါ (၁၀.၁) ဇယားတွင်ကြည့်ပါ။

ဇယား (၁၀.၁) <cmath> Functions

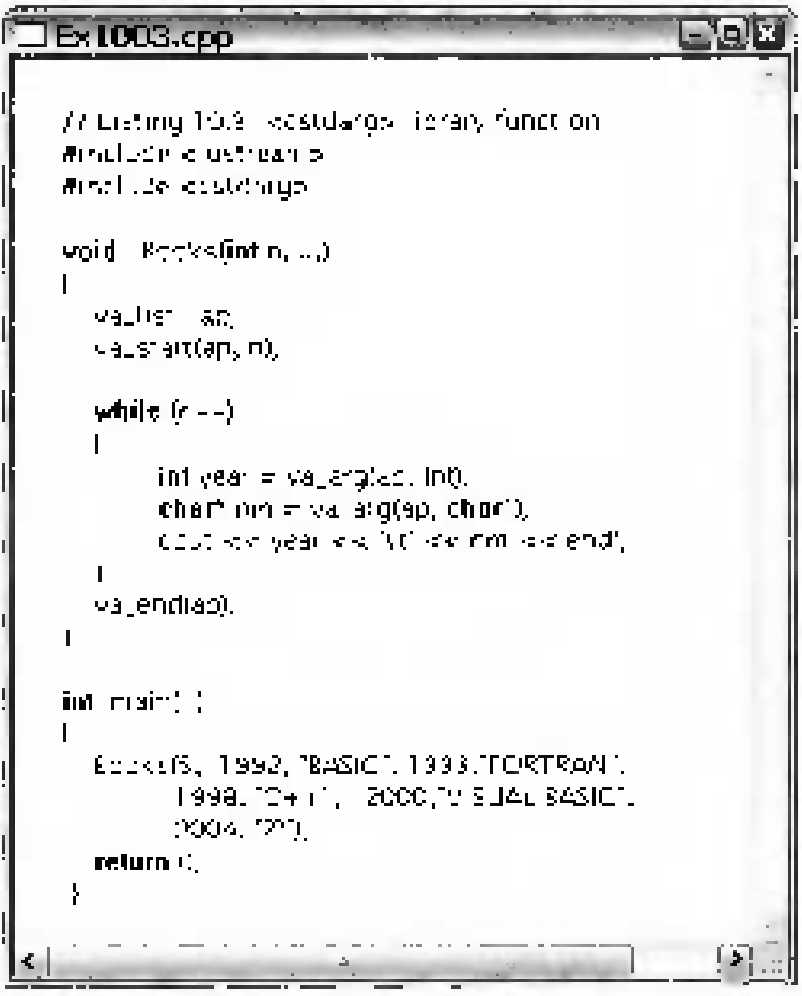
Function	Returns
double acos (double x);	Arc cosine of x

Library Functions ကို

double asin (double x);	Arc sin of x
double atan (double x);	Arc tangent of x
double atan2 (double y, double x);	Arc tangent of y/x
double ceil (double x);	Smallest integer not < x
double cos (double x);	Cosine of x
double cosh (double x);	Hyperbolic cosine of x
double exp (double x);	Exponential value of x
double fabs (double x);	Absolute value of x
double floor (double x);	Largest integer not > x
double log (double x);	Natural logarithm of x
double log10 (double x);	Base-10 logarithm of x
double pow (double x, double y);	x raised to the power of y
double sin (double x);	Sin of x
double sinh (double x);	Hyperbolic sine of x
double sqrt (double x);	Square root of x
double tan (double x);	Tangent of x
double tanh (double x);	Hyperbolic tangent of x

၁၀.၃ <cstdarg>

C++ program တစ်ခုမှာ variable argument list တစ်ခုကို function တစ်ခုသို့ ပေးပို့လိုပါက <cstdarg> header ဖိုင်ကို ထည့်သွင်းပေးရပါမည်။ Ex1003.cpp program မှာဆိုရင် Books() function ကို call ခေါ်ပြီး passed argument ဖြစ်တဲ့ n ကို n = 5 ကို ပေးပို့ထိုး ထားပြီး variable list ကို called function ဖြစ်တဲ့ print() ထဲသို့ ပေးပို့ထားပါသည်။ ဒီ program ရဲ့ အဓိကအချက်မှာ variable list type (va_list _ap) starting variable argument macro (va_start(ap, n)) ၊ variable argument list ကိုဖတ်ယူနိုင်တဲ့အတွက် ပေးပို့ပြုလုပ်ရမယ့် va_arg(ap, char*) နဲ့ va_arg(ap, int) တို့ဖြစ်ပြီး data အဆုံးကိုဖတ်ရှုနိုင်ဖို့အတွက် ကြေငြာပေးရမယ့် macro ပြီး va_end(ap) ဖြစ်ပါသည်။ ဥပမာကြည့်ပါ။



ပုံ (၁၀-၄)

Ex1003.cpp program ကို ပုံ (၁၀-၅) ကဲ့သို့ run ပြုလုပ်လိုက်သော် output ထွက်ပေါ်လာမည်ဖြစ်သည်။

ပုံ (၁၀-၅)

০০.৫ <cstdlib>

<cstdlib> header এ standard library function সমূহকে ৫ টি ভাগে ভাগ করা হয়েছে (1) প্রোগ্রাম নিয়ন্ত্রণ (2) Numerical functions (3) Memory allocation functions (4) system functions এবং (5) random number generation function সমূহ। নিম্নে কয়েকটি <cstdlib> Numerical functions এবং <cstdlib> Memory Allocation function এর তালিকা দেওয়া হলো।

[illegible] <cstdlib> Numerical Functions

Function	Returns
int abs (int i);	The absolute value of i
int atoi (const char *s);	The integer value of the string
long atol (const char *s);	The long integer value of the string
float atof (const char *s);	The float value of the string

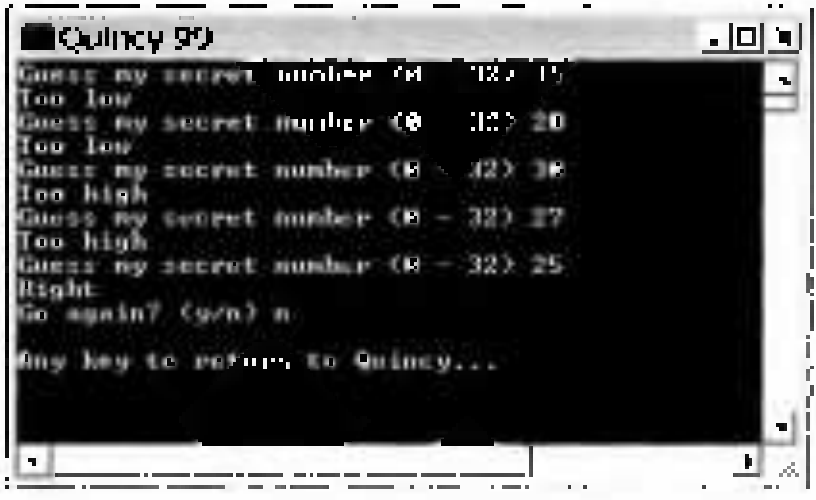
[illegible] <cstdlib> Memory Allocation Functions

Function	Returns
void *calloc (int sz, int n);	Address of buffer or 0
void *malloc (int sz);	Address of buffer or 0
void free (void *buf);	Nothing

```
// Listing 10.4 : Random number generation functions
#include <iostream>
#include <cstdlib>
#include <ctime>
```

```
int main( )
{
        srand  (time(0));
        char   ans;
        int    num;
        do  {
                int   assNum = rand( ) % 33;   // Choose a secret number
                do  {
                        cout << "Guess my secret number (0 - 32) ";
                        cin >> num;

                        // Report the status of the guess.
                        cout << (num < assNum ? "Too low" :
                        num > assNum ? "Too high" : "Right") << endl;
                }  while (num != assNum);

                cout << "Go again? (y/n) ";          cin >> ans;
        }  while (ans == 'y');

        return 0;
}
```

Example program of a guessing game



Sample run

၁၀.၅ <cstring>

<cstring> header ဖိုင်တွင်ပုံစံ null-terminated character array များအတွက် ပြုလုပ်ပေးသော function များဖြစ်သည်။ ၎င်းတို့တွင် (၁) comparison functions (၂) copy functions (၃) concatenation functions (၄) strlen( ) function နှင့် (၅) memset( ) function တို့ပါဝင်သည်။ ၎င်း function prototype များကို ဖော်ပြထားသည်။

```
int     sbcmp (const  char *s1, const  char *s2);
int     strncmp (const  char *s1, const  char *s2, int  n);
char    *strcpy (char  *s1, const  char *s2);
char    *strncpy (char  *s1, const  char *s2, int  n);
int     strlen (const  char *s);
char    *strcat (char *s1, const  char *s2);
char    *strncat (char  *s1, const  char *s2, int  n);
char    *memset (void  *s, int  c, int  n)
```

```
// Listing 10.5  Using <cstring>
#include <iostream>
#include <cstring>

int  main( )
{
      int   len;
      char  msg[ ] = "Wrong.";
      char  pwd[40];

      cout << "Password? ";
      cin >> pwd;
      len = strlen(pwd);
```

```
    if (strcmp(pwd, "C++") == 0)
        strcpy(msg, "Right.");
    cout << msg << " \nYou typed "
        << len << " characters\n";
    return 0;
}
```

Ex1005.cpp program ကို run (သို့) ကြည့်ပါက အောက်ပါအတိုင်း ရလာမှာဖြစ်ပါသည်။

ပုံ (၁၀.၄)

၁၀.၆ <ctime>

<ctime> header ဟာမှာတော့ time နဲ့ date တို့ဆက်သွယ်နေတဲ့ structure တစ်ခု၊ data type တစ်ခုနဲ့ function အများအပြားကို declare လုပ်ပေးစေနိုင်ပါတယ်၊ structure ကိုးကောက်ပုံအနေနဲ့ ကြည့်ပေးပါတယ်

```
struct  tm
{
    int     tm_sec;     // seconds (0-61)
    int     tm_min;     // minutes (0-59)
    int     tm_hour;    // hours (0-23)
    int     tm_mday;    // day of the month (1-31)
```

```
    int     tm_mon;     // months since January (0-11)
    int     tm_year;    // years since 1900
    int     tm_wday;    // days since Sunday (0-6)
    int     tm_yday;    // days since January 1 (0-365)
    int     tm_isdst;   // Daylight Saving Time flag
};
```

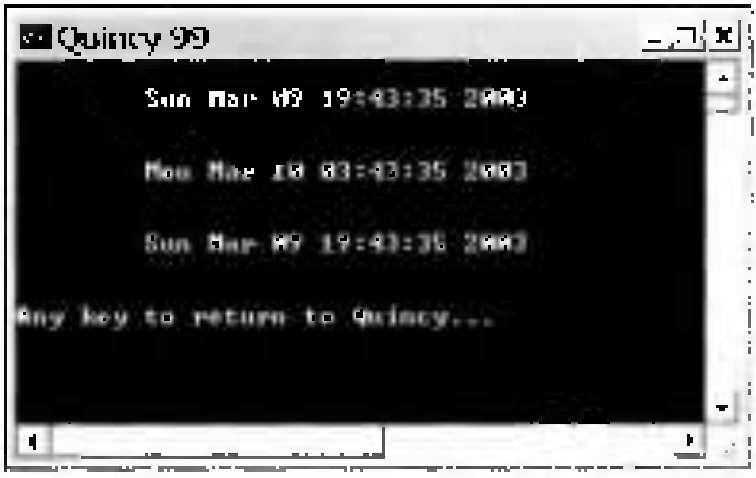
<ctime> function summary; function prototype begin: copyright

```
char        *asctime (const struct tm *tm);
char        *ctime (const time_t *t);
double      difftime (time_t t1, time_t t2);
struct tm   *gmtime (const time_t *t);
struct tm   *localtime (const time_t *t);
time_t      mktime (struct tm *tm);
time_t      time (time_t *t);
```

Ch1006.cpp

```
// Listing 10.6 Using <ctime>
#include <iostream>
#include <ctime>

int main()
{
    time_t now = time(0);

    cout << "It's " << asctime(localtime(&now)) << endl;
    cout << "GMT is " << asctime(gmtime(&now)) << endl;
    cout << "Time: " << ctime(&now) << endl;

    return 0;
}
```

Ex1006.cpp program ကို run လိုက်ပေးခြင်းဖြင့် ပုံ (၁၀.၀၇) ပါအတိုင်းရလဒ်ကိုတွေ့ရမည်



ပုံ (၁၀.၀၇)