



Microsoft  
Windows

```
1: // This program asks the user for information about an object of
2: // class history, and then writes this object to HISTORY.DAT
3: #include <fstream.h>
4:
5: class history
6: {
7: protected:
8: char name[30];
9: char degree[30];
10: int age;
11: public:
12: void getData(void)
13: {
14: cout << "Enter name : "; cin >> name;
15: cout << "Enter age : "; cin >> age;
16: cout << "Enter degree : "; cin >> degree;
17: }
18: };
19:
20: main( )
21: {
22: history person; // Create a history
23: person.getData(); // Get data for history
24:
25: ofstream outfile("HISTORY.DAT");
26: outfile.write((char *) &person, sizeof(person));
27: }
```



**Vol:2**

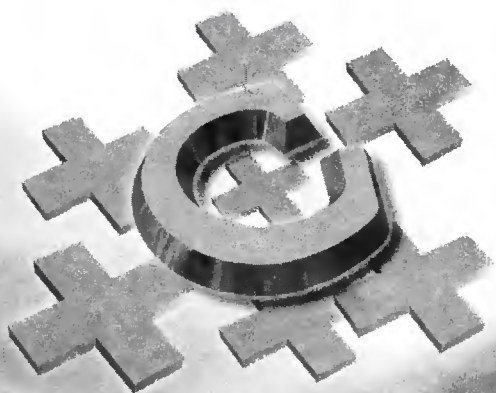
*Complete*

**AUNG MYINT (M.E., AUSTRALIA)**

# CHAPTER 5

## C++ REFERENCES

5.1.	Reference is an Alias	10
5.2.	Passing References	13
5.3.	Returning References	16
5.4.	Using const References	18
5.5.	C++ Preprocessor	21



**Complete**

# CHAPTER 6

## C++ CLASSES

6.1.	Create A Simple Class	33
6.2.	Using the Inline Function	37
6.3.	Constructor and Destructor	41
6.4.	Classes and const	45
6.5.	static Members	46
6.6.	Overloaded Constructors	49
6.7.	Conversion Constructors	61
6.8.	Member Conversion Functions	63
6.9.	Using Friends	67
6.10.	Friend Functions	71



**Complete**



# CHAPTER 7

## C++ ARRAYS

7.1.	Array Basics	74
7.2.	Initializing an Array	76
7.3.	Processing an Array	81
7.4.	Passing Array to a Function	84
7.5.	Multidimensional Arrays	91
7.6.	Arrays of Objects	97



**Complete**

## CHAPTER 8

# OVERLOADED OPERATORS

8.1.	Overloaded ++ Operator	108
8.2.	Overloaded + Operator	111
8.3.	Overloading + with a Nonmember Function	127
8.4.	Overloading the Assignment += Operator	130
8.5.	Overloaded Relational Operators	134
8.6.	Overloading == Operators	138
8.7.	Overloading - Operators	139
8.8.	Overloading [ ] Operators	141
8.9.	Overloading -> Operators	143



**Complete**

## **CHAPTER 9**

# **INHERITANCE**

9.1.	Create a Derived Class	147
9.2.	A Complex Inheritance	156
9.3.	Class Heirarchies	158
9.4.	Multiple Levels of Inheritance	163

## **CHAPTER 10**

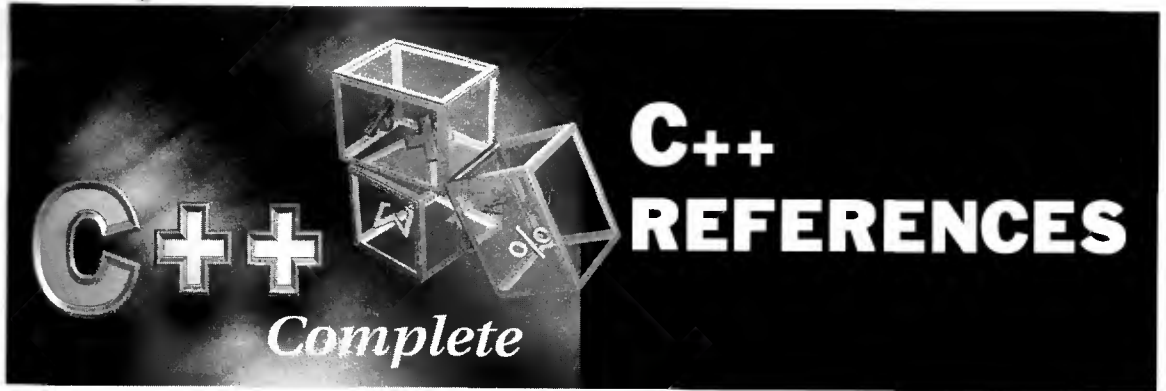
# **LIBRARY FUNCTIONS**

10.1.	<cerrno>	169
10.2.	<cmath>	170
10.3.	<cstdarg>	171
10.4.	<cstdlib>	173
10.5.	<cstring>	175
10.6.	<ctime>	176



# **Complete**

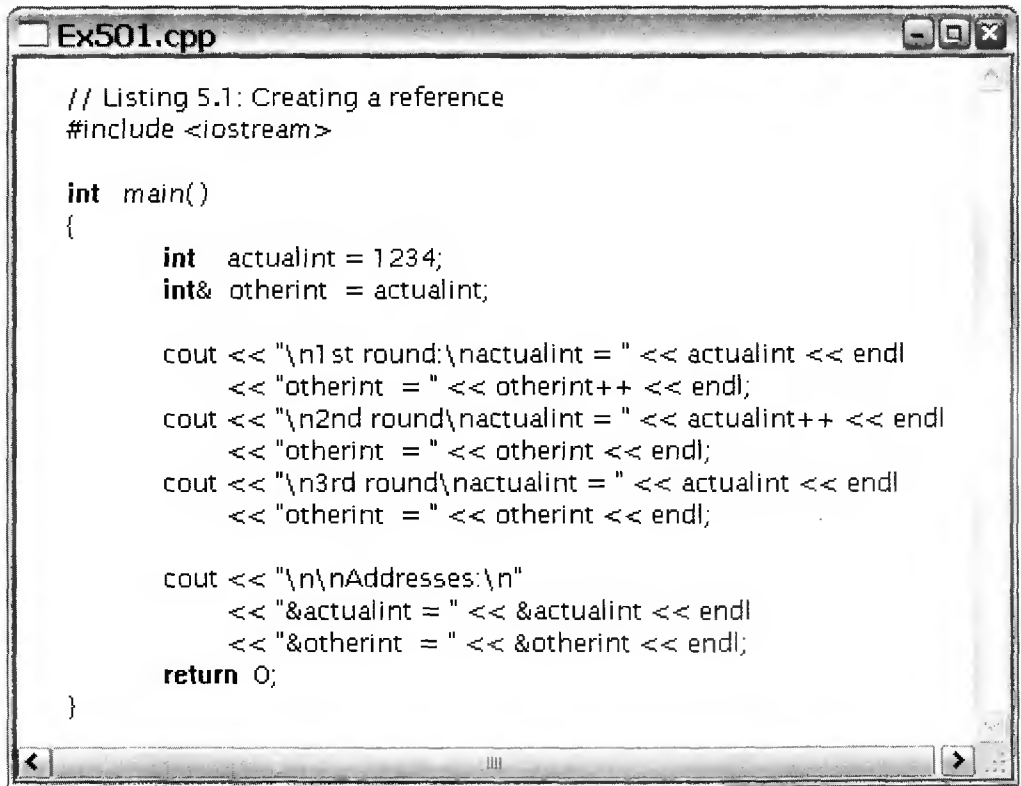
## Chapter 5



reference variable ဆိုတာ သူနဲ့ address ချင်းတူတဲ့ နောက် variable တစ်ခုရဲ့ နာမည်ကဲ့တစ်ခုပါ။ reference တွေကို pointer တွေလိုပဲ function တွေမှာအသွားအပြန် pass လုပ်ပေးလို့ရပါတယ်။ pointer နဲ့မတူတဲ့အချက်က သူ့ကို declare လုပ်တဲ့အခါမှာ actual object နဲ့ reference ကို initialize လုပ်ပေးရပါမယ်။ reference တစ်ခုကို initialize လုပ်ပြီးသွားရင် reference value ကို ပြောင်းလို့မရတော့ပါဘူး။ reference မှာ pointer လို null reference ဆိုတာမရှိပါဘူး။ & (ampersand) operator ဟာ variable တစ်ခုကို reference variable ဖြစ်အောင်သတ်မှတ်ပေးပါတယ်။ ဥပမာ `int actualint;` နဲ့ `int& otherint = actualint;` ဆိုတဲ့ statement တွေမှာ `actualint` ကို integer variable လို့အရင်ကြေငြာပြီး `otherint` ဟာ `actualint` အတွက် reference variable ပါပဲ။ ကောင်းပြီ ၊ reference variables နဲ့ပတ်သက်တဲ့ program တွေကိုလေ့လာကြည့်ရအောင်။

## ၅.၁ Reference is an Alias

၁။ ပုံ (၅. ၁) မှာဖော်ပြထားတဲ့ Ex501.cpp program ကိုလေ့လာကြည့်ရင် စေ့ချင်း actualint ကို 1234 ဂဏန်းနဲ့ initialize လုပ်ပေးပြီး variable အတွက် reference ဖြစ်တဲ့ otherint ဆိုတဲ့ alias variable တစ်ခုကို define လုပ်ပါမယ်။ အခုနေ ဒီ variable တွေရဲ့တန်ဖိုးတွေကို display လုပ်ကြည့်ရင် 1234 တွေကိုပဲ တွေ့ရမှာပါ။ ကောင်းပြီ ၊ reference ကို increment လုပ်ကြည့်ပါမယ်။ ဒါဆိုရင် reference တန်ဖိုးဟာ 1234 ဖြစ်သွားသလို actualint လည်း 1234 ဖြစ်မှာပဲ။ actualint ကို increment လုပ်ရင်လည်း otherint မှာတိုးနေမှာပါ။



```
// Listing 5.1: Creating a reference
#include <iostream>

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

    cout << "\n1st round:\nactualint = " << actualint << endl
         << "otherint = " << otherint++ << endl;
    cout << "\n2nd round\nactualint = " << actualint++ << endl
         << "otherint = " << otherint << endl;
    cout << "\n3rd round\nactualint = " << actualint << endl
         << "otherint = " << otherint << endl;

    cout << "\n\nAddresses:\n"
         << "&actualint = " << &actualint << endl
         << "&otherint = " << &otherint << endl;
    return 0;
}
```

ပုံ (၅. ၁)

၂။ ပုံ (၅. ၂) ဟာဆိုရင် Ex501.cpp program ကို run ပြထားတာဖြစ်ပါတယ်။ actualint နဲ့ otherint တို့က ဘယ်ဟာပဲပြောင်းပြောင်း ၊ ဘယ်နေရာမှာပဲပြောင်းပြောင်း အကုန်လိုက်ပြောင်းတာကို output မှာလေ့လာ ကြည့်ရင်သိပါလိမ့်မယ်။



```
Quincy 99
1st round:
actualint = 1235
otherint  = 1234

2nd round
actualint = 1235
otherint  = 1235

3rd round
actualint = 1236
otherint  = 1236

Addresses:
&actualint = 0x241fff6c
&otherint  = 0x241fff6c

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

ပုံ (၅.၂)

## Reducing Complex Notation by References

ပုံ (၅.၃) မှာရေးထားတဲ့ Ex502.cpp program ဟာဆိုရင် structure member တွေကို အရှည်ကြီး မရေးရအောင် reference အသုံးပြုပြီး ရေးတဲ့နည်းဖြစ်ပါတယ်။ ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- main( ) ထဲမှာရေးထားတဲ့ pl ဟာ array pointer တစ်ခုပါ။ players[ ] ကို point လုပ်နေပါတယ်။ players element တစ်ခုချင်းဟာ struct type အမျိုးအစားဖြစ်ပါတယ်။ while (pl->playerNo != 0) ဆိုတဲ့ expression ဟာ players[0] က first member ကို zero လားလို့မေးတာပါ။ မဟုတ်ပါဘူး ၊ 001 ဖြစ်ပါတယ်။ ဒီတော့ while loop ထဲဝင်လာပါပြီ။
- pl က point လုပ်နေတဲ့ players[0] ရဲ့ reference ကို rp လို့ define လုပ်ပေးပါတယ်။ ဒီတော့ rp.name ကို print လုပ်ရင် ပထမဆုံး player ရဲ့နာမည်ပေါ်လာမှာပါ။ ထိုနည်းတူပဲ players[0] ရဲ့ birthdate ကို print မလုပ်ခင် rp.birthdate ရဲ့ reference ကို rd လို့ define လုပ်ပေးရအောင်။ data type က struct Date ပေါ့။ ဒီတော့ rd.month ကို print

လုပ်ရင် `rp.birthdate.month` သို့မဟုတ် `pl->birthdate.month` ကို `print` လုပ်တာနဲ့ အတူတူဖြစ်ပါလိမ့်မယ်။ `rd.month` နဲ့အတူ `rd.day` ၊ `rd.year` တို့ကိုတွဲပြီး `print` လုပ်ပေးရင် ပထမဆုံး `player` ရဲ့ `birthdate` ကိုရမှာပါ။

- `player` တစ်ယောက်ပြီးတစ်ယောက် `print` လုပ်ပေးချင်ရင် `array element` နေရာကိုပြောင်းဖို့ အတွက် `pl` ကို `increment` လုပ်ပြီး `while loop` ကို ထပ်ပတ်ခိုင်းရမှာပါပဲ။ `pl->playerNo`

```

Ex502.cpp

// Listing 5.2: Reducing complex notation
#include <iostream>

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

struct playerRec
{
    int    playerNo;
    char*  name;
    Date   birthdate;
};

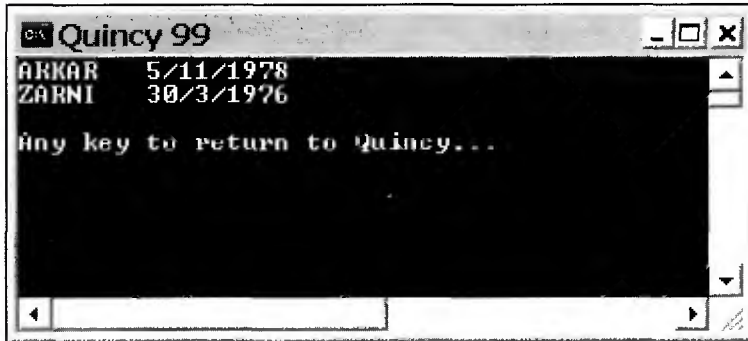
playerRec players[] = {
    {001,"ARKAR",{5,11,1978} },
    {002,"ZARNI",{30,3,1976} },
    {0}
};

int main()
{
    playerRec* pl = players;
    while (pl->playerNo != 0)
    {
        playerRec& rp = *pl;
        cout << rp.name << '\t';
        Date& rd = rp.birthdate;
        cout << rd.month << '/' << rd.day << '/'
             << rd.year << endl;
        pl++;
    }
    return 0;
}

```

ပုံ (၅.၃)

= 0 ဖြစ်သွားရင် while loop ထဲမဝင်တော့ပဲ program stop လုပ်ပေးလိုက်ပါပြီ။ ပုံ (၅. ၄) မှာ Ex502.cpp program ကို run ပြထားပါတယ်။



ပုံ (၅. ၄)

## ၅.၂ Passing References

၁။ ပုံ (၅. ၅) မှာဖော်ပြထားတဲ့ Ex503.cpp program ဟာဆိုရင် main( ) function ကနေ pass လုပ်ပေးလိုက်တဲ့ data တွေကို called function ကနေ reference တွေအနေနဲ့လက်ခံရယူပြီး အလုပ်ဆက်လုပ်သွားပုံကိုတင်ပြထားပါတယ်။ calling function မှာ x နဲ့ y တို့ကို argument တွေအဖြစ် pass လုပ်ပေးတာကို called function ထဲမှာ y ၊ x ဖြစ်အောင်ပြောင်းပေးမှာပါ။

၂။ Ex503.cpp program ကိုလေ့လာကြည့်ရင်

- main( ) ထဲမှာ int x နဲ့ y တို့ကို 15 နဲ့ 500 လို့ initialize လုပ်ပေးထားပါတယ်။ နောက်ပြီး x ၊ y တန်ဖိုးတွေကို display လုပ်ခိုင်းပါတယ်။ ပြီးတော့ရင် swap( ) function ကနေ x ၊ y ကို pass လုပ်ပေးပါတယ်။ swap( ) function ထဲမှာ int& i = x = 15 ၊ int& j = y = 500 အနေနဲ့ဝင်သွားပါပြီ။ ဒီ function ထဲမှာပဲ i နဲ့ j တို့ကိုနေရာပြောင်းပေးပါတယ်။ ဒီတော့ x = 500 နဲ့ y = 15 ဖြစ်သွားပြီလေ။ main( ) ပြန်ရောက်တဲ့အခါ x ၊ y ကို display လုပ်ခိုင်းရင် ကွန်ပျူတာမှာ x = 500 ၊ y = 15 လို့ပေါ်နေပါလိမ့်မယ်။

```
EX503.cpp
// Listing 5.3: Passing references
#include <iostream>

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 (၅.၅)

၃။ ပုံ (၅.၆) မှာ Ex503.cpp program ကို run ပြထားပါတယ်။ 15 နဲ့ 500 တို့က pass လုပ်ပေးမယ့် data တစ်စုံပါ။ swap( ) function ကနေအပြန်မှာ x နဲ့ y value တို့ဟာ နေရာချင်းပြောင်းသွားပါပြီ။

```
Quincy 99
BEFORE SWAP:
15 500

AFTER SWAP:
500 15

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

ပုံ (၅.၆)



# Passing Structure Data References

၁။ data structure တစ်ခုကို pointer အသုံးပြုပြီး function တစ်ခုကနေ တစ်ခုဆီ pass လုပ်တဲ့အခါမှာ လွယ်ကူသလို reference ကိုအသုံးပြုရင်လည်း လွယ်ကူတာကိုတွေ့ရပါတယ်။ ပုံ (၅.၇) မှာဖော်ပြထားတဲ့ Ex504 program မှာဆိုရင် main( ) function ကနေ structure data တစ်ခုကို pass လုပ်ပြထားပါတယ်။

Ex504.cpp

```
// Listing 5.4: Passing references
#include <iostream>

struct sides
{
    int length, width;
};

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

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

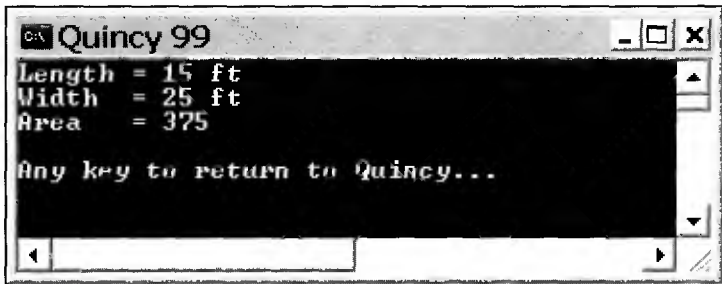
int main()
{
    sides rect = {15,25};

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

ပုံ (၅.၇)

၂။ Ex504.cpp program ကိုလေ့လာကြည့်ရင် main( ) ထဲမှာ struct type rect ကို create လုပ်ပြီး length နဲ့ width တို့ကို initialize လုပ်ပေးထားပါတယ်။ ပြီးတော့ရင် calArea( ) function ကနေ rect ကို

pass လုပ်ပေးပါတယ်။ rect ဟာ calArea( ) function ထဲကို sides & s = rect အနေနဲ့ဝင်သွားပါပြီ။ ဒီ function ထဲမှာပဲ l နဲ့ w တို့ကို reference တွေဖြစ်အောင် define ထပ်လုပ်ပေးပါတယ်။ ဒီတော့ l နဲ့ w နဲ့ l\*w ချိုးကို display လုပ်ခိုင်းရင်ကွန်ပျူတာမှာ length , width နဲ့ area တန်ဖိုးတွေကိုမြင်ရပါလိမ့်မယ်။ ပုံ (၅. ၈) မှာ Ex504.cpp program ကို run ပြထားပါတယ်။ 15 နဲ့ 25 တို့က pass လုပ်ပေးမယ့် structure data တစ်စုံ ပြစ်ပါတယ်။ 375 က rect.length နဲ့ rect.width တို့ မြှောက်လို့ရတဲ့အဖြေပါ။



ပုံ (၅. ၈)

## ၅.၃ Returning References

၁။ ကျွန်တော်တို့တွေ function ထဲကို reference တွေ pass လုပ်နည်းကို Ex504.cpp program မှာ လေ့လာခဲ့ပြီးပါပြီ။ အခုတစ်ခါ called function ကနေ reference တစ်ခုကို return ပြန်ပေးတဲ့နည်းကိုလေ့လာမှာပါ။ ပုံ (၅. ၉) မှာဖော်ပြထားတဲ့ Ex505.cpp program မှာ getLength(choice) ဆိုတဲ့ reference ကို main( ) function ဆီ return ပြန်ပေးထားပါတယ် ၊ လေ့လာကြည့်ပါ။

၂။ Ex505.cpp program ကိုလေ့လာကြည့်ရင်

- struct type array တစ်ခုဖြစ်တဲ့ shapes[ ] ကို program စတင်ချင်းမှာ define လုပ်ထားပါတယ်။ main( ) ထဲမှာ int type choice ကို define လုပ်ပြီး do loop ထဲကိုဝင်ပါတယ်။ choice ဟာ 1 ၊ 2 ၊ 3 ၊ 4 တစ်ခုခုဖြစ်ရင် if block statement ထဲကိုဝင်လို့ရပါတယ်။ ဒါဖြင့် choice = 3 ကို ရိုက်ထည့်လိုက်မယ်ဆိုပါစို့။

```
Ex505.cpp

// Listing 5.5: Returning references
#include <iostream>
struct sides
{
    int length, width;
};

sides shapes[] = {
    { 5, 15},
    {12, 27},
    {31, 75},
    {45, 90},
};

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

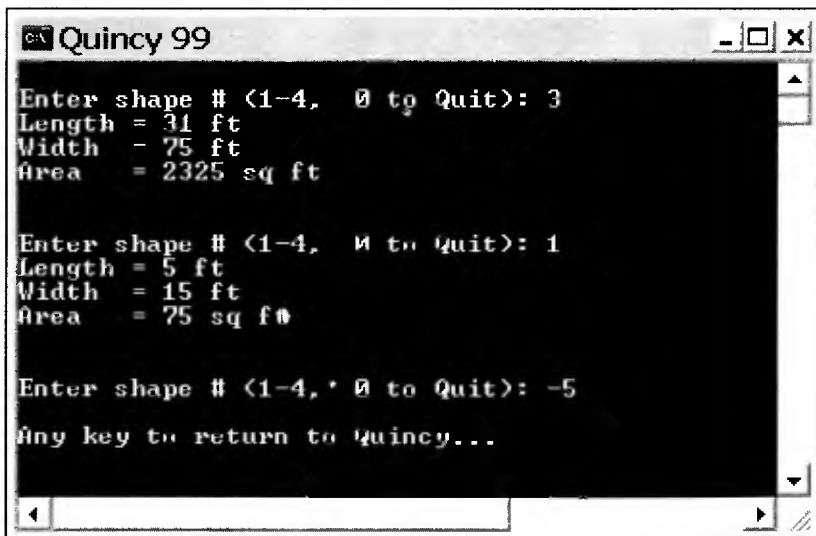
int main()
{
    int choice;
    do {
        cout << "\nEnter shape # (1-4, 0 to Quit): ";
        cin >> choice;
        if (choice > 0 && choice < 5) {
            sides& rs = getLength(choice);
            int& rl = rs.length;
            int& rw = rs.width;
            cout << "Length = " << rl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

ပုံ (၅.၉)

- ဒါဆိုရင် if block ထဲဝင်လာပြီး sides& rs = getLength(choice); ဆိုတဲ့ statement ကနေ getLength( ) function အတွက် reference ကို rs လို့ define လုပ်ပေးပါတယ်။

data type က sides ပါ။ အခုနေ `getLength( )` function ကို call ခေါ်ပြီး choice ကို pass လုပ်ပေးတဲ့အခါမှာ called function ထဲကို `i = choice = 3` အနေနဲ့ဝင်သွားပါပြီ။ called function က return လုပ်ပေးတာက `shapes[i-1]` သို့မဟုတ် `shapes[2]` ဖြစ်ပါတယ်။

- `main( )` ကိုပြန်ရောက်လာတဲ့အခါမှာ `rs = getLength(3) = shapes[2]` ဖြစ်သွားပါလိမ့်မယ်။ ဒီတော့ `rs.length = 12` နဲ့ `rs.width = 27` ဖြစ်နေပါပြီ။ `rl = rs.length` နဲ့ `rw = rs.width` လို့ define ထပ်လုပ်ပြီး `rl`၊ `rw` နဲ့ `rl*rw` တို့ကို `print` လုပ်မယ်ဆိုရင် `shapes[2]` ရဲ့ `length`၊ `width` နဲ့ `area` တို့ကို `display` လုပ်ပြပါလိမ့်မယ်။
- `do loop` အတွက်မှာ choice ကိုစစ်ကြည့်ရင် `True` ဖြစ်တာကြောင့် `do loop` ထဲကိုပြန်ဝင်ပါလိမ့်မယ်။ ဒီတစ်ခါ choice = 1 ကိုရိုက်ထည့်ရင် `shapes[0]` အတွက်အဖြေကိုထုတ်ပေးမှာပါ။ ပြီးရင် `do loop` ထဲကို တတိယအကြိမ်ထပ်ဝင်ပါဦးမယ်။ choice = -5 ကိုရိုက်ထည့်ရင် `do loop` ထဲကိုမဝင်တော့ပဲ `program exit` ဖြစ်သွားပါပြီ။ ပုံ (၅. ၁၀) ကိုကြည့်ပါ။



```

Quincy 99
Enter shape # <1-4, 0 to Quit>: 3
Length = 31 ft
Width = 25 ft
Area = 2325 sq ft

Enter shape # <1-4, 0 to Quit>: 1
Length = 5 ft
Width = 15 ft
Area = 75 sq ft

Enter shape # <1-4, 0 to Quit>: -5
Any key to return to Quincy...
```

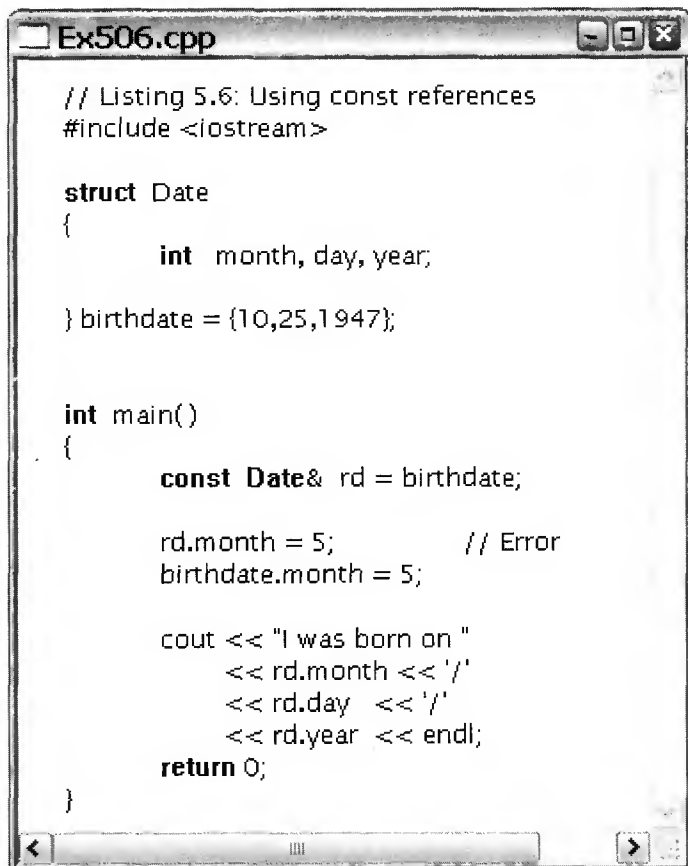
ပုံ (၅. ၁၀)

## ၅.၄ Using const References

၁။ `const` ဆိုတဲ့ specifier ကို reference တစ်ခုမှာထည့်ပေးမယ်ဆိုရင် `main( )` program ထဲမှာ အဲဒီ object ကို reference ကနေပြင်ခိုင်းလို့မရတော့ပါဘူး။ object ကိုယ်တိုင်မှာပဲပြင်ရပါမယ်။ ပုံ (၅. ၁၁) မှာရေးထားတဲ့

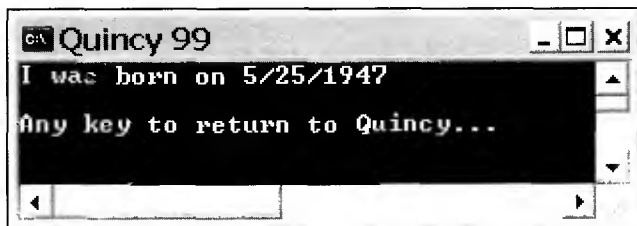


Ex506.cpp program မှာ `rd.month = 5;` ဆိုတဲ့ statement ပါနေလို့ error ဝင်နေပါတယ်။ `birthdate` object မှာပြင်ရင်တော့ အမှားမဖြစ်ပါဘူး။ အဲဒါဟာ `const specifier` ကြောင့်ပါပဲ။



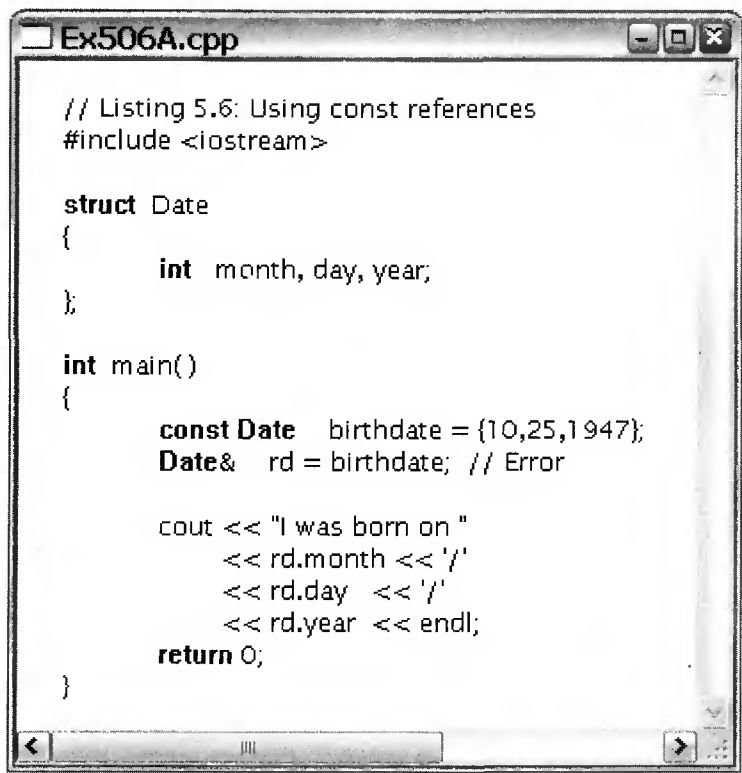
ပုံ (၅. ၁၁)

၂။ `rd.month = 5;` statement ကိုဖြုတ်ပြီး Ex506.cpp ကို run မယ်ဆိုရင် month နေရာမှာပြင်ထားတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၅. ၁၂) ကိုကြည့်ပါ။



ပုံ (၅. ၁၂)

၃။ **const** နဲ့ပတ်သက်လို့ ပြောစရာနောက်တစ်ခုက **const object** လို့ကြေငြာထားတဲ့ဥစ္စာတစ်ခုကို **non-constant reference** တစ်ခုနဲ့ **initialize** လုပ်ပေးလို့မရပါဘူး။ ဥပမာ ပုံ (၅. ၁၃) မှာရေးထားတဲ့ **program** ကို **run** ကြည့်ရင် မ **run** ပါဘူး။ ဘာပြုလို့လဲဆိုတော့ **Date birthdate = {10, 25, 1947}** ကို **const** လုပ်ထားတဲ့အတွက် **Date& rd = birthdate** ဆိုတဲ့ **initialization** ကိုလုပ်လို့မရပါဘူး။ **rd** က **non-constant reference** ဖြစ်နေလို့ပါပဲ။ **const** ကိုဖျက်လိုက်ရင်တော့ **program run** ပါလိမ့်မယ်။



```
// Listing 5.6: 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;
}
```

ပုံ (၅. ၁၃)

၄။ ပုံ (၅. ၁၄) မှာဖော်ပြထားတဲ့ **Ex507.cpp program** ဟာ **const reference** တစ်ခုကို **parameter** အနေနဲ့ **pass** လုပ်နည်းကို ဖော်ပြထားပါတယ်။ ဒီ **program** ကို **run** မယ်ဆိုရင် ပုံ (၅. ၁၂) ကအဖြေပဲရမှာပါ။ တစ်ကယ်လို့ **main( )** ထဲက **birthdate.month = 5;** ဆိုတဲ့ **statement** ကိုဖျက်ပြီးတော့ **rd.month = 5;** ဆိုတဲ့ **statement** ကို **displayDate( ) function body** ထဲမှာထည့်ရေးပြီး **run** မယ်ဆိုရင် ဒီ **program run** မှာမဟုတ်ပါဘူး ၊ သတိပြုပါ။

```
Ex507.cpp

// Listing 5.7: Passing const references
#include <iostream>

struct Date
{
    int month, day, year;
} birthdate = {10,25,1947};

void displayDate(const Date& rd)
{
    cout << "I was born on "
         << rd.month << '/'
         << rd.day  << '/'
         << rd.year  << endl;
}

int main()
{
    const Date& rd = birthdate;

    birthdate.month = 5;
    displayDate(rd);

    return 0;
}
```

ပုံ (၅.၁၄)

## ၅.၅ C++ Preprocessor

၁။ C++ processor ရဲ့ရည်ရွယ်ချက်က ကျွန်တော်တို့တွေရေးတဲ့ C++ program က source code တွေကို compile မလုပ်ခင်မှာ preprocessor ကနေအရင်ဖတ်ပြီးတော့ လိုအပ်တဲ့ translation တစ်ချို့ကိုကြိုတင် ပြုလုပ်ပါလိမ့်မယ်။ ဥပမာ program comment တွေ ၊ white space အပိုတွေပါရင် compiler ဆီမပို့ခင်မှာ ဖြုတ်ချပေးပါလိမ့်မယ်။ macro definition တွေကိုလည်း translate လုပ်ပါလိမ့်မယ်။ အဲဒါတွေလုပ်ပြီးသွားရင် source code အသစ်ကိုရမှာပါ။ ဒီ source code အသစ်ကိုမှ compiler က compile လုပ်မှာဖြစ်ပါတယ်။

preprocessor ရဲ့ output ဟာ compiler ရဲ့ input ဖြစ်ရပါမယ်။ ဒီသဘောတရားကနေ preprocessing directive တွေကို အသုံးပြုဖို့ဖြစ်လာတာပါ။ # (pound) သင်္ကေတနဲ့စထားတဲ့ code တွေကို preprocessing directive တွေလို့ခေါ်ပါတယ်။ အဲဒါတွေကို ဇယား (၅. ၁) မှာဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

ဇယား (၅. 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 display error message

## Including Files: #include

```
C++ program တစ်ခုမှာ header ဖိုင်တွေကို နည်း (2) နည်းနဲ့ထည့်ပေးလို့ရပါတယ်။
#include <iostream>
#include "menus.h"
```

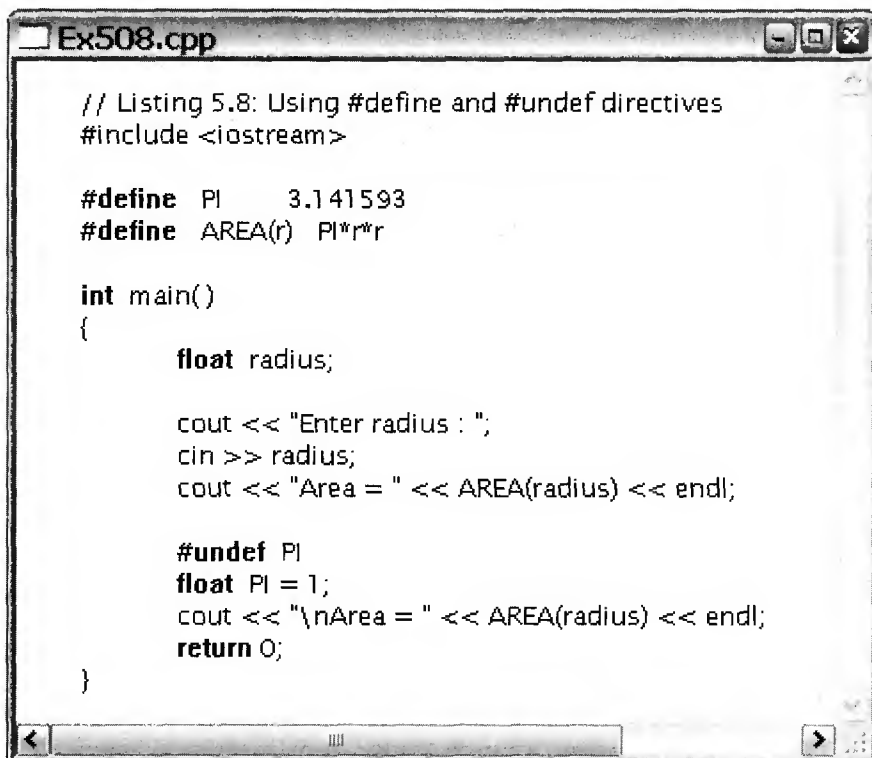
ပထမနည်းမှာဖိုင် name ကို <(less than) နဲ့ >(greater than) သင်္ကေတတွေနဲ့ပိတ်ပေးထားပါတယ်။



compiler system ရဲ့ header ဖိုင်တွေအများကြီးထဲက iostream ဖိုင်ကို preprocessor ကရှာပေးဖို့ ပြောတာပါ။  
ခုတ်ယူနည်းမှာ ဖိုင် name ကို double quotation marks ( " " ) တွေနဲ့ပိတ်ပေးထားပါတယ်။ ဒီအဓိပ္ပါယ်က  
compile လုပ်မယ့် application source code တွေမှာပါဝင်တဲ့ header ဖိုင် menus ကို include လုပ်ပေးပါလို့  
ဆိုလိုပါတယ်။ အဲဒီမှာမတွေ့ရင် compiler system ထဲကိုလိုက်ရှာပါလိမ့်ဦးမယ်။

## Macros: #define and #undef

#define processing directive ရဲ့ပုံစံကအခုလိုမျိုးပါ။ macro definition တစ်ခုကိုရေးမယ်ဆိုရင်  
#define MAXI 100 သို့မဟုတ် #define Square(x) x\*x ဆိုတာမျိုးဖြစ်ပါတယ်။ #define directive  
definition ကို ပြန်ဖျက်ချင်ရင် ဥပမာ #undef MAXI လို့ရေးလိုက်တာနဲ့ macro definition ပျက်သွားပါပြီ။  
ပုံ (၅. ၁၅) မှာ #define directive ရေးနည်းကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



```
// Listing 5.8: Using #define and #undef directives
#include <iostream>

#define PI    3.141593
#define AREA(r) PI*r*r

int main()
{
    float radius;

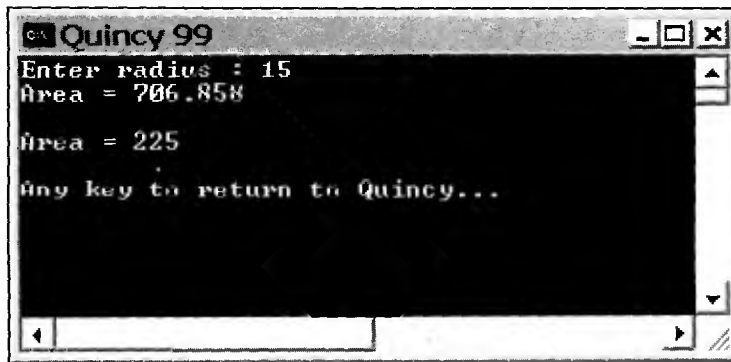
    cout << "Enter radius : ";
    cin >> radius;
    cout << "Area = " << AREA(radius) << endl;

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

ပုံ (၅. ၁၅)

## ၂။ Ex508.cpp program ကို trace လုပ်ကြည့်မယ်ဆိုရင်

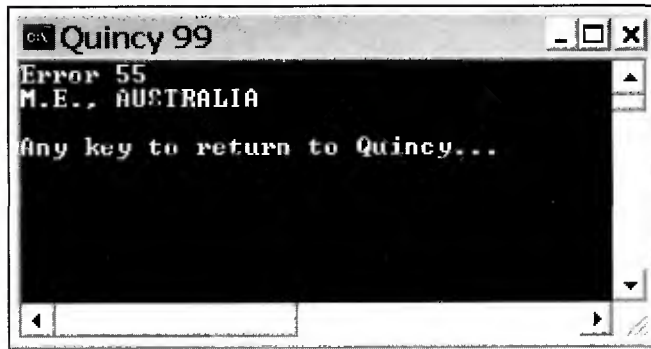
- program ထိပ်ဆုံးမှာ macro (2) ခုကို define လုပ်ထားပါတယ်။ PI နဲ့ AREA(r) ပါ။ main( ) function ထဲမှာ radius အတွက် data ကိုတောင်းတဲ့အခါကျရင် 15 ကိုရိုက်ထည့်ပေးမယ်ဆိုပါစို့။ ဒါဆိုရင်  $AREA(radius)$  ကနေ  $PI*r*r = 3.141593*15*15 = 706.858$  ကိုတွက်ပေးမှာပါ။
- `#undef PI;` ဆိုတဲ့ statement ဟာ PI ရဲ့ macro definition ကို cancel လုပ်ပစ်လိုက်ပါပြီ။ ဒါကြောင့် PI ကို new value တစ်ခုနဲ့ assign လုပ်လို့ရတာပါ။ ကောင်းပြီ ၊  $PI = 1$  ဆိုပါစို့။ ဒါဆိုရင်  $PI*r*r = 1*15*15 = 225$  လို့ နောက်တစ်မျိုးတွက်ပေးပါလိမ့်မယ်။
- ပုံ (၅. ၁၆) ဟာဆိုရင် Ex508.cpp program ကို စောစောကပြောတဲ့ data ဝဲထည့်ပြီး run ပြထားတာဖြစ်ပါတယ်။



ပုံ (၅. ၁၆)

## Stringizing and Concatenation Operators

၁။ macro definition တစ်ခုမှာ stringizing operator ( # ) တစ်ခု ဥပမာ `#n` ကိုထည့်ရေးထားမယ်ဆိုရင် argument `n` ကို string အနေနဲ့ပြောင်းပေးမှာပါ။ argument တွေကိုဆက်ပေးချင်ရင် concatenation operator ( ## ) ကိုအသုံးပြုရပါမယ်။ ပုံ (၅. ၁၇) မှာ ဒီ operator (2) ခုအသုံးပြုပုံကို program ရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



ပုံ (၅. ၁၈)

## Compile-time Conditionals Directives

၁။ ဒီ directive အမျိုးအစားတွေဟာဆိုရင် program ထဲကဘယ် code statement တွေကိုပဲ compile လုပ်ပြီး ဘယ်ဟာတော့မလုပ်ပါနဲ့လို့ ထိန်းချုပ်ပေးဖို့အတွက်အသုံးပြုပါတယ်။ `#if ... #endif` ၊ `#if ... #else ... #endif` သို့မဟုတ် `#if ... #elif ... #else ... #endif` ဆိုတဲ့ directive တွေထဲက ကြိုက်တာကိုအသုံးပြုပြီး compile-time condition ကို control လုပ်လို့ရပါတယ်။ ပုံ (၅. ၁၉) က Ex5010.cpp program မှာ `#if ... #endif` နဲ့ `#if defined ... #endif` ဆိုတဲ့ conditional directive (2) ခု အသုံးပြုနည်းကိုရေးပြထားပါတယ်။ လေ့လာကြည့်ပါ။

၂။ Ex5010.cpp program ကို trace လုပ်ကြည့်မယ်ဆိုရင်

- program ရဲ့ထိပ်ဆုံးမှာ macro (4) ခုအတွက် `#define` directive တွေကိုရေးထားပါတယ်။ `#define DEBUG` ကို 1 လို့သတ်မှတ်ပေးထားပေမယ့် `#define CHECK` မှာ `define` လုပ်ရုံသက်သက်ပဲ လုပ်ထားတာကိုသတိပြုကြည့်ပါ။
- `main( )` ထဲဝင်လာတဲ့အခါကျရင် `#if DEBUG` နဲ့တွေ့ပါလိမ့်မယ်။ ဆိုလိုတာက `#if (true)` ဖြစ်တဲ့အတွက် `#if` block `main( )` ထဲဝင်လာပြီး data တောင်းရင် ရိုက်ထည့်မယ်။ နောက်ပြီး တွက်စရာချက်စရာကိုတွက်ချက်ပြီး `#endif` နဲ့တွေ့ရင် `#if` block ထဲကနေပြန်ထွက်လာပါပြီ။
- အခုတစ်ခါ `#if defined CHECK` နဲ့တွေ့ရင် `true` ဖြစ်တဲ့အတွက် `#if defined` block ထဲ ဝင်လာပြီး `#undef` ကနေ PI macro ကို delete လုပ်ပစ်ပါလိမ့်မယ်။ ဒီ program ကို run ခဲ့ရင်အဖြေက ပုံ (၅. ၁၆) ကအဖြေနဲ့အတူတူပါပဲ။

```
Ex5010.cpp

// Listing 5.10: Compile-time conditionals
#include <iostream>

#define PI      3.141593
#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

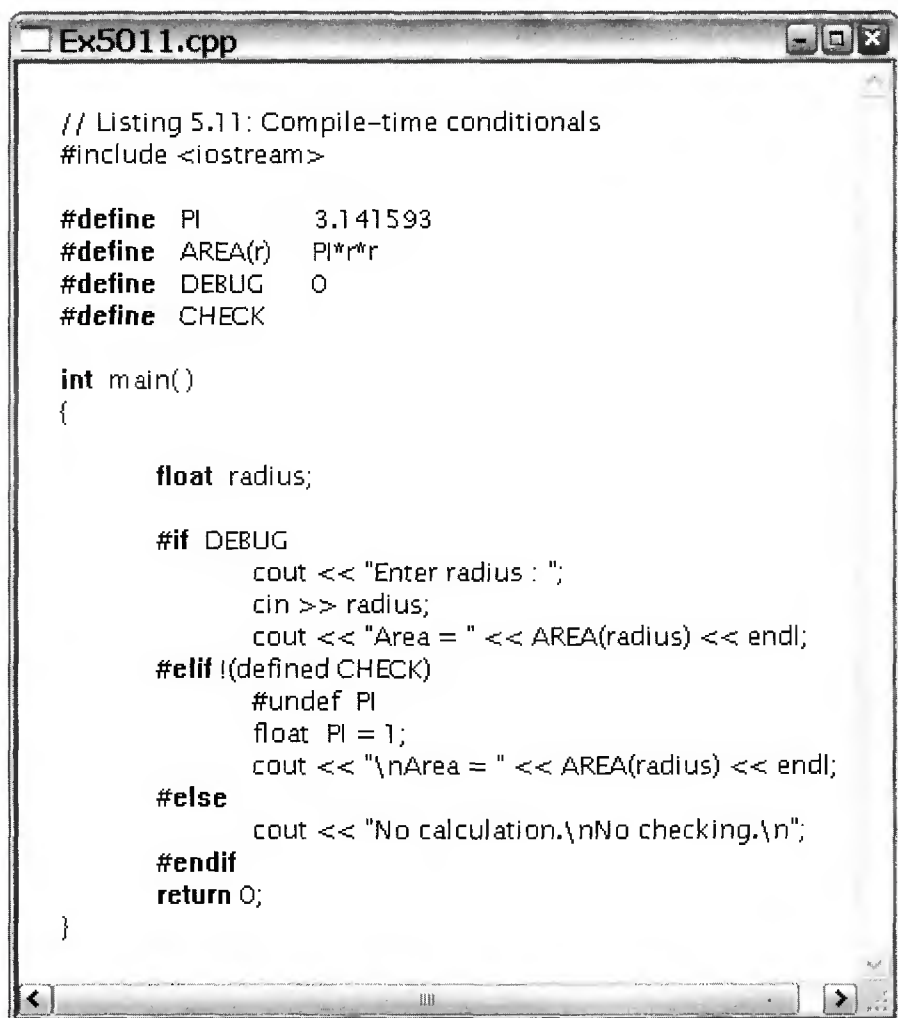
    #if defined CHECK
        #undef PI
        float PI = 1;
        cout << "\nArea = " << AREA(radius) << endl;
    #endif
    return 0;
}
```

ပုံ (၅. ၁၉)

တစ်ကယ်လို့ Ex5010.cpp program ကို ပုံ (၅. ၂၀) မှာပြထားတဲ့အတိုင်းပြင်ရေးပြီး run မယ်ဆိုရင် **#if DEBUG** ဟာ **false** ဖြစ်တဲ့အတွက် အဲဒီ **#if** block ထဲကိုမဝင်တော့ပဲ အောက်ဆင်းလာမှာပါ။ **#elif (defined CHECK)** ကလည်း **false** ဖြစ်ပါတယ်။ ဒီတော့ အောက်ကိုဆက်ဆင်းလာပြီး **#else** ကအလုပ်ကို လုပ်မှာပါ။ ဒါဆိုရင် ကွန်ပူတာမှာ **No calculation. No checking.** ဆိုတဲ့စာကြောင်း (2) ကြောင်း ကိုပေါ်ပေး ခါလိမ့်မယ်။

ပုံ (၅. ၂၁) မှာ Ex5011.cpp program ကို run ပြထားပါတယ်။





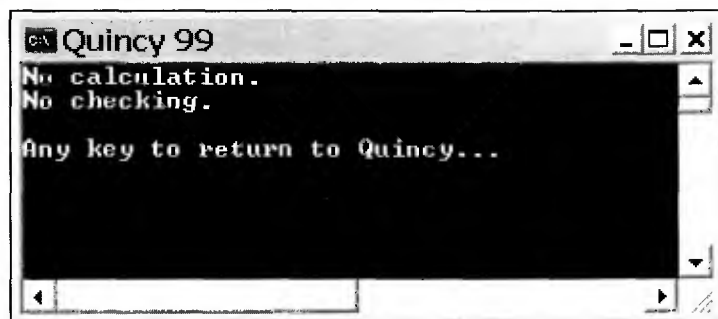
```
// Listing 5.11: Compile-time conditionals
#include <iostream>

#define PI      3.141593
#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;
}
```

٥ (٩٠ ج)



```
Quincy 99
No calculation.
No checking.
Any key to return to Quincy...
```

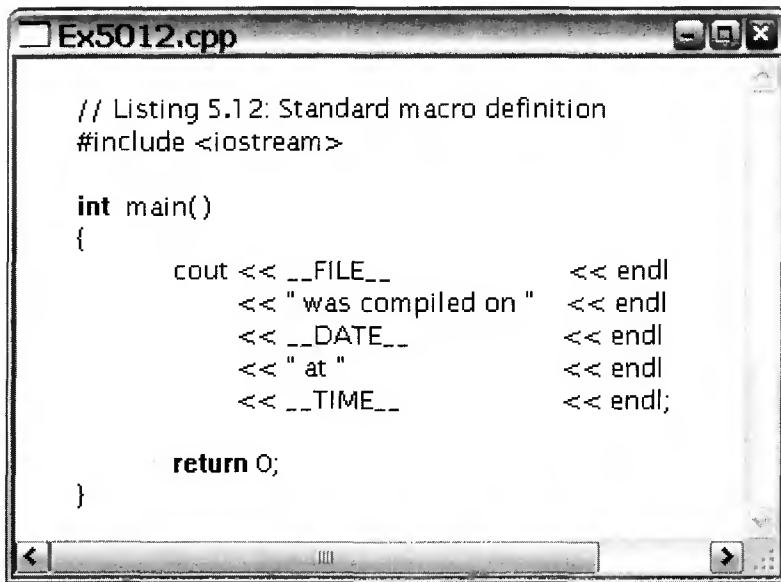
٥ (٩٠ ج)

# Standard Defined Macro Names

compiler system ထဲမှာ define လုပ်ပြီးသားရှိနေတဲ့ standard macro တွေကို ဇယား (၅. ၂) မှာ ပြထားပါတယ်။ အဲဒီထဲကတစ်ချို့ကို ပုံ (၅. ၂၂) က Ex5012.cpp program မှာ use လုပ်ပြထားပါတယ်။

ဇယား (၅. ၂) 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.



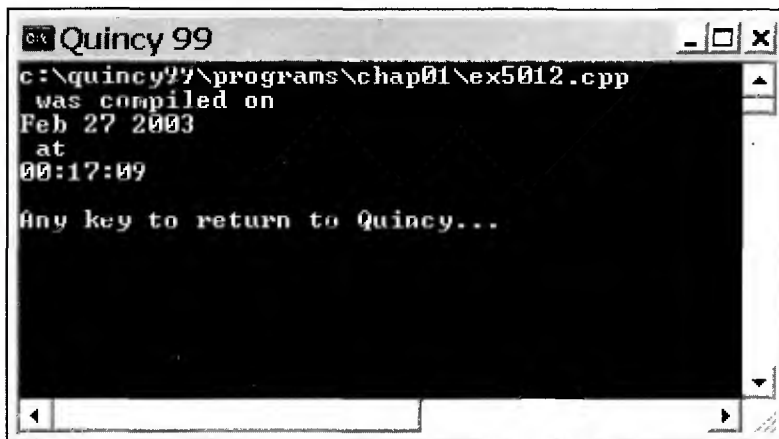
```
// Listing 5.12: Standard macro definition
#include <iostream>

int main()
{
    cout << __FILE__          << endl
          << " was compiled on " << endl
          << __DATE__          << endl
          << " at "             << endl
          << __TIME__          << endl;

    return 0;
}
```

ပုံ (၅. ၂၂)

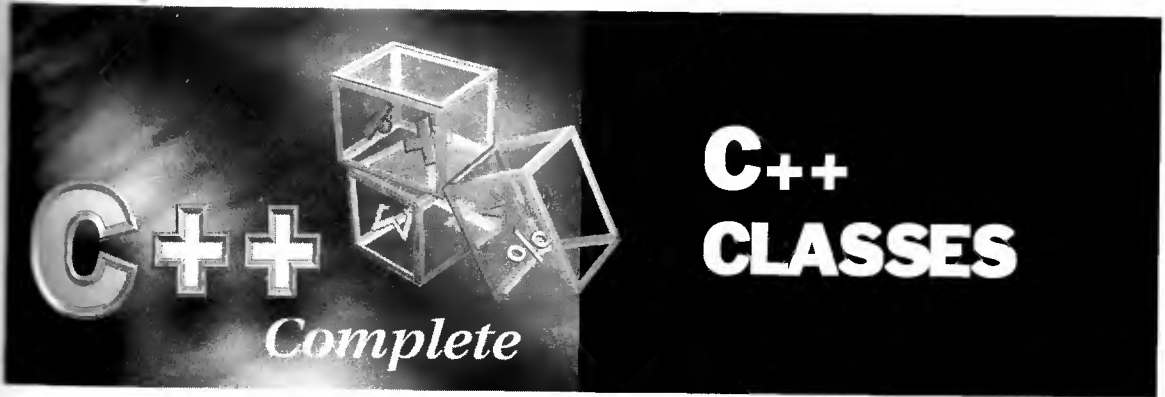
၂။ Ex5012.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၅. ၂၃) မှာပြထားတဲ့အတိုင်း ကွန်ပိုက်တာမှာပေါ်လာမှာပါ။ စာဖတ်သူကိုယ်တိုင်လက်တွေ့စမ်းကြည့်ပါ။



```
c:\quincy99\programs\chap01\ex5012.cpp
was compiled on
Feb 27 2003
at
00:17:09
Any key to return to Quincy...
```

ပုံ (၅. ၂၃)

# Chapter 6



class ဆိုတာ C++ program တွေရဲ့လုပ်ဆောင်မှုစွမ်းရည်ကို တိုးမြှင့်ပေးတဲ့အမျိုးအစားတစ်ခု ဖြစ်ပါတယ်။ C++ မှာ အမျိုးအစားမတူတဲ့ variable တွေ၊ function တွေကိုစုပေါင်းပြီး class တစ်ခုကိုကြေငြာလို့ ရပါတယ်။ class ကိုကြေငြာပြီးရင်အဲဒီ class အမျိုးအစားဖြစ်တဲ့ object အသစ် (instance) တွေကို create လုပ်ချင်ရင် လုပ်လို့ရပြီလေ။ class တစ်ခုကြေငြာမယ်ဆိုရင် class ဆိုတဲ့ keyword ကိုအသုံးပြုရမှာပါ။ ကောင်းပြီ၊ class declaration တစ်ခုရဲ့ပုံစံကိုကြည့်ပါ။

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

ဒီပုံစံမှာဆိုရင် private က default ဖြစ်တဲ့အတွက် class declaration မှာထည့်မရေးလည်းရပါတယ်။ class မှာကြေငြာထားတဲ့ function နဲ့ variable တွေကို member တွေလို့အလွယ်ခေါ်ရင်ပါတယ်။ private

အနေနဲ့ကြေငြာထားတဲ့ member တွေကို class member function တွေကပဲ access လုပ်ပေးနိုင်ပါတယ်။ class ထဲမှာမပါတဲ့ တစ်ခြား function တွေကနေ call ခေါ်လို့မရပါဘူး။ ဒါပေမယ့် public member တွေကိုတော့ program မှာပါဝင်တဲ့တစ်ခြား function တွေက call ခေါ်ပြီးအသုံးပြုလို့ရပါတယ်။ ဒီတော့ private member တွေကို ကျွန်တော်တို့အသုံးပြုချင်ရင် public member တွေကနေလှည့်ပြီး call ခေါ်ရင်ရတာပေါ့ မဟုတ်ဘူးလား။ public member တွေကိုကြေငြာတဲ့အခါမှာ public ဆိုတဲ့ keyword ကို အသုံးပြုရပါတယ်။ public keyword နောက်မှာ colon (:) တစ်ခုထည့်ပေးဖို့မမေ့ပါနဲ့။ class တစ်ခုကို create မလုပ်ခင် အောက်ဖော်ပြပါ program ကိုလေ့လာကြည့်စေချင်ပါတယ်။

```
// 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;
}
```

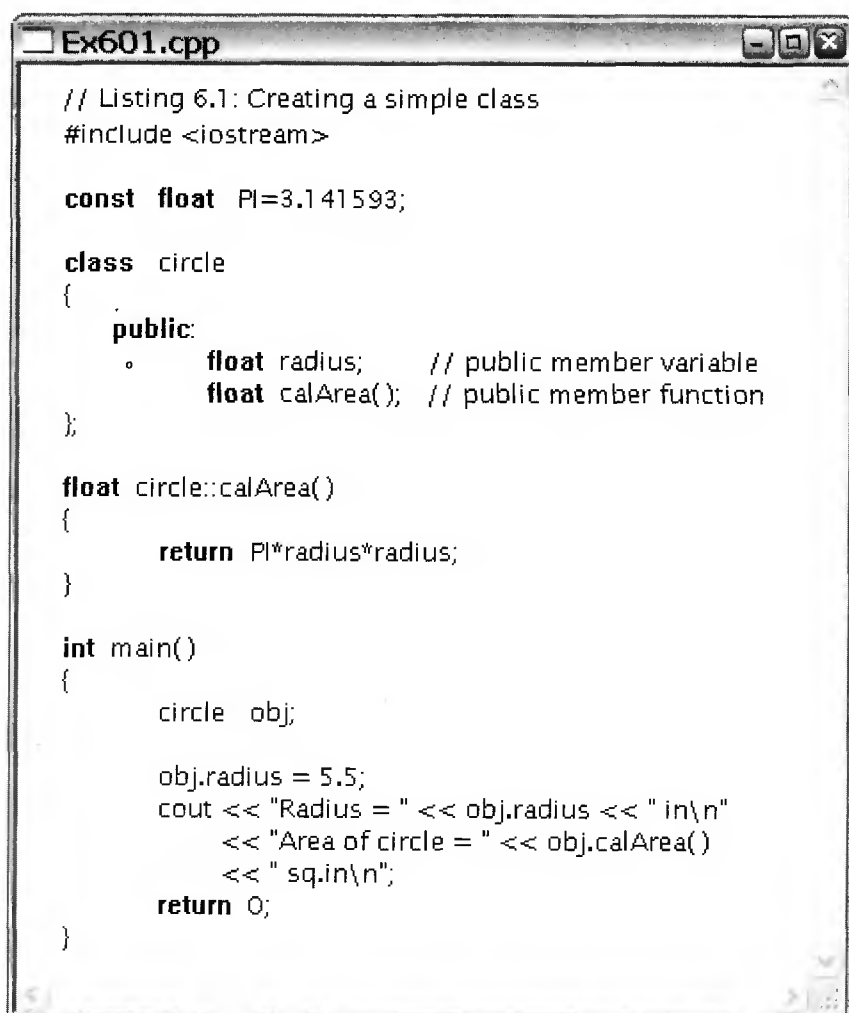
```
float calArea( )
{
    return PI*radius*radius;
}
```

အထက်ပါ program မှာဆိုရင် radius ကို global variable အနေနဲ့ကြေငြာထားတဲ့အတွက် ဒီ radius ဟာ program ထဲက function တိုင်းအတွက် public ဖြစ်နေပါတယ်။ ထိုနည်းတူ calArea( ) function ဟာလည်း public ဖြစ်နေတယ်နော်။ ဒီ program မှာ private ဖြစ်တဲ့ variable တို့၊ function တို့လုံးဝမပါပါဘူး။ ဒီ program ကို class ပုံစံနဲ့ရေးမယ်ဆိုရင်ရေးလို့ရပါတယ်။ ပုံ (၆. ၁) မှာဖော်ပြထားတဲ့ Ex601.cpp program ကိုလေ့လာကြည့်ပါ။

## ၆.၁ Create A Simple Class

Ex601.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- class type circle ကိုကြေငြာတဲ့အထဲမှာ private member ကိုထည့်မထားပါဘူး။ public member variable radius နဲ့ member function calArea( ) တို့ကိုပဲထည့်ထားပါတယ်။ ဒါဆိုရင် radius ကို class member function နဲ့ main( ) function တို့ကပါ အသုံးပြုလို့ရပါပြီ။ တစ်ခုတော့ရှိတယ်။ main( ) ကနေ လှမ်းအသုံးပြုနိုင်ဖို့အတွက် သူ့ထဲမှာ circle



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

const float PI=3.141593;

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

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

int main()
{
    circle obj;

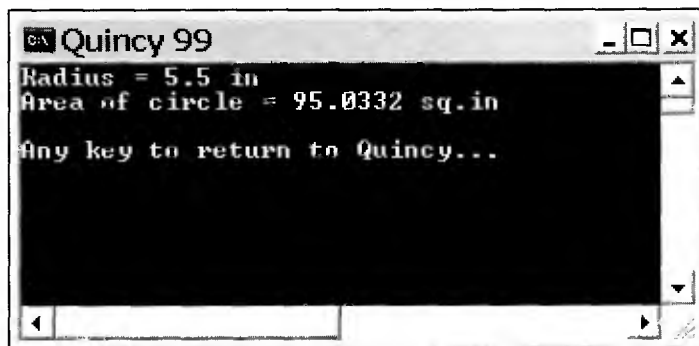
    obj.radius = 5.5;
    cout << "Radius = " << obj.radius << " in\n"
          << "Area of circle = " << obj.calArea()
          << " sq.in\n";
    return 0;
}
```

ပုံ (၆. ၁)



class instance တစ်ခုကို create လုပ်ရပါမယ်။ ကောင်းပြီ ၊ main( ) function ထဲမှာ circle class instance တစ်ခုဖြစ်တဲ့ obj ကို create လုပ်ရအောင်။ `obj.radius = 5.5` လို့ ရေးလိုက်ခြင်းအားဖြင့် class declaration ထဲက `radius = 5.5` ဖြစ်သွားပါတယ်။

- `obj.calArea( )` ဆိုတာ class declaration ထဲက member function ကို call ခေါ်လိုက်တာပါပဲ။ ဒီတော့ `float circle::calArea( )` ဆိုတဲ့ class member function ထဲကိုဝင်လာပါပြီ။ member function header ကိုကြည့်မယ်ဆိုရင် ရှေ့ဆုံးမှာ function return type ရှိပါတယ်။ နောက်ပြီး class name ၊ scope resolution operator ( `::` ) နဲ့ နောက်ဆုံးမှာ function name နဲ့ parameter list ပါ။ parameter list ထဲမှာ argument မရှိရင် ကွင်းအလွတ်ပဲထားပေါ့။ `calArea( )` function ထဲမှာ  $PI * radius * radius = 3.141593 * 5.5 * 5.5 = 95.0332$  ကိုတွက်ယူပြီး `main( )` ကို return ပြန်ပို့ပေးပါတယ်။
- `main( )` function ကိုပြန်ရောက်တဲ့အခါကျရင် `obj.calArea( ) = 95.0332` ဖြစ်နေပါလိမ့်မယ်။ အခုနေ Ex601.cpp program ကို run လိုက်မယ်ဆိုလို့ရှိရင် ပုံ (၆. ၂) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၆. ၂)

## Radius is Made Private

တစ်ကယ်လို့ Ex601.cpp program က radius member ကို public အစား private လုပ်မယ်ဆိုရင် program ကို ပုံ (၆. ၃) မှာပြထားတဲ့အတိုင်း ပြင်ရေးရပါမယ်။ Ex602.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

```
Ex602.cpp

// Listing 6.2: Using the private member variable
#include <iostream>
const float PI=3.141593;

class circle
{
    float radius;    // private member variable
public:
    void setRadius(float); // public member functions
    float calArea();
};

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

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

int main()
{
    circle obj;

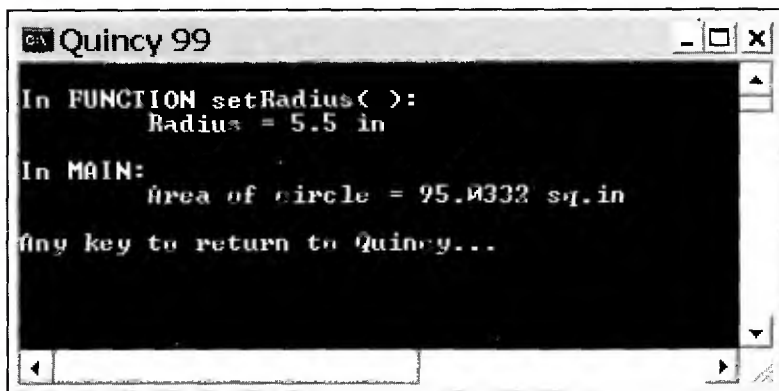
    obj.setRadius(5.5);    // call public function
    cout << "In MAIN:\n"
         << "\tArea of circle = " << obj.calArea()
         << " sq.in\n";
    return 0;
}
```

ပုံ (၆. ၃)

- စာချင်းမှာ `const float PI` ကို `3.141593` လို့ `global value` အနေနဲ့ `initialize` လုပ်ပေးပါတယ်။ နောက်ပြီး `circle` class instance တစ်ခုဖြစ်တဲ့ `obj` ကို `create` လုပ်ပါတယ်။ `obj.setRadius(5.5);` ကနေ `member function` ကို `call` ခေါ်ပါတယ်။ `argument` က `5.5`

ပါ။ pass လုပ်ပေးလိုက်တဲ့ 5.5 ဟာ public member function မှာ  $r = 5.5$  အနေနဲ့ဝင်သွားပါပြီ။ radius ကို class declaration မှာ private လို့ကြေငြာထားတာမို့ main( ) function ကနေ radius အတွက် data ကိုတိုက်ရိုက် pass လုပ်လို့မရပါဘူး။ setRadius(float r ) function ထဲမှာ radius = r; ဆိုတဲ့ assignment statement တစ်ခုကိုထပ်ရေးပေးရပါမယ်။ class ထဲမှာတော့ public member က private member ကို access လုပ်လို့ရတာကိုး။ အခုဆိုရင် radius = 5.5 လို့ ကွန်ပျူတာကသိနေပါပြီ။ member function ထဲမှာ radius ကို display လုပ်ခိုင်းပါမယ်။

- တစ်ချိန်တည်းမှာပဲ main( ) ကိုပြန်ရောက်လာရင် class ထဲက public member function တစ်ခုဖြစ်တဲ့ calArea( ) ကလည်း radius တန်ဖိုးကိုသိသွားပါပြီ။ ဒီတော့ main( ) ထဲကနေ calArea( ) function call ခေါ်တဲ့အခါမှာ  $PI * radius * radius$  တန်ဖိုးကို တွက်ပေးနိုင်ပါတယ်။ နောက်ပြီးအဖြေကို main( ) ဆီပြန်ပို့ပေးတဲ့အခါမှာ obj.calArea( ) ကလက်ခံပါတယ်။ အခုနေ obj.calArea( ) တန်ဖိုးကို display လုပ်ခိုင်းရင် circle area ကိုသိရမှာပါပဲ။
- Ex602.cpp program ကို run လိုက်ရင် ပုံ (၆. ၄) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ main( ) ကနေ pass လုပ်ပေးလိုက်တဲ့ radius အတွက် data ကို setRadius( ) function ကနေ display လုပ်ခိုင်းပြီး area တန်ဖိုးကို main( ) ထဲမှာကျမှ display လုပ်ခိုင်းတာပါပဲ။
- Ex602.cpp program ကို စာဖတ်သူအနေနဲ့ အလွယ်တကူ trace လုပ်လို့ရအောင်လို့ flow diagram တစ်ခုကိုဆွဲပြပေးထားပါတယ်။ ပုံ (၆. ၅) ကိုကြည့်ပါ။ class အကြောင်းကို သေချာနားလည်လာရင် နောက်ပိုင်းမှာ flow diagram လိုမှာမဟုတ်ပါဘူး။



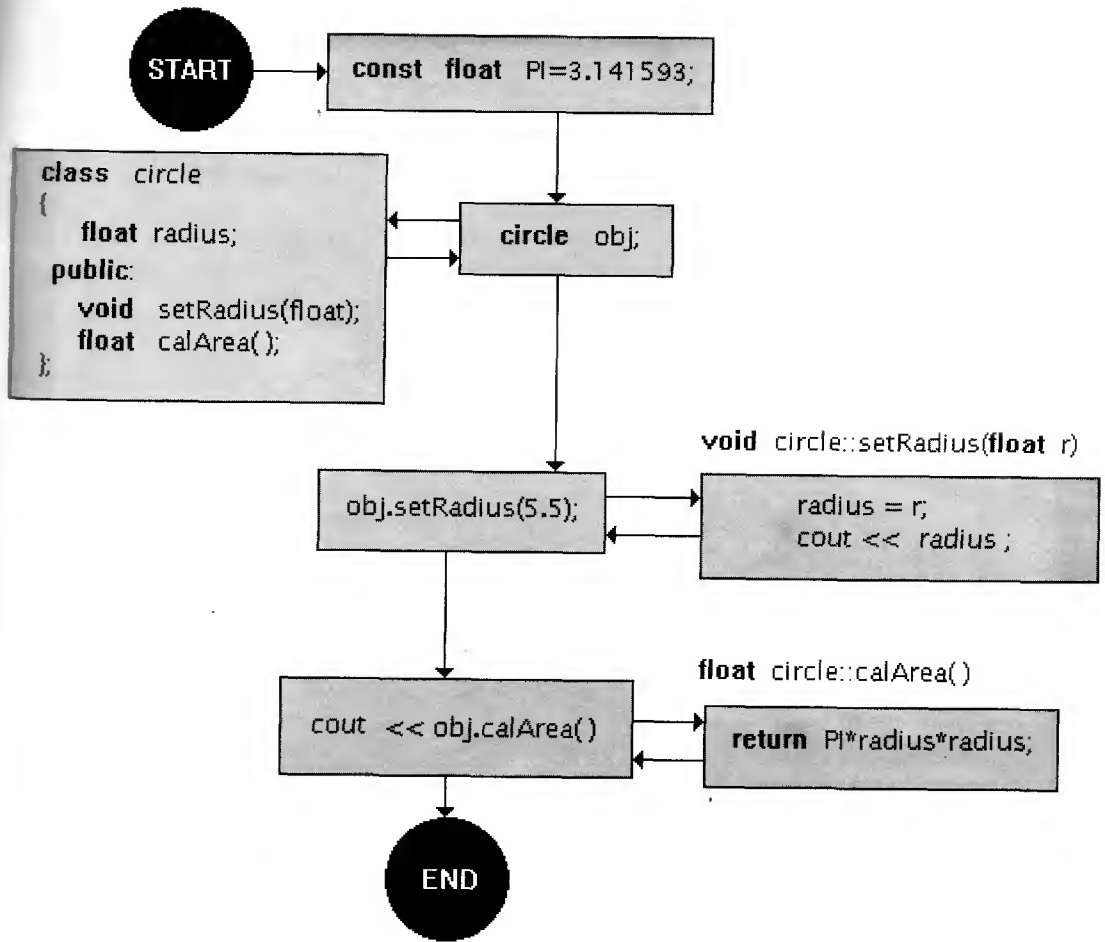
```

Quincy 99
In FUNCTION setRadius( ):
    Radius = 5.5 in

In MAIN:
    Area of circle = 95.4332 sq.in

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

ပုံ (၆. ၄)



ပုံ (၆. ၅)

## ၆.၂ Using the Inline Function

တစ်ကယ်လို့ `setRadius( )` နဲ့ `calArea( )` function တွေကို `class declaration` မှာပဲ တွဲရေးမယ်ဆိုရင် လည်း `program` ကမမှားပါဘူး။ ဒီဥစ္စာကို `Inline function declaration` လို့ခေါ်ပါတယ်။ ပုံ (၆. ၆) မှာရေးထားတဲ့ `Ex603.cpp` `program` ဟာ `inline function` နည်းနဲ့ `Ex602.cpp` `program` ကိုပြင်ရေးထားတာဖြစ်ပါတယ်။

```
Ex603.cpp

// Listing 6.3: Using the inline function
#include <iostream>
const float PI=3.141593;

class circle
{
    float radius;
public:
    void setRadius(float r)
    {
        radius = r;
        cout << "\nIn 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. ၆)

- Ex603.cpp program ကို run မယ်ဆိုရင် Ex602.cpp program ကို run သလိုပဲ အဖြေ အတူတူရမှာပါ။

## More on the Inline Function

inline function အကြောင်းကို ပိုမိုပြည့်စုံစွာနားလည်အောင် နောက် program တစ်ခုကိုတင်ပြပါဦးမယ်။

ပုံ (၆. ၇) မှာဖော်ပြထားတဲ့ Ex604.cpp program ဟာ cube တစ်ခုရဲ့ volume ကိုတွက်ပေးပါလိမ့်မယ်။

```
Ex604.cpp

// Listing 6.4: 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 << " ft\n"
              << "Width  = " << width  << " ft\n"
              << "Depth  = " << depth  << " ft\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 (၆.၇)

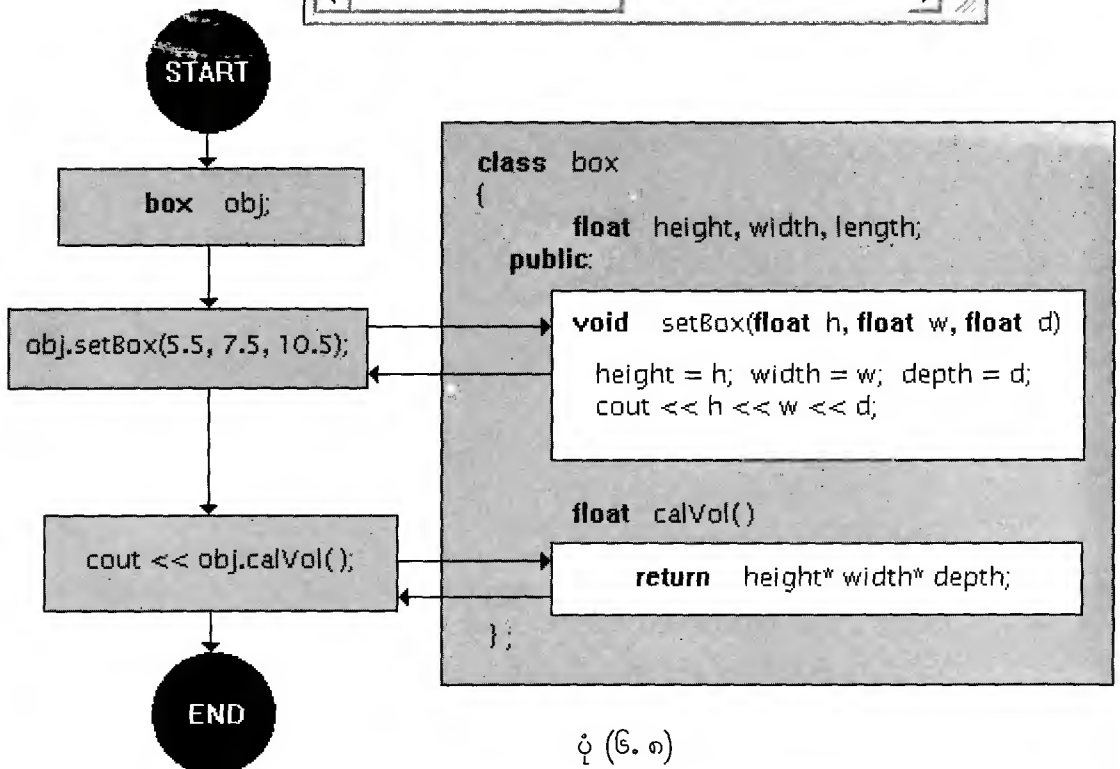
Ex604.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

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



7.5 နဲ့  $depth = d = 10.5$  အနေနဲ့ assign လုပ်ပေးပါလိမ့်မယ်။ private member တွေကို public member function တွေက သိသွားပြီဆိုရင် obj.calVol( ) ကနေ call ခေါ်ပြီး calVol( ) function ထဲမှာ volume ကို  $5.5 * 7.5 * 10.5 = 433.125$  လို့တွက်ပါလိမ့်မယ်။ နောက်ပြီးအဖြေကို main( ) ဆီ return လုပ်ပေးမှာပါ။

- main( ) ကိုပြန်ရောက်တဲ့အခါမှာ obj.calVol( ) ကို print လုပ်ခြင်းအားဖြင့် 433.125 ဆိုတဲ့ အဖြေကိုတွေ့ရပါလိမ့်မယ်။ အခုနေ Ex604.cpp program ကို run လိုက်မယ်ဆိုလို့ရှိရင် ပုံ (၆. ၈) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ flow diagram ကိုလည်းအောက်မှာဖော်ပြထားပါတယ်။



ပုံ (၆. ၈)

# Constructor and Destructor

C++ program တစ်ခုက class declaration မှာ action statement တစ်ခုကိုကြေငြာပေးချင်ရင် constructor function ကိုအသုံးပြုရပါမယ်။ inline function ဆန်ဆန်ပါပဲ။ မတူတာက constructor ရဲ့နာမည်

```
Ex605.cpp

// Listing 6.5: Using constructor and destructor
#include <iostream>
const float PI=3.141593;

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

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

circle::~circle() { }

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

int main()
{
    circle obj(5.5);

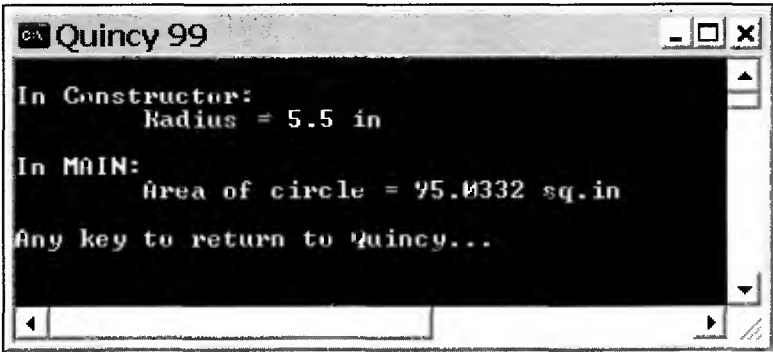
    cout << "In MAIN:\n" << "\tArea of circle = "
         << obj.calArea() << " sq.in\n";
    return 0;
}
```

ပုံ (၆.၉)

ဟာသူပါဝင်နေတဲ့ class name ဝဲဖြစ်နေပါတယ်။ သူ့မှာ return type ထည့်ပေးဖို့မလိုပါဘူး။ constructor function အသုံးပြုနည်းကို လေ့လာကြည့်ချင်တယ်ဆိုရင် ပုံ (၆. ၉) မှာရေးထားတဲ့ Ex605 program ကိုကြည့်ပါ။ ဒီဥပမာ Ex601 program ကို နောက်တစ်မျိုးပြင်ရေးထားတာဖြစ်ပါတယ်။

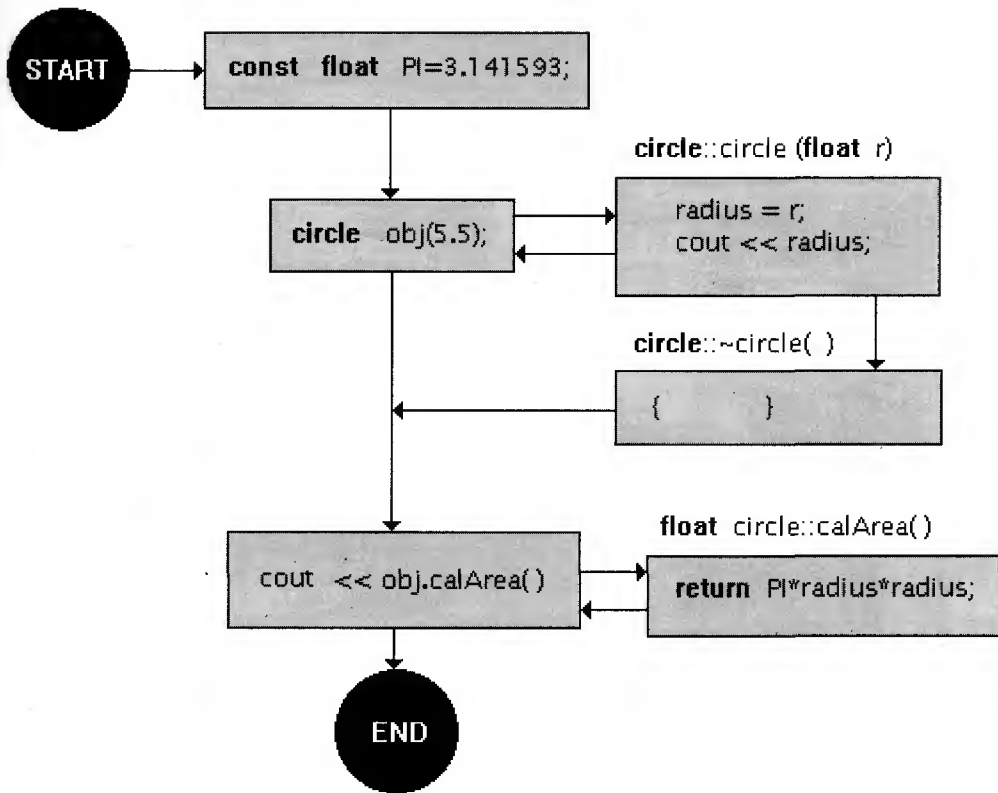
၂။ Ex605.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- program စတင်ချင်း main( ) function ထဲမှာ obj ဆိုတဲ့ class instance တစ်ခုကို create လုပ်လိုက်တာနဲ့ constructor ဟာအရင်ဆုံး call ခေါ်ပြီးသားဖြစ်သွားပါပြီ။ constructor ထဲမှာ obj ကနေ pass ဝင်လာတဲ့ 5.5 ကို r ကနေတစ်ဆင့် radius ထဲကိုပြောင်းပေးပါလိမ့်မယ်။ ဒါကြောင့်မို့ constructor ထဲမှာ radius value ကို display လုပ်ပြနိုင်တာပါပဲ။
- ပြီးရင် main( ) ကိုပြန်လာပြီး obj.calArea( ) ကနေ calArea( ) function ကို call ခေါ်ပါလိမ့်ဦးမယ်။ ဒါဆိုရင် public member function calArea( ) ထဲရောက်သွားပြီး  $PI * radius * radius = 3.141593 * 5.5 * 5.5 = 95.0332$  ကိုတွက်ယူပြီးအဖြေကို main( ) ဆီ return လုပ်ပေးမှာပါ။ main( ) ထဲမှာ obj.calArea( ) ကို print လုပ်ရင်အဖြေပေါ်လာပါလိမ့်မယ်။
- ~circle( ) {     }; statement ဟာ destructor function ကိုရေးထားတာပါပဲ။ ဒီဥပမာက program မှာမထည့်လည်းရပါတယ်။ တစ်ကယ်လို့ constructor ကအလုပ်ပြီးသွားပေမယ့် memory allocatin အတွက် အလုပ်ကျန်နေခဲ့တယ်ဆိုရင် destructor က clear လုပ်ပေးမှာဖြစ်ပါတယ်။ Ex605.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၆. ၁၀) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၆. ၁၀)

- Ex605.cpp program အတွက် flow diagram ကို ပုံ (၆. ၁၁) မှာဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



ပုံ (၆. ၁၁)

## Show Timer

၁။ Ex606.cpp program ဟာဆိုရင် timer တစ်ခုအလုပ်လုပ်ပုံကို constructor နဲ့ destructor တို့ကို အသုံးပြုပြီး program ရေးထားတာဖြစ်ပါတယ်။ ပုံ (၆. ၁၂) ကိုကြည့်ပါ။ program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- main( ) ထဲမှာ obj ဆိုတဲ့ timer object တစ်ခုကို define လုပ်လိုက်တာနဲ့ constructor function က အလုပ်လုပ်ပါပြီ။ ဒါဆိုရင် timer( ) constructor ထဲမှာရေးထားတဲ့ clock( ) standard library function ကိုအသုံးပြုပြီး start = clock( )=0 လို့ initialize လုပ်ပါတယ်။ ကွန်ပျူတာစကရင်မှာ Start time = 0 လို့အရင်ဆုံးပေါ်လာမှာပါ။ နောက်ပြီး main( ) ကိုပြန် ရောက်လာပြီး Press a key followed by ENTER: ဆိုတဲ့စာကြောင်းပေါ်လာပါလိမ့်မယ်။

```
Ex606.cpp

// Listing 6.6: Showing timer
#include <iostream>
#include <ctime>

class timer
{
    clock_t start;
public:
    timer();
    ~timer();
};

timer::timer()
{
    start = clock();
    cout << "Start time = " << start << endl;
}

timer::~timer()
{
    clock_t end;

    end = clock();
    cout << "End time = " << end << endl
         << "Elapsed time = " << (end - start) << endl;
}

int main()
{
    timer obj;
    char c;

    cout << "Press a key followed by ENTER\n";
    cin >> c;
    return 0;
}
```

ပုံ (၆. ၁၂)

ကျွန်တော်တို့က key တစ်ခုကိုနှိပ်ပြီး ENTER key ကို နှိပ်လိုက်တာနဲ့ကွန်ပျူတာက destructor function နှိတ်နေရာကိုတန်းထွက်သွားပါပြီ။ destructor မှန်းသိအောင်လို့ timer ရှေ့မှာ အခုလို သင်္ကေတ (~) ထည့်ပေးထားတာကိုသတိပြုကြည့်ပါ။

- destructor function ထဲမှာ `end = clock( )` လို့ assign လုပ်ပေးလိုက်တဲ့အတွက် (လောလောဆယ်) နောက်ဆုံးအချိန်ကို `end` ထဲမှာထည့်ပေးမှာပဲ။ `end` ကို `display` လုပ်ကြည့်ရင် အချိန်ကိုသိရပါမယ်။ `elapsed time` ဟာ `end - start` မဟုတ်ဘူးလား။ အဲဒါကို `display` လုပ်ပြီးရင် destructor ကနေ object ကို ဖျက်ပစ်လိုက်ပြီဖြစ်ပါတယ်။
- ပုံ (၆. ၁၃) ဟာဆိုရင် `Ex606.cpp` program ကို run ပြထားတာပါ။ Press a key followed by ENTER ဆိုတဲ့စာပေါ်လာတဲ့အချိန်မှာ keyboard ကနေ character တစ်ခုခုကို ရိုက်ထည့်ရင် timer အလုပ်လုပ်ပုံကိုမြင်ရပါလိမ့်မယ်။

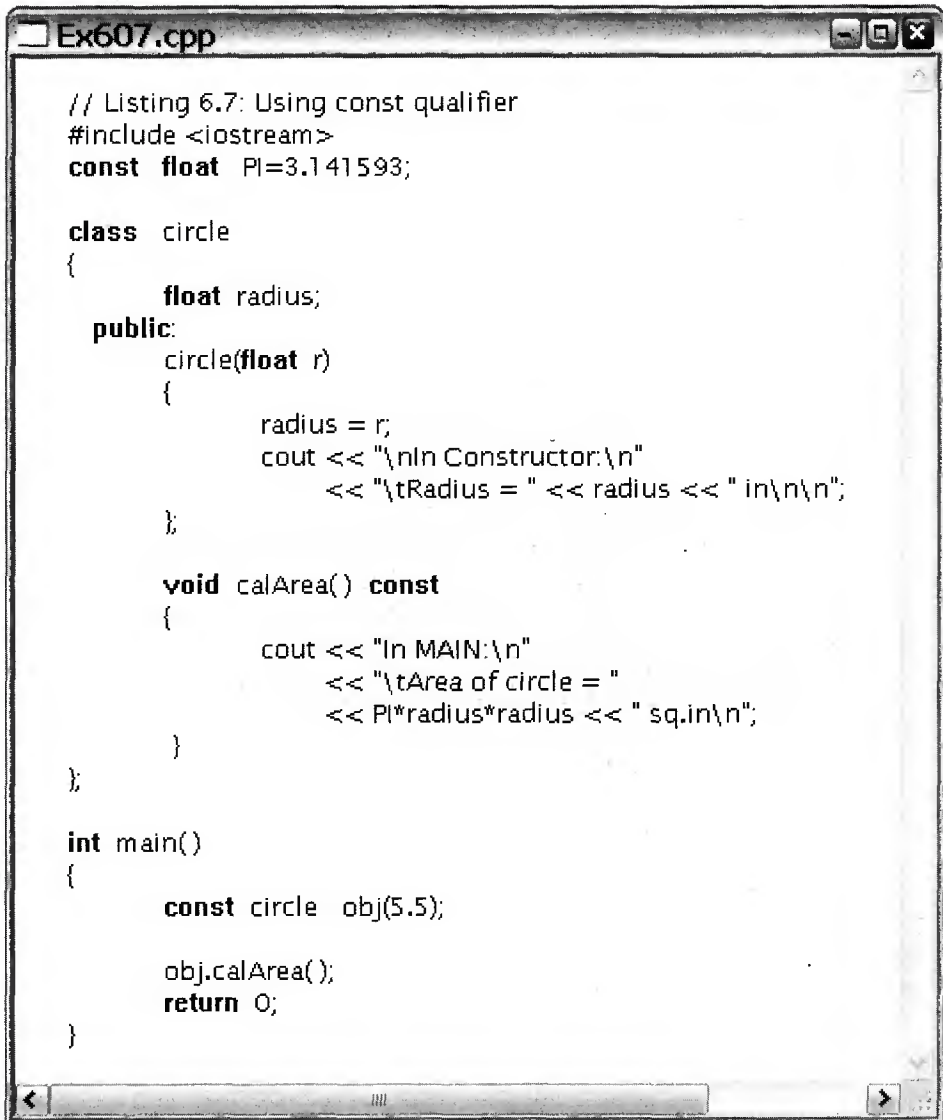
ပုံ (၆. ၁၃)

## ၆.၄ Classes and const

၁။ C++ program တစ်ခုမှာ object တစ်ခုကို `const` အနေနဲ့ declare လုပ်ပေးမယ်ဆိုရင် `const` မဟုတ်တဲ့ class member function တွေကနေ ဒီ object ကို call ခေါ်လို့မရပါဘူး။ call ခေါ်လို့ရချင်ရင် လုပ်ပေးရမှာက member function declaration မှာ `const` qualifier ကိုထည့်ပေးရပါမယ်။ ပုံ (၆. ၁၄) မှာရေးထားတဲ့ `Ex607.cpp` program ဟာဆိုရင် `const` qualifier အသုံးပြုနည်းကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

- `Ex607.cpp` program ကို run လိုက်မယ်ဆိုရင် `Ex605.cpp` program ကို run တဲ့အဖြစ်ပဲရမှာဖြစ်ပါတယ်။ ဘာပြုလို့လဲဆိုတော့ `Ex607.cpp` ဟာ `const` qualifier ကိုအသုံးပြုပြီး `Ex605` program ကိုပြင်ရေးထားလို့ပါပဲ။





```
// Listing 6.7: Using const qualifier
#include <iostream>
const float PI=3.141593;

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

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

int main()
{
    const circle obj(5.5);

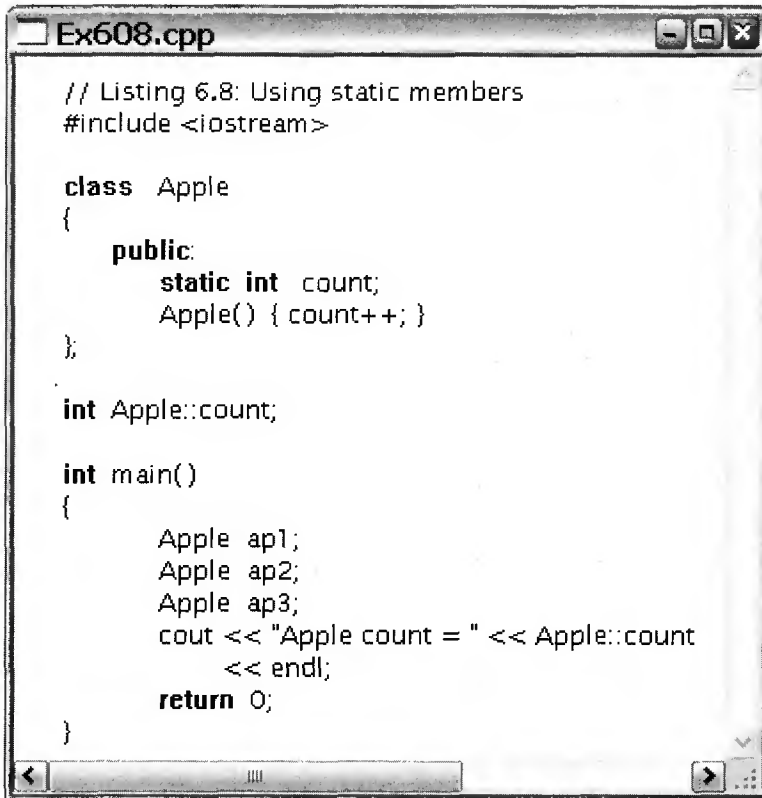
    obj.calArea();
    return 0;
}
```

ပုံ (၆. ၁၄)

## ၆.၅ static Members

၁။ C++ program မှာ class member တစ်ခုကို static လို့ကြေငြာမယ်ဆိုရင် အဲဒီ data member

class instance အကုန်လုံးနဲ့ဆိုင်တဲ့ global value တစ်ခုကိုပေးမှာပါ။ static ကို class definition ထဲမှာ ကြေငြာရသလို အပြင်မှာလည်းထပ်ကြေငြာရပါတယ်။ ပုံ (၆. ၁၅) မှာဖော်ပြထားတဲ့ Ex608.cpp program မှာ static member တွေအသုံးပြုနည်းကိုရေးထားပါတယ်။



```
// Listing 6.8: Using static members
#include <iostream>

class Apple
{
public:
    static int count;
    Apple() { count++; }
};

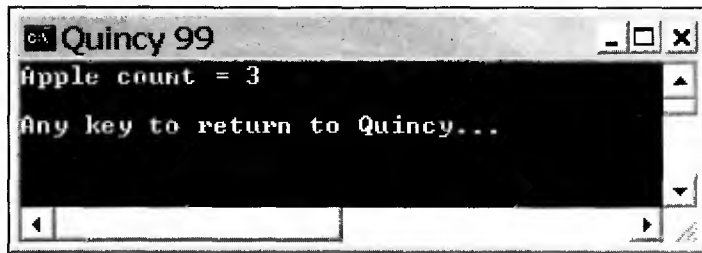
int Apple::count;

int main()
{
    Apple ap1;
    Apple ap2;
    Apple ap3;
    cout << "Apple count = " << Apple::count
          << endl;
    return 0;
}
```

ပုံ (၆. ၁၅)

၂။ Ex608.cpp program ကို လေ့လာကြည့်မယ်ဆိုရင်

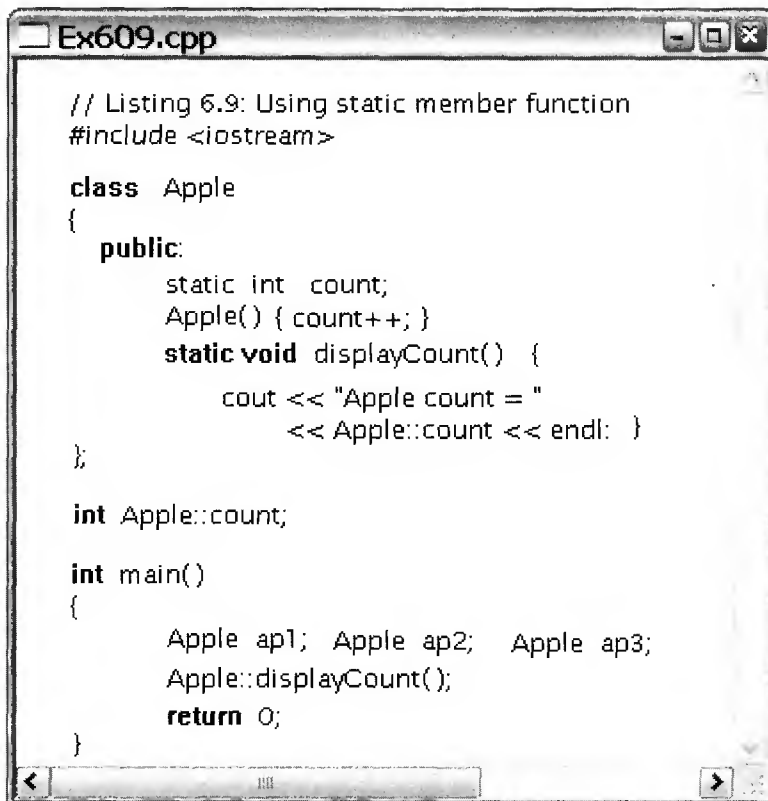
- class Apple definition ထဲမှာ static int count; လို့ရေးထားတဲ့ statement ကနေ static data member ကို declare လုပ်ပေးပါတယ်။ program စတင်ချင်း main( ) ထဲမှာ Apple class instance (3) ခုကို define လုပ်တဲ့အခါတိုင်း constructor ထဲမှာ count ကို တစ်စီတိုင်းပေးသွားပါတယ်။ ဒီတော့ Apple::count ကို print လုပ်ရင် (3) ပေါ်လာမှာပါပဲ။ ပုံ (၆. ၁၆) မှာ Ex608.cpp program ကို run ပြထားပါတယ်။



ပုံ (၆. ၁၆)

## static Member Functions

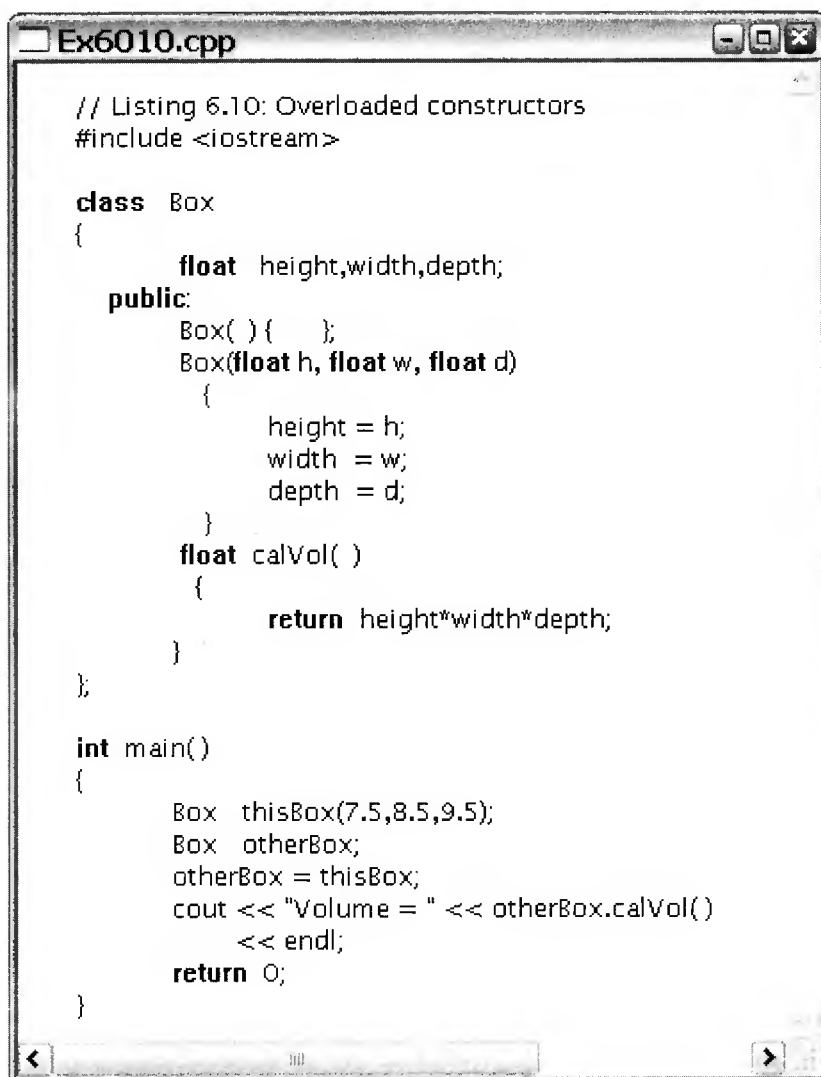
Ex608.cpp program ကို static member function အသုံးပြုပြီးပြင်ရေးမယ်ဆိုရင် ပုံ (၆. ၁၇) က Ex609.cpp ကိုရမှာပါ။ class definition ထဲက member function header ရဲ့ရှေ့ဆုံးမှာ static ထည့်ရေးထားတာကိုသတိပြုကြည့်ပါ။ static member function ကို call ခေါ်တဲ့အခါမှာ class name နဲ့တွဲရေးရပါမယ်။



ပုံ (၆. ၁၇)

## 6.6 Overloaded Constructors

၁။ ရှေ့ပိုင်းမှာတုန်းက overloaded function တွေအကြောင်းကို စာရေးသူတင်ပြခဲ့ပြီးပါပြီ။ တစ်ကယ်တော့ constructor ဟာ class member function တစ်ခုပဲဆိုတော့ သူ့ကိုလည်း function လိုပဲ overload လုပ်လို့ရပါတယ်။ ပုံ (၆. ၁၈) မှာဖော်ပြထားတဲ့ Ex6010.cpp program မှာ Box constructor ကို overload လုပ်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



```
// Listing 6.10: Overloaded constructors
#include <iostream>

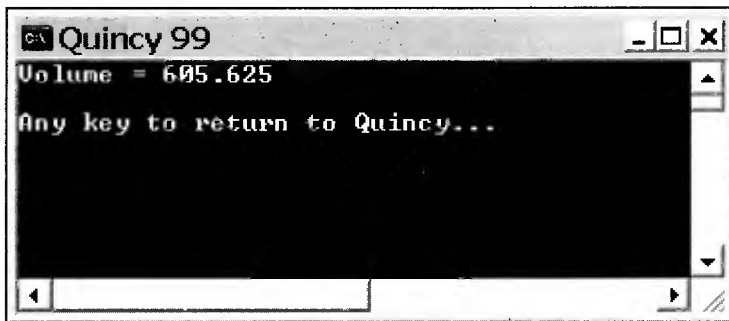
class Box
{
    float height,width,depth;
public:
    Box( ) {    };
    Box(float h, float w, float d)
    {
        height = h;
        width = w;
        depth = d;
    }
    float calVol( )
    {
        return height*width*depth;
    }
};

int main()
{
    Box thisBox(7.5,8.5,9.5);
    Box otherBox;
    otherBox = thisBox;
    cout << "Volume = " << otherBox.calVol()
        << endl;
    return 0;
}
```

ပုံ (၆. ၁၈)

## ၂။ Ex6010.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်း main( ) ထဲမှာ parameter ပါတဲ့ Box constructor ကို define လုပ်ပြီး class definition ထဲက Box constructor ကို call ခေါ်ပြီးသားဖြစ်နေပါပြီ။ constructor တွေထဲမှာ parameter ရှိတဲ့ constructor ထဲကိုရွေးဝင်လာပြီး height = h = 7.5 ၊ width = w = 8.5 နဲ့ depth = d = 9.5 လို့ set လုပ်ပေးပါတယ်။ ပြီးတော့ရင် main( ) ကိုပြန်လာမှာပါ။
- ဒီတစ်ခါ main( ) ထဲမှာ otherBox ဆိုတဲ့ Box instance အသစ်တစ်ခုကို define လုပ်ပါတယ်။ ဒီတော့ class definition ထဲက Box( ) {     }; constructor ထဲဝင်လာပြီး ဘာမှမလုပ်ပဲ main( ) ကိုပြန်သွားပါတယ်။
- main( ) ထဲမှာ thisBox ရဲ့ value ကို other Box နဲ့ assign လုပ်ပေးလိုက်တဲ့အတွက် otherBox.calVol( ) ကို display လုပ်ခိုင်းတဲ့အခါကျရင် float calVol( ) function ထဲဝင်သွားပြီး height\*width\*depth = 7.5\*8.5\*9.5 = 605.25 ကိုတွက်ပြီးတော့ return လုပ်ပေးပါလိမ့်မယ်။ ဒါကြောင့်မို့လည်း Ex6010.cpp program ကို run လိုက်တဲ့အခါမှာ ဒီအဖြေက ကွန်ပျူတာမှာပေါ်လာတာပါပဲ။ ပုံ (၆. ၁၉) ကိုကြည့်ပါ။



ပုံ (၆. ၁၉)

## More Overloaded Constructors

၁။ C++ program တွေမှာ overloaded constructor တွေကို ပိုပြီးအသေးစိတ်အသုံးပြုချင်တယ်ဆိုလို့ရှိရင် Ex6011.cpp program ကိုလေ့လာကြည့်ပါ။ ဒီ program မှာ clas တစ်ခုကို နောက်တစ်ခုထဲမှာ 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( );

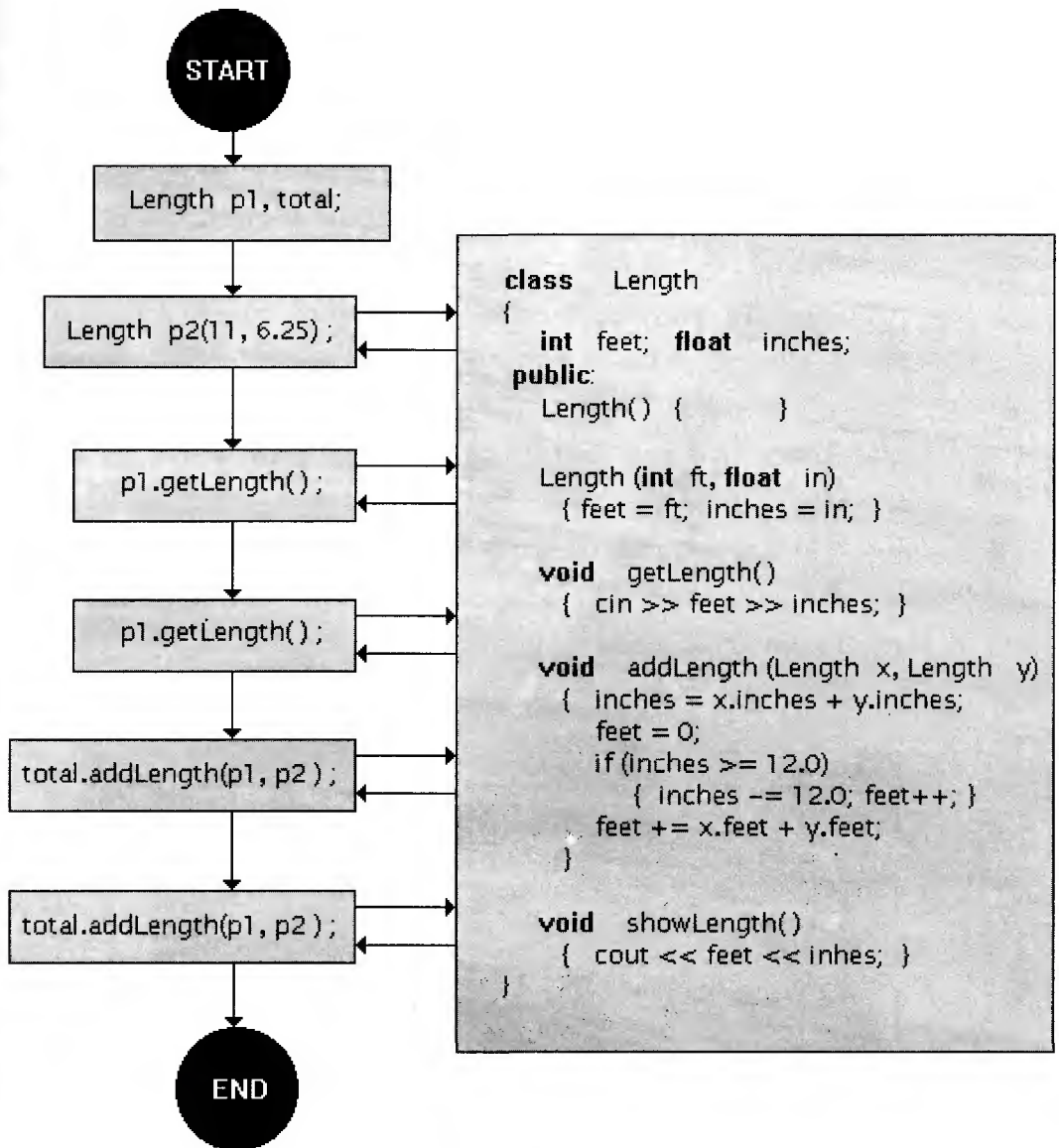
    cout << "\nTotal Length = ";
    total.showLength( );

    return 0;
}

```

၂။ Ex6011.cpp program အတွက် flow diagram ကို ပုံ (၆. ၂၀) မှာရေးပြထားပါတယ်။ Ex6011.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်းမှာ p1 နဲ့ total ဆိုတဲ့ class Length object (2) ခုကို create လုပ်ပါတယ်။ အဲဒါကြောင့် constructor Length( ) function ထဲဝင်သွားပြီး feet နဲ့ inches တို့ကို private member တွေအနေနဲ့ define လုပ်ထားလိုက်ပါပြီ။ တစ်ခြားဘာမှတော့မလုပ်ပါဘူး။ main( ) ကိုဒီတိုင်းပြန်လာပါတယ်။



ပုံ (၆.၂၀)

- ဒီတစ်ခါ object p2 (11,6.25) ကို define လုပ်ပါတယ်။ overloaded constructor `Length(int ft, float in)` ထဲရောက်လာပြီး `feet = ft = 11` နဲ့ `inches = in = 6.25` လို့ assign လုပ်ပေးပြီး `main( )` ကိုပြန်လာလိမ့်မယ်။
- `p1` object အတွက် `get_length( )` function ကို call ခေါ်ပါတယ်။ အဲဒီ function ထဲမှာ `feet` နဲ့ `inches` အတွက် data တွေကိုထည့်ပေးရပါမယ်။ `feet = 15` ကို ရိုက်ထည့်လိုက်မယ်။

ဆိုပါစို့။ ဒါဆိုရင် `p1.feet = 15` ဖြစ်သွားပါပြီ။ ဒီနည်းအတိုင်း `inches` အတွက် 11 ကိုရိုက်ထည့်ပေးမယ်ဆိုရင် `p1.inches = 11` ဖြစ်သွားပါလိမ့်မယ်။ ပြီးရင် `main( )` function ကိုပြန်လာပါပြီ။

- `total` object အတွက် `add_length( )` function ကို call ခေါ်ပါတယ်။ `argument` တွေက `p1` ၊ `p2` ဆိုတဲ့ object (2) ခုပါပဲ။ ဒီတော့ `x = p1` ၊ `y=p2` လို့ `argument` နေရာချင်းလဲလိုက်ပါတယ်။ ပြီးရင် ကွန်ပျူတာက `inches = x.inches + y.inches = p1.inches + p2.inches = 11 + 6.25 = 17.25` လို့ တွက်ယူပါတယ်။ `if block` နဲ့ တွေ့တဲ့အခါမှာ `inches = 17.25` က 12 ထက် ကြီးလား၊ ညီလားလို့မေးပါလိမ့်မယ်။ `<test expression>` ဟာ `true` ဖြစ်တဲ့အတွက် `inches = inches - 12 = 17.25 - 12 = 5.25` လို့တွက်ယူသလို `feet = feet + 1 = 0 + 1 = 1` လို့တွက်ယူပါတယ်။ နောက်ပြီး `feet += x.feet + y.feet`; statement ကနေ `feet = feet + x.feet + y.feet = 1 + 15 + 11 = 27` လို့ဆက်တွက်ပြီး `total` object ထဲက `feet` ကို 27 လို့ ကွန်ပျူတာကသိနေပါပြီ။
- ဒါဆိုရင် `total.showLength( )`; statement ကနေ `total` object ရဲ့ `feet` နဲ့ `inches` အသီးသီးကို `print` ထုတ်ပေးနိုင်ပြီဖြစ်ပါတယ်။ ဒီ `program` ကို `run` လိုက်မယ်ဆိုရင် ပုံ (၆. ၂၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။

```

C:\Quincy 99
Enter feet   : 15
Enter inches : 11

Piece1 = 15'-11"

Piece2 = 11'-6.25"

Total Length = 27'-5.25"

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

ပုံ (၆. ၂၁)

## 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;
}

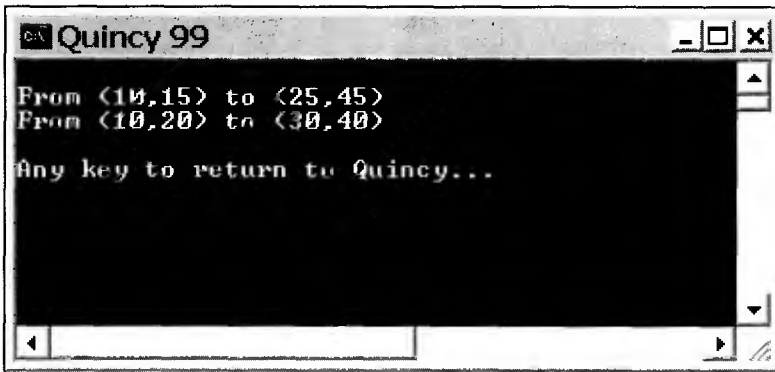
```

၂။ Ex6012.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်း main( ) ထဲမှာ Point class instance တစ်ခုဖြစ်တဲ့ p1 ကို define လုပ်ပြီး Point constructor ထဲကို data (2) ခု pass လုပ်ပေးလိုက်တယ်လေ။ x = xp = 10 နဲ့ y = yp = 15 တို့ပါပဲ။ ဒီတော့ p1.x = 10 နဲ့ p1.y = 15 လို့ main( ) ကသိနေပါပြီ။ ထို့နည်းတူ instance p2 ကိုလည်း define လုပ်ပြီး Point constructor ထဲကို x = xp = 20 နဲ့ y = yp = 45 တို့ကို pass လုပ်ပေးပါတယ်။ ဒီတော့ p2.x = 20 နဲ့ p1.y = 45 ဖြစ်သွားပါပြီ။
- ဒီတစ်ခါ main( ) ထဲမှာ Lines class instance တစ်ခုဖြစ်တဲ့ line1 ကို define လုပ်ပြီး Lines constructor ထဲကို object (2) ခု pass လုပ်ပေးလိုက်တယ်လေ။ ဒီတော့ Lines (Point st, Point sp) ဆိုတဲ့ constructor ထဲရွေးဝင်သွားပြီး start = st = p1 နဲ့ stop = sp = p2 တို့ကို assign လုပ်ပေးပါတယ်။ ဒါဆိုရင် line1.start.x = 10 နဲ့ line1.start.y = 15 နဲ့ line1.stop.x = 25 နဲ့ line1.stop.y = 45 ဖြစ်သွားပါပြီ။
- Lines class instance နောက်တစ်ခုဖြစ်တဲ့ line2 ကို define လုပ်ပြီး Lines constructor ထဲကို parameter (4) ခုကို pass လုပ်ပေးတဲ့အခါမှာ Lines (int x1, int y1, int x2, int y2) ဆိုတဲ့ constructor ထဲရွေးဝင်ပါလိမ့်မယ်။ ဒီတော့ x1 = 10 ၊ y1 = 20 ၊ x2 = 30 နဲ့ y2 = 40 တို့ကို assign လုပ်ပေးမှာပါ။ ပြီးတော့ရင် အဲဒီ constructor ထဲမှာပဲ Point class instance st(x1,y1) ကို define လုပ်ပါတယ်။ ဒီတော့ x = xp = x1 = 10 နဲ့ y = yp = y1 = 20 ကို pass လုပ်ပေးမှာပါ။ အခုဆိုရင် line2.st.x = 10 နဲ့ line2.st.y=20 ဖြစ်သွားမှာပါ။ ထို့နည်းတူ line2.sp.x = 30 နဲ့ line2.sp.y = 40 ဖြစ်နေပါလိမ့်မယ်။ အခုနေ st ကို private member ဖြစ်တဲ့ start နဲ့ assign လုပ်ပေးမယ်ဆိုရင် line1.start.x =

line1.st.x ဖြစ်သွားပါပြီ။ ထိုနည်းတူ line1.sp.y = line1.start.y ၊ line2.start.y = line2.st.x = 30 နဲ့ line2.sp.y = line2.stop.y = 40 တို့လည်းဖြစ်သွားမှာပါ။

- main ( ) ထဲမှာ line1.Draw ( ) ကို call ခေါ်မယ်ဆိုရင် Draw ( ) function ထဲဝင်သွားပြီး start.display ( ) ကို call ထပ်ခေါ်ပါတယ်။ ဒီတော့ display ( ) function ထဲဝင်လာပြီး x နဲ့ y ကို display လုပ်ခိုင်းပါတယ်။ အမှန်က line1.start.x နဲ့ line1.start.y တို့ကို display လုပ်ခိုင်းတဲ့သဘောပါပဲ။ ထိုနည်းတူ stop.display ( ) ဟာလည်း line1.stop.x နဲ့ line1.stop.y တို့ကို display လုပ်ပြမှာပါ။
- line2.display ( ) ဟာလည်း စောစောကပုံစံအတိုင်း display လုပ်ပြပါလိမ့်မယ်။ Ex6012.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၆. ၂၂) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၆. ၂၂)

# Copy Constructors

■ C++ program မှာ a new object တစ်ခုကို class ချင်းတူတဲ့ existing object နဲ့ initialize လုပ်ပေးဖို့အတွက်အသုံးပြုတဲ့ constructor ကို copy constructor လို့ခေါ်ပါတယ်။ ဒါမှမဟုတ် function တစ်ခုကနေ object တစ်ခုကို passing by value နည်းနဲ့ copy ကို pass လုပ်ရင်လည်း copy constructor ကိုအသုံးပြုပါတယ်။ နောက်ပြီး class object ကို by value အနေနဲ့ return ပြန်ပေးတဲ့အခါမှာလည်း copy constructor ကိုအသုံးပြုပါတယ်။ အဲဒီနည်းတွေကို Ex6013.cpp program မှာရေးပြထားပါတယ်။

// 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( );

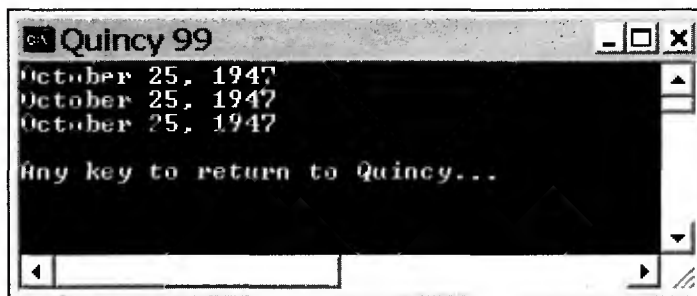
    Date newday = birthday;
    newday.display( );

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

    return 0;
}

```

- စစချင်း main( ) ထဲမှာ class Date instance တစ်ခုဖြစ်တဲ့ birthday object ကို define လုပ်ပြီး 10,25,1947 တို့ကို pass လုပ်ပေးပါတယ်။ ဒီချိန်မှာ Date constructor ထဲဝင်သွားပြီး m =10 ၊ d =25 နဲ့ yr =1947 တို့ကို assign လုပ်ပေးမှာပါ။ နောက်ပြီး mo = m =10 ၊ da = d= 25 နဲ့ yr =yr =1947 လို့ဆက်ပြီး assign လုပ်ပေးပါလိမ့်မယ်။ if (m != 0) ဆိုတဲ့ expression က true ဖြစ်တဲ့အတွက် if block ထဲဆက်ဝင်လာမှာပါ။ month = new char [len(mos[m-1])+1] = new char [strlen (mos[9])+1] = new char ["October" + 1] = new size[8] ကနေ memory size ကိုတွက်ပေးပါတယ်။ နောက်ပြီး strcpy (month, mos[m-1]); statement ကနေ "October" ကို month ထဲ ကော်ပီကူး ထည့်ပေးလိုက်ပါပြီ။ အဲဒါပြီးရင် main( ) ကိုပြန်လာမှာပါ။
- birthday.display( ) ကနေ call ခေါ်လိုက်တဲ့အခါမှာ display member function ထဲကို ဝင်လာပြီး month="October" ၊ da=25 နဲ့ yr=1947 တို့ကိုတွဲပြီး ကွန်ပျူတာမှာ display လုပ်ပြပါလိမ့်မယ်။ ဒီဥစ္စာဟာ date ကို display လုပ်တဲ့ ပထမဆုံးနည်းပါပဲ။
- အခုတစ်ခါ data သိပြီးသား birthday object ကို new Data object ဖြစ်တဲ့ newday နဲ့ assign လုပ်ပေးပါမယ်။ ဒါဆိုရင် birthday data တွေကို newday က လက်ခံရရှိသွားပါပြီ။ newday.display() ကို call ခေါ်မယ်ဆိုရင် ပထမနည်းကအဖြစ်ပဲရမှာပါ။ ဒီဥစ္စာဟာ date ကို display လုပ်တဲ့ ဒုတိယနည်းပါပဲ။
- နောက်ဆုံးနည်းက lastday ဆိုတဲ့ new object ကို create လုပ်ပြီး birthday object ကို argument အနေနဲ့ pass လုပ်ပေးတာပါ။ ဒီခါမှာ copy constructor ထဲဝင်သွားပြီး dt = birthday ဖြစ်သွားပါတယ်။ ဒီတော့ mo = dt.mo = birthday.mo =10 ၊ da = dt.da = birthday.da=25 နဲ့ yr = dt.yr = birthday.yr = 1947 ဖြစ်သွားပါပြီ။ ကျန်တာက အရင်အတိုင်းပါပဲ။ ပုံ (၆. ၂၃) မှာ Ex6013.cpp program ကို run ပြထားပါတယ်။



ပုံ (၆. ၂၃)

## 6.7 Conversion Constructors

parameter list မှာ entry တစ်ခုပဲပါတဲ့ constructor function ကို conversion constructor ခေါ်ပါတယ်။ parameter type ဟာ constructor class မျိုးမဟုတ်ရပါဘူး။ အောက်မှာဖော်ပြထားတဲ့ 6.14.cpp program ဟာဆိုရင် C standard time( ) function ကနေ return လုပ်ပေးတဲ့ time\_t value ကို Date class ဖြစ်အောင်ပြောင်းပေးတဲ့ program ပါပဲ ၊ လေ့လာကြည့်ပါ။

// Listing 6.14: 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* tim = localtime(&now);
```

```
    day = tim->tm_mday;
```

```
    month = tim->tm_mon + 1;
```

```
    year = tim->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;
```

```
}
```

```
};
```

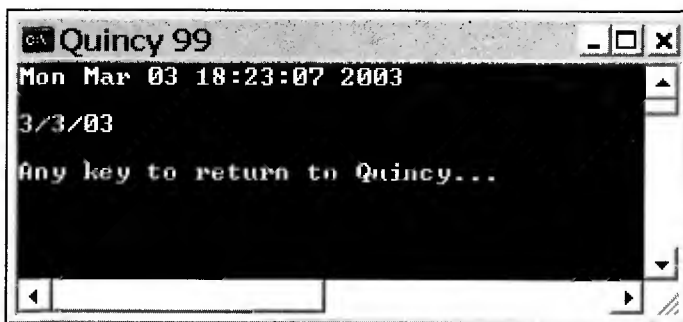
```

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;
}

```

၂။ Ex6014.cpp ကို လေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်း main( ) ထဲမှာ ဒီနေ့ရက်စွဲနဲ့အချိန်ကို system ကနေထုတ်ယူပြီး time( ) function ကနေ time\_t object အနေနဲ့ပြောင်းလိုက်ပါတယ်။ cout << asctime(gmtime(&now)) statement က asctime( ) function ကနေ std::tm structure ကို point လုပ်နေတဲ့ now အတွက် date & time string format နဲ့ display လုပ်ပြမှာပါ။
- Date dt(now); ဆိုတဲ့ statement ဟာ conversion constructor ကိုအသုံးပြုပြီးတော့ Date object တစ်ခုကို construct လုပ်တာပါပဲ။ ဒါဆိုရင် conversion constructor function ထဲမှာ Date format ကိုအသစ်ပြောင်းပေးပြီး dt.display( ); ကနေ display လုပ်ပြပါလိမ့်မယ်။ Ex6014.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၆. ၂၄) မှာပြထားတဲ့ အတိုင်းမြင်ရမှာပါ။



ပုံ (၆. ၂၄)

## 6.6 Member Conversion Functions

class တစ်ခုထဲမှာ define လုပ်ထားတဲ့ member conversion function တစ်ခုဟာ အဲဒီ class ရဲ့ object တစ်ခုကို data type မတူတဲ့ တစ်ခြား object တစ်ခုဖြစ်အောင် ပြောင်းလဲပေးနိုင်ပါတယ်။ ဒီ function ကို declare လုပ်တဲ့အခါမှာ class declaration ထဲမှာ အခုလိုမျိုးရေးရပါမယ်။

```
operator long( );
```

operator က C++ keyword တစ်ခုပါ။ long က converted ဖြစ်သွားမယ့် data type အတွက် type specifier ဖြစ်ပါတယ်။ member conversion function ရဲ့ header မှာ classname::operator long( ) ဆိုတာမျိုးကိုရေးရပါမယ်။ အောက်မှာရေးထားတဲ့ Ex6015.cpp program မှာ member conversion function အသုံးပြုနည်းကိုရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

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

```
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 << "Cumulative days since 1900 = " << since << endl;
    return 0;
}

```

၂။ Ex6015.cpp ကို trace လုပ်ကြည့်မယ်ဆိုရင်

- စတင်ချင်း main( ) ထဲမှာ Date class object တစ်ခုဖြစ်တဲ့ birthday ကို create လုပ်ပြီး Date constructor ထဲကို parameter (3) ခု pass လုပ်ပေးပါတယ်။ constructor ထဲမှာ month = m = 10 ၊ day = d = 25 နဲ့ year = y = 2003 တို့က assign ဖြစ်သွားပါပြီ။ main( ) ကိုပြန်ရောက်တဲ့အခါ long since = birthday; ဆိုတဲ့ statement ကနေ membe conversion function ထဲဝင်သွားပြီး

$x = \text{year} - 1900 = 2003 - 1900 = 103 \text{ years}$

$x * = 365 = 103 * 365 = 37595 \text{ days}$

$x += \text{year}/4 = 37595 + 2003/4 = 38095 \text{ days}$

ဒီဥစ္စာဟာ (4) နှစ်တစ်ကြိမ် တစ်ရက်တိုးတာကို ထည့်ပေါင်းထားပါ။

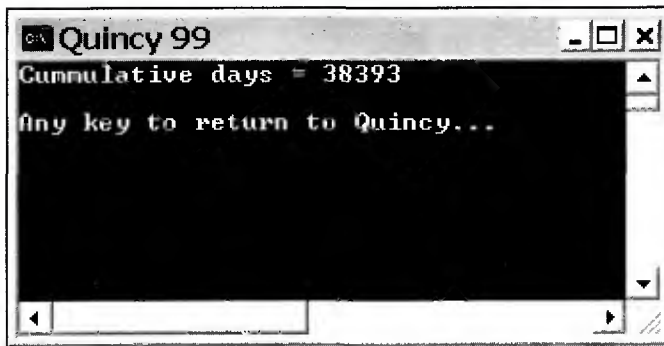
- for loop ထဲမှာ (9) လအတွက် ရက်တွေစုစုပေါင်းကို x ထဲမှာထပ်ထည့်ပေါင်းပါတယ်။
- $x += \text{days}[i] = 38095 + 31 + 28 + 31 + 30 + 30 + 31 + 31 + 30$   
 $= 38368 \text{ days}$

$x += \text{day} = 38368 + 25 = 38393 \text{ days}$  ဖြစ်ပါတယ်။

ဒါဆိုရင် birthday data type ကနေ long type ကိုပြောင်းသွားတာ တွေ့တယ်မဟုတ်လား။

ဒါဟာ member conversion function ကြောင့်ပါ။ Ex6015.cpp program ကို run

လိုက်မယ်ဆိုရင် ပုံ (၆. ၂၅) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၆.၂၅)

## Improved Member Conversion Function

၁။ Ex6016.cpp program မှာ `Date::operator customDate( )` function ကို `const` လို့ declare လုပ်ထားပါတယ်။ ဒီတော့ `customDate( )` ဟာ `Date` object နဲ့ `data value` ကို `modify` လုပ်ပုံ temporary object ဖြစ်တဲ့ `customDate` ကိုအသုံးပြုပြီး `data modify` လုပ်သွားတာကိုသတိပြုကြည့်ပါ။ `main( )` function မှာ စတင်ချင်း `Date` object `dt` နဲ့ `customDate` object `cd` တို့ကိုသီးခြားစီ `construct` လုပ်ပြီး `dt` ကို `cd` နဲ့ `assign` လုပ်ပေးခြင်းအားဖြင့် `Date` `data value` ဟာ `customDate` `data type` ကို ပြောင်းသွားပါလိမ့်မယ်။

// 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 << endl
        << "Total days in " << mo << " months equal "
        << 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;
}

```

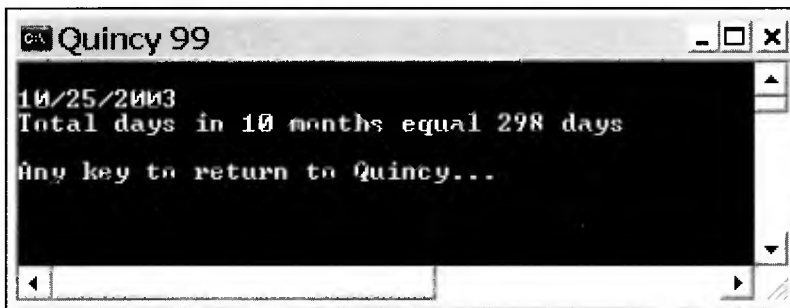
```

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 လိုက်မယ်ဆိုရင် ပုံ (၆.၂၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၆.၂၆)

## ၆.၉ Using Friends

တစ်ခါတစ်ရံ ကျွန်တော်တို့တွေဟာ class တစ်ခုရဲ့ private member တွေကို အဲဒီ class member မဟုတ်တဲ့ function တွေကနေ call ခေါ်တာမျိုးကိုလုပ်ချင်ပါတယ်။ အဲဒါမျိုးလုပ်လို့ရအောင် C++ ထဲမှာ friend function ဆိုတာကိုထည့်ပေးထားပါတယ်။ friend function တစ်ခုကို define လုပ်နည်းဟာ non-member function တစ်ခုကိုလုပ်သလိုပါပဲ။ တစ်ခုပဲသတိထားရမှာက class declaration ထဲမှာ prototype function ပုံစံမျိုးနဲ့ရေးပြီး ထိပ်စည်းမှာ friend keyword ကိုထည့်ပေးရမှာဖြစ်ပါတယ်။ နမူနာရေးပြထားတဲ့ Ex6017.cpp program ကိုလေ့လာကြည့်ရအောင်။

// 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)
```

```
            { da = d; yr = y; }
```

```
        void display( ) const
```

```
            { cout << endl << yr << '-' << da; }
```

```
        friend Date;
```

```
};
```

```
class Date
```

```
{
```

```
    int mo, da, yr;
```

```
    public:
```

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

```
            { mo = m; da = d; yr = y; }
```

```
        operator CustomDate( );
```

```
};
```

```
Date::operator CustomDate( )
```

```
{
```

```
    static int days[] = {31,28,31,30,31,30,  
                        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;
```

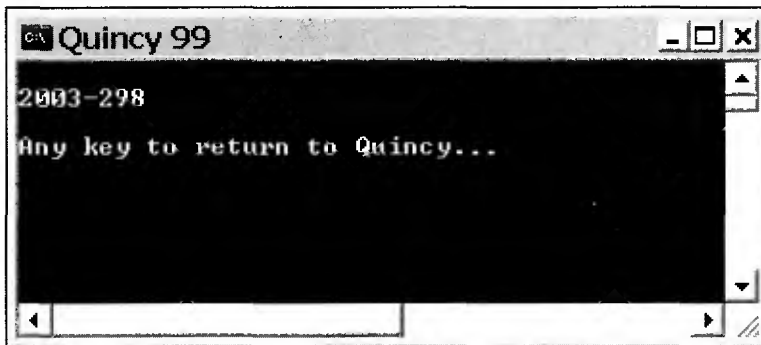
```
}
```

```

int main( )
{
    Date  dt(10,25,2003);
    CustomDate cd(dt);
    cd.display( );
    cout << endl;
    return 0;
}

```

Ex6017.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၆. ၂၇) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၆. ၂၇)

## 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    ca1(6,55), ca2(2,75);
```

```
    truck  tr1(100,45), tr2(120,75);
```

```
    cout << "Comparing car1 and truck1\n";
```

```
    int x = spGreater(ca1,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";
```

```

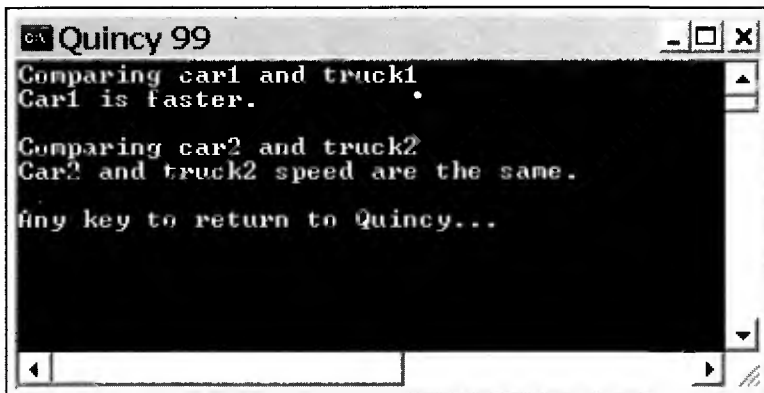
    cout << "\nComparing car2 and truck2\n";
    x = spGreater(ca2,tr2);
    if (x < 0)
        cout << "Truck2 is faster.\n";
    else if (x == 0)
        cout << "Car2 and truck2 speed are the same.\n";
    else
        cout << "Car2 is faster.\n";

    return 0;

}

```

Ex6018.cpp ကိုလေ့လာကြည့်မယ်ဆိုရင် spGreater( ) function ဟာ class car သို့မဟုတ် class truck မဟုတ်တဲ့အတွက် ပုံမှန်အားဖြင့်အဲဒီ class တွေရဲ့ member တွေကိုအသုံးပြုလို့မရပါဘူး။ ဒါပေမယ့် friend class လို့ကြေငြာထားလို့ call ခေါ်လို့ရသွားပါပြီ။ ပုံ (၆. ၂၈) မှာ Ex6018.cpp program ကို run ပြထားပါတယ်။



ပုံ (၆. ၂၈)

## ၆.၁၀ Friend Functions

၁။ တစ်ခါတစ်ရံမှာ ကျွန်တော်တို့ class တစ်ခုလုံးကို တစ်ခြား class ရဲ့ friend အဖြစ်လုပ်ပေးချင်ဘူးဆိုရင်

(တစ်နည်းအားဖြင့် အဲဒီလောက်မလိုဘူးဆိုရင်) current class ရဲ့ member data တွေကို read/write လုပ်ဖို့ အတွက်တစ်ခြား class ထဲက friend function ကိုပဲအသုံးပြုရင်ရပါတယ်။ Ex6019.cpp program မှာ friend function အသုံးပြုနည်းကိုဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// 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 };



```

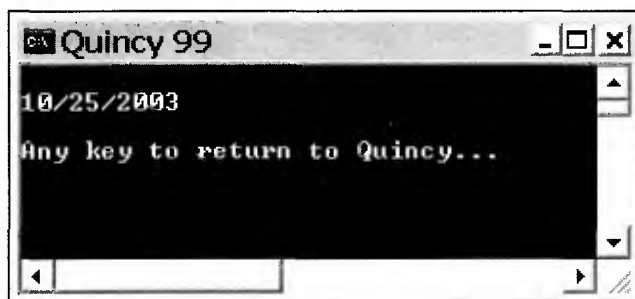
yr = cd.yr;
da = cd.da;

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

int main( )
{
    Date dt (CustomDate(298, 2003));
    dt.display();
    cout << endl;
    return 0;
}

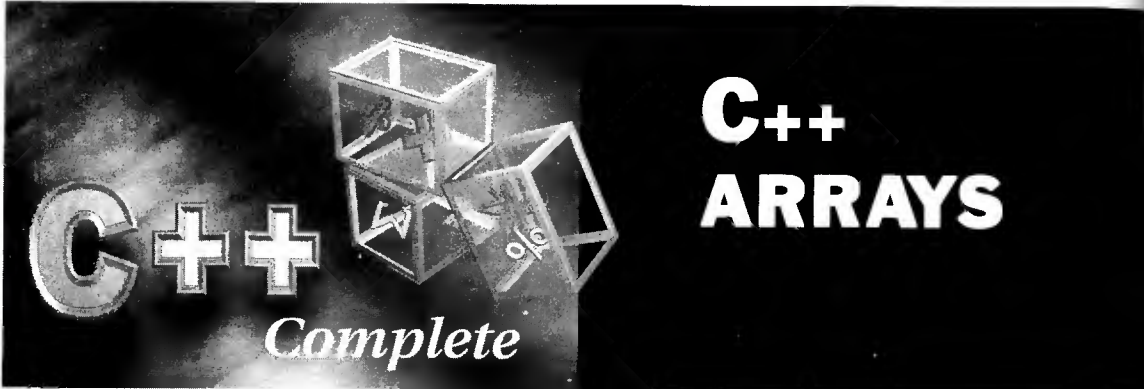
```

Ex6019.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၆. ၂၉) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၆. ၂၉)

# Chapter 7

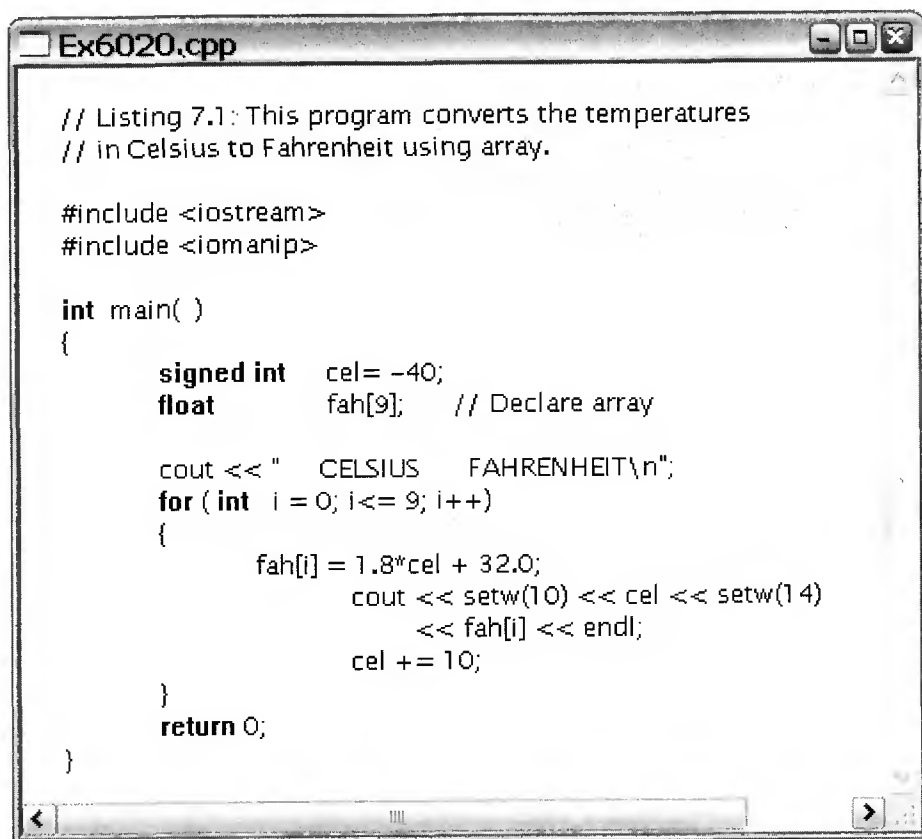


multiple data item တွေကို group လုပ်ချင်ရင် structure လိုပဲအသုံးပြုလို့ရတာက array ပါပဲ။ structure တွေဟာ type မတူတဲ့ item တွေကို group လုပ်ပေးပါတယ်။ structure တစ်ခုကို access လုပ်ချင်ရင် နာမည်ခေါ်ပြီးအလုပ်လုပ်ရပါတယ်။ array မှာကျတော့ index number တွေကိုအသုံးပြုရမှာပဲ။ C++ မှာအသုံးပြုတဲ့ array တွေဟာ int သို့မဟုတ် float တို့လို simple type မျိုးတွေကို group လုပ်ပေးနိုင်သလို user-defined type တွေဖြစ်တဲ့ structure တို့ class object တို့ကိုလည်း group လုပ်နိုင်ပါတယ်။ သာမန် variable တွေကို array လုပ်ပြီးသုံးလို့ရသလို object တွေကိုလည်း array အနေနဲ့အသုံးပြုလို့ရပါတယ်။ စတင်လွယ်ကူတဲ့ဥစ္စာကနေစပြီး တစ်ဆင့်ချင်းလေ့လာကြည့်ရအောင်။

## ၇.၁ Array Basics

multiple data item တွေကို common name တစ်ခုပေးပြီးအလုပ်လုပ်ချင်တယ်ဆိုရင် array ကို အသုံးပြုရမှာပါ။ array ဆိုတာ တစ်ကယ်တော့ရိုးရိုး variable တွေနဲ့သိပ်မထူးပါဘူး။ ထူးတာက array name

နောက်မှာ size specification ပိုပါတာပဲထူးပါတယ်။ array ကိုကြေငြာချင်ရင် array name နောက်က square brackets [ ] ထဲမှာ non-negative integer (အနှုတ်တန်ဖိုးမဟုတ်တဲ့ ကိန်းပြည့်တစ်ခု) ကိုထည့်ကြေငြာလိုက်ရင် array ဖြစ်သွားတာပါပဲ။ အဲဒီကိန်းကို subscript လို့ခေါ်ပါတယ်။ one-dimensional array တစ်ခုကို ကြေငြာချင်ရင် square brackets ထဲမှာ subscript တစ်ခုရေးထည့်ပေးပါ။ ဥပမာ myarray[10] တာဟာ one-dimensional array တစ်ခုပါပဲ။ array ရဲ့နာမည်က myarray ဖြစ်ပြီးတော့ subscript က 10 ဖြစ်ပါတယ်။ array element က (10) ခုပေါ့။ first element က myarray[0] ဖြစ်ပြီး last element က myarray[9] ဖြစ်ပါတယ်။ two-dimensional array ဆိုရင် subscript (2) ခုကို square brackets (2) ခု ထဲမှာထည့်ရေးရမှာပါ။ ဥပမာ myarray[5][5] ဟာ two-dimensional array တစ်ခုပါပဲ။ ပုံ (၇. ၁) ကဖော်ပြထားတဲ့ Ex701. cpp program ဟာဆိုရင် array ကိုအသုံးပြုပြီး temperature conversion table တစ်ခုကို create လုပ်ထားတာဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။



```
// Listing 7.1: This program converts the temperatures
// in Celsius to Fahrenheit using array.

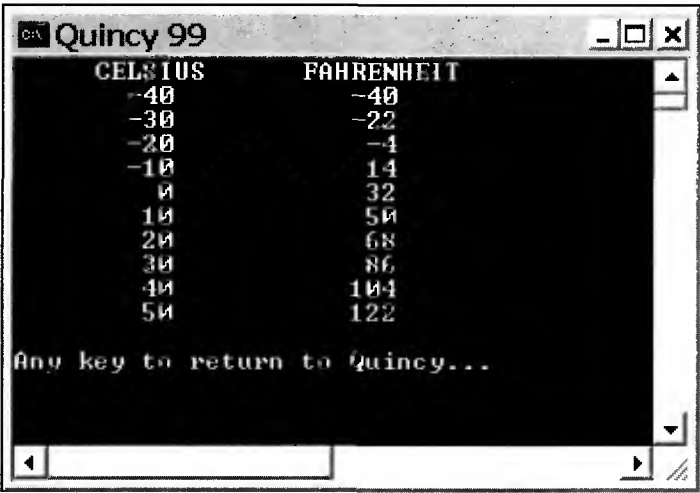
#include <iostream>
#include <iomanip>

int main( )
{
    signed int    cel= -40;
    float         fah[9];    // Declare array

    cout << "    CELSIUS    FAHRENHEIT\n";
    for ( int    i = 0; i <= 9; i++)
    {
        fah[i] = 1.8*cel + 32.0;
        cout << setw(10) << cel << setw(14)
              << fah[i] << endl;
        cel += 10;
    }
    return 0;
}
```

ပုံ (၇. ၁)

ဒီ program ကိုလေ့လာကြည့်မယ်ဆိုရင် စတင်ချင်းမှာ fah ကို array လို့ကြေငြာထားပါတယ်။ ဒီ fah array ထဲမှာ element (9) ခုပါပါတယ်။ for loop ကိုအသုံးပြုပြီး cel = -40 ကနေ cel = 40 အထိ (9) ခု တိတိတွက်ပြီးတော့ fahrenheit ဖြစ်အောင်ပြောင်းပေးသွားပါလိမ့်မယ်။ တစ်ကြိမ်တွက်ပြီးတိုင်း fahrenheit ကို တွေ့ကို fah[0]၊ fah[1]၊ fah[2] အစရှိတာတွေထဲမှာ သိမ်းထားတဲ့အတွက် အချိန်မရွေးပြန်ခေါ်ကြည့်လို့ရပါတယ်။ cel အတွက်တော့ နောက်ဆုံးတန်ဖိုးဖြစ်တဲ့ cel = 40 ကိုပဲ ကွန်ပျူတာကသိမှာပါ။ Ex701.cpp program (၇.၂) မှာ run ပြထားပါတယ်။



ပုံ (၇.၂)

## ၇.၂ Initializing an Array

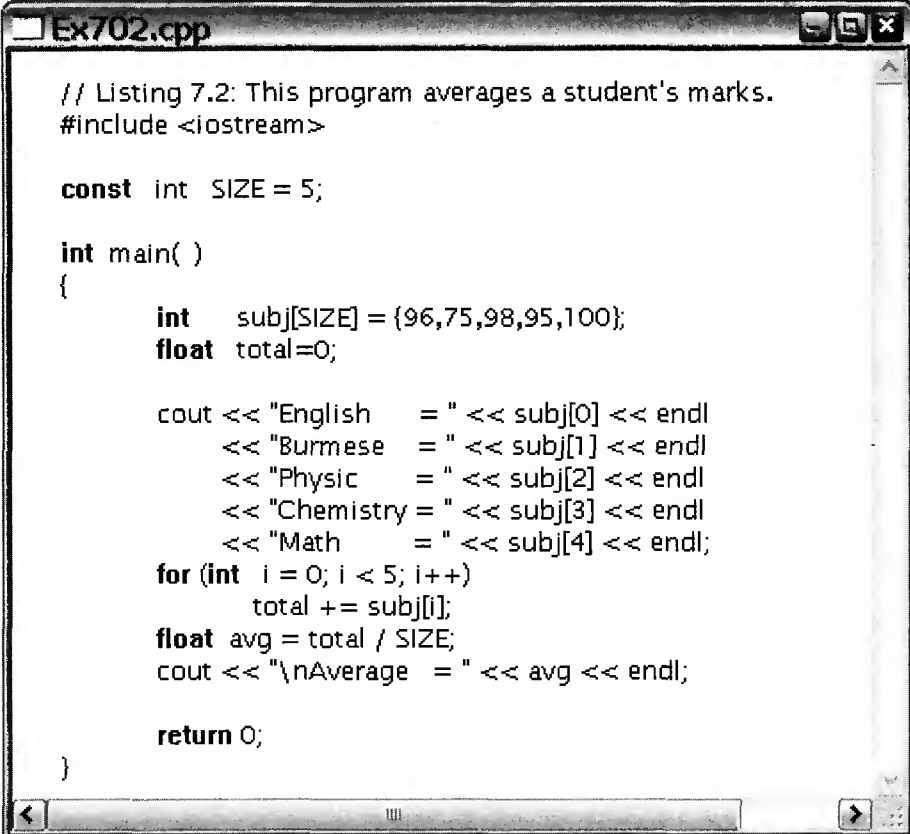
C++ program တွေမှာ array တစ်ခုရဲ့စဦးတန်းဖိုးတွေကို initialize လုပ်ပေးချင်ရင် array မှာပါဝင်တဲ့ subscript အရေအတွက်အတိုင်း အစီအစဉ်တကျတန်းစီပြီး (comma ခံပြီး) ရေးချသွားရပါမယ်။ data item တွေကို brace တွေနဲ့ပိတ်ပေးရမှာပါ။

```
float    x [6] = { 0, 1, 0.25, -5, 1.0, 7.5 };
char     color [3] = { 'R', 'E', 'D' };
```

အောက်မှာဖော်ပြထားတဲ့ နမူနာပုံစံတွေကိုလေ့လာကြည့်ရင် initial assignment တွေကို ကွန်ပျူတာက ခုလိုနားလည်ပါတယ်။

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	

ကောင်းပြီ ၊ ပုံ (၇.၃) မှာရေးထားတဲ့ Ex702.cpp program မှာ array subj[ ] ကို initialize လုပ်ထားတာကိုလေ့လာကြည့်ပါ။



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

const int SIZE = 5;

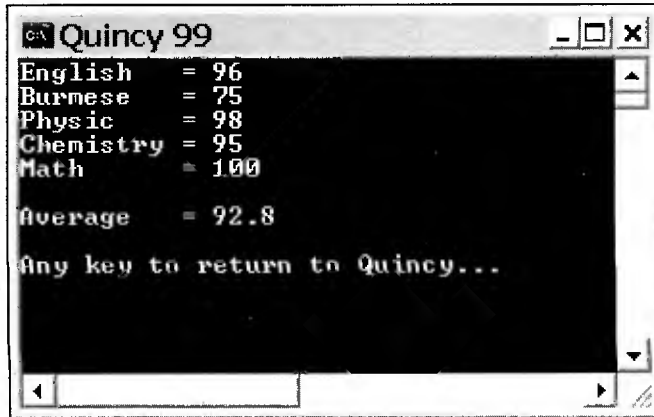
int main( )
{
    int subj[SIZE] = {96,75,98,95,100};
    float total=0;

    cout << "English   = " << subj[0] << endl
         << "Burmese   = " << subj[1] << endl
         << "Physic    = " << 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;
}
```

ပုံ (၇.၃)

Ex702.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် subj ကို array တစ်ခုလို့ကြံငြာပြီး initialize လုပ်ထားပါတယ်။ array size ကို `SIZE = 5` လို့ကြံငြာထားပါတယ်။ average ရှာတာကို `float avg = total/SIZE;` statement ကနေတွက်ပေးမှာပါ။ avg ကို display လုပ်ခိုင်းရင် အဖြေကိုတွေ့ရမှာပါပဲ။ ပုံ (၇.၄) မှာ Ex702.cpp ကို run ပြထားပါတယ်။



ပုံ (၇.၄)

## Substituting Strings

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex703.cpp program မှာ char array တွေဖြစ်တဲ့ str1 နဲ့ str2 တို့ကို initialize အရင်လုပ်ပေးပြီး သင့်တော်သလို display လုပ်ခိုင်းထားပါတယ်။

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

```
#include <iostream>
```

```
int main( )
```

```
{
```

```
    char str1[ ] = { "BASKET" };
```

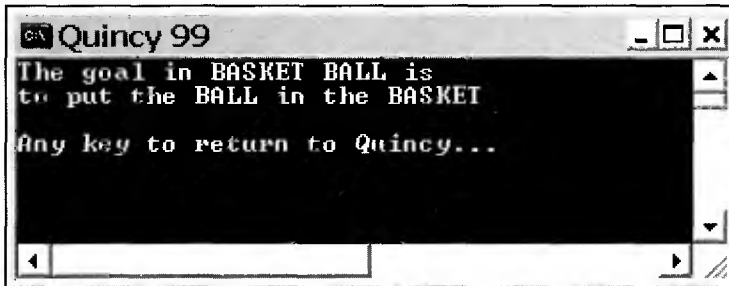
```
    char str2[ ] = { 'B', 'A', 'L', 'L', '\0' };
```

```

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

```

Ex703.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၅) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၇. ၅)

## Finding the Maximum and Minimum Values

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex704.cpp program မှာ array တစ်ခုကနေ minimum/maximum value တွေကိုရှာပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

```

// 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,
```

```
        25, 45, 2, 84, -3,
```

```
        7, -9, 32, 16, 54,
```

```
        82, 51, 99, -7, 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]);
```

```
    }
```

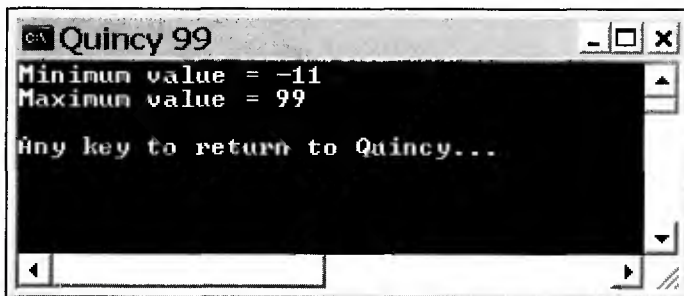
```
    cout << "Minimum value = " << minval << endl;
```

```
    cout << "Maximum value = " << maxval << endl;
```

```
    return 0;
```

```
}
```

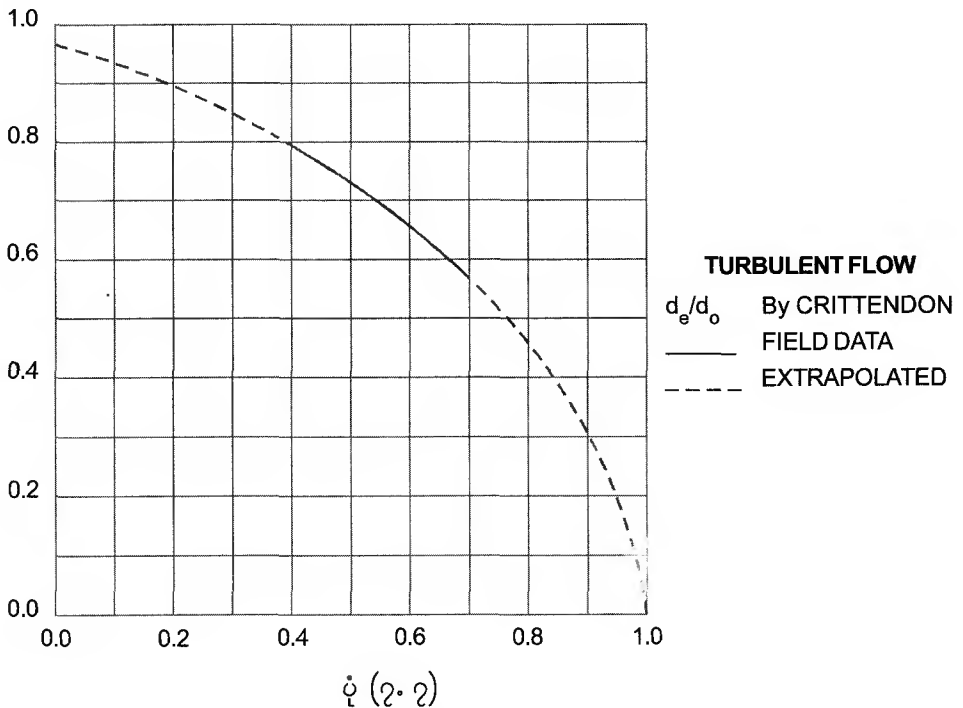
၂။ Ex704.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇.၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၇.၆)

# Processing an Array

အခုတစ်ခါတင်ပြမှာကတော့ array တစ်ခုကို ဘယ်လိုကိုင်တွယ်ပြီး use လုပ်ရမလဲဆိုတာကို ကျွန်တော် ချက်ချင်းပြပါမယ်။ အောက်မှာဖော်ပြထားတဲ့ Ex705.cpp program ဟာဆိုရင် linear interpolation method အသုံးပြုပြီးကြိုက်တဲ့ graph တစ်ခုကို ကွန်ပျူတာနဲ့ဖတ်လို့ရအောင်လုပ်ပေးနိုင်ပါတယ် ၊ လေ့လာကြည့်ပါ။



// 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( )  
{
```

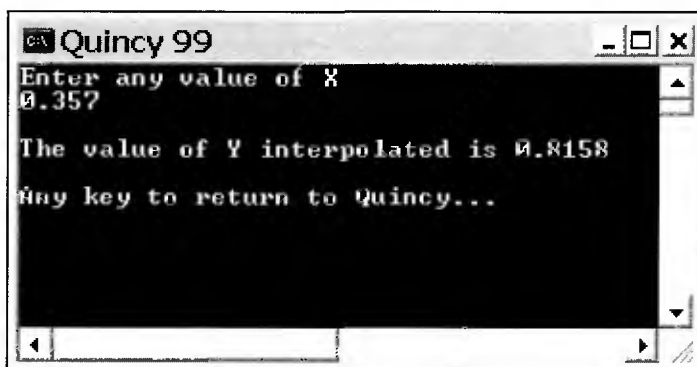
```
    float x[ ]= { 0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1.0 };  
    float y[ ]= { .97,.94,.89,.85,.79,.73,.66,.56,.46,.3,0 };  
    float anyx, ycal;
```

```

cout << "Enter any value of X\n";
cin >> anyx;
if ( anyx <= x [0] )
    ycal = y [0];
else if( anyx >= x [10])
    ycal = y [10];
else
    for (int i= 1; i < 10; ++i )
        if ( x [i] > anyx )
        {
            ycal = 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 " << ycal << endl;
    return 0;
}

```

၂။ Ex705.cpp program ရဲ့ရည်ရွယ်ချက်က ပုံ (၇. ၇) မှာဖော်ပြထားတဲ့ graph ကို ကွန်ပျူတာကနေ ဖတ်ပြီး တွက်ချက်ပေးနိုင်အောင်ရေးထားတဲ့ program တစ်ခုပါ။ X တန်ဖိုးတစ်ခုကို data ထည့်ပေးလိုက်တာနဲ့ ကွန်ပျူတာကအလိုလို Y တန်ဖိုးကိုတွက်ပြီး အဖြေထုတ်ပေးမှာပါ။ graph ကိုလက်နဲ့ထောက်ပြီး ဖတ်လိုက်သလိုပါပဲ။ ဒီ program ဟာဘယ်လို graph မျိုးကိုမဆို အသုံးပြုလို့ရပါတယ်။ ဒီ program ဘယ်လိုအလုပ်လုပ်သွားလဲဆိုတာကို ကျွန်တော်ရှင်းမပြော့ပါဘူး။ စာဖတ်သူကိုယ်တိုင်ပဲ trace လုပ်ကြည့်ပါ။ Ex705.cpp program ကို run လိုက် မယ်ဆိုရင် ပုံ (၇. ၈) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၇. ၈)

# Generating a Pascal triangle

အောက်မှာဖော်ပြထားတဲ့ Ex705.cpp program ဟာဆိုရင် array အသုံးပြုပြီး Pascal triangle ထုတ်ဖော်လုပ်အောင်လုပ်ပေးတဲ့ program ပါပဲ ၊ လေ့လာကြည့်ပါ။

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

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

const int MAXROW = 10;

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;

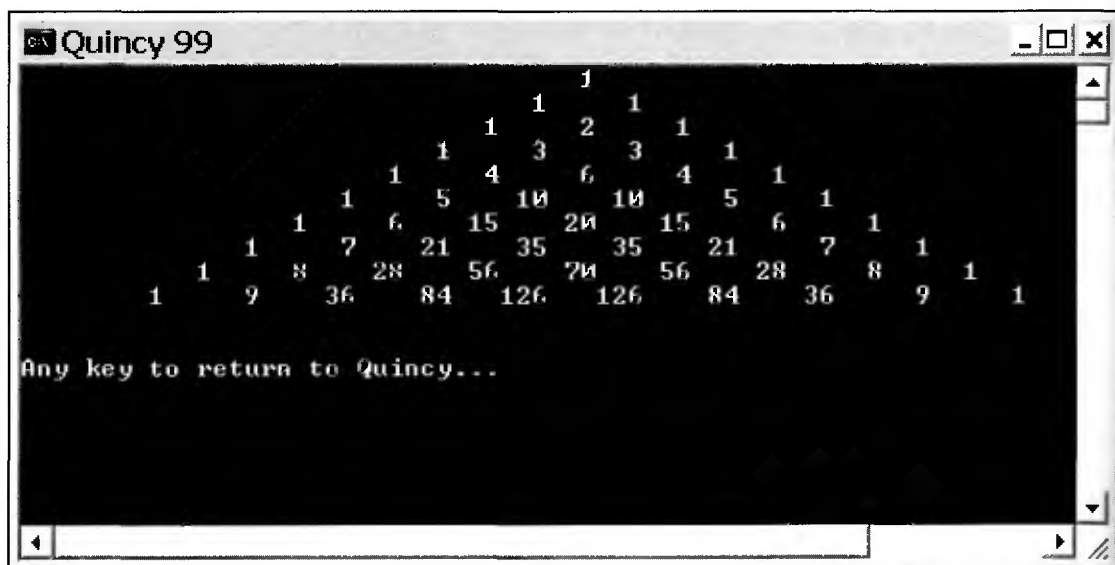
        for (int j=0; j<row_no; j++)
            cout << setw(6) << row[j];
        cout << endl;
        if (row_no == MAXROW) break;
        for (int k=row_no; k>=1; --k)
            row[k] += row[k-1];
    }
}
```

```

    cout << endl;
    return 0;
}

```

၂။ Ex706.cpp program ကို မကည လိုက်မယ်ဆိုရင် ပုံ (၇.၉) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ စာဖတ်သူ ကိုယ်တိုင်ပဲ program ကို trace လုပ်ကြည့်ပါ။



ပုံ (၇.၉)

## ၇.၄ Passing Array to a Function

ဒီတစ်ခါတော့ function ထဲကို array တွေ pass လုပ်တဲ့နည်းကို ကျွန်တော်တင်ပြမှာပါ။ အလွယ်ဆုံး ပြောရရင် function definition မှာ array name + [ ] ကို argument အနေနဲ့ကြေငြာပေးရင်ပြီးတာပါ။ ဒါပေမယ့် argument ဖြစ်တဲ့ array name မှာ square bracket [ ] အလွတ်ပဲဖြစ်နေရပါမယ်။ subscript တွေဘာတွေ ထည့်စရာမလိုပါဘူး။ function call ခေါ်တဲ့အခါကျတော့ argument ဟာ array name ပဲသုံး ပါမယ်။ square bracket [ ] တောင် ပါစရာမလိုတော့ပါဘူး။ Ex707.cpp program ကိုလေ့လာကြည့်ပါ။

// 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;
```

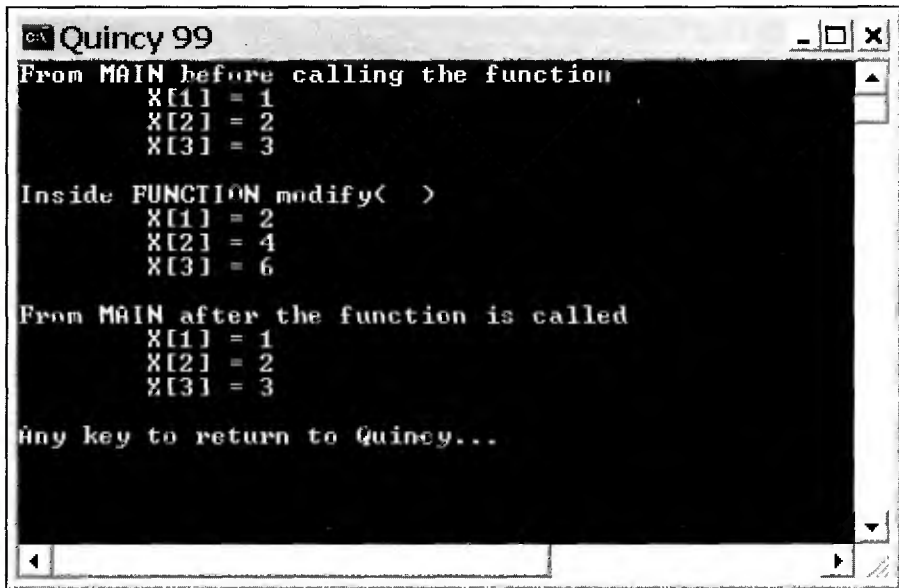
```
void modify(int x[ ])
```

```
{  
    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 ထဲကနေ function modify ထဲကိုဝင်လာတဲ့ array element တွေဟာ modify( ) function ထဲမှာ အပြောင်းအလဲတစ်ခုကြုံလိုက်တာနဲ့ main( ) ထဲကိုပြန်ရောက်လာတဲ့အခါမှာ array element တွေရဲ့တန်ဖိုးတွေက အသစ်ဖြစ်သွားတာကိုတွေ့ရမှာပါ။ ဒါဟာ array ကိုသုံးပြီးတော့ function တစ်ခုကနေ တစ်ခုကို data pass လုပ်တဲ့နည်းပါပဲ။ Ex707.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၁၀) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၇. ၁၀)

## Sorting

၁။ ပုံ (၇. ၁၁) မှာဖော်ပြထားတဲ့ Ex708.cpp program ဟာဆိုရင် array အသုံးပြုပြီး bubble sorting လုပ်နည်းကိုရေးထားတဲ့ program ပါပဲ ၊ လေ့လာကြည့်ပါ။ ဒီ program ရဲ့ရည်ရွယ်ချက်က

- ဂဏန်းတွေကို ငယ်စဉ်ကြီးလိုက်စီပေးမှာဖြစ်ပါတယ်။ ဂဏန်းအရေအတွက် (25) လုံးကိုစီပေးနိုင်ပါတယ်။ ကွန်ပျူတာမှာ How many numbers ? လို့ prompt လုပ်တဲ့အခါမှာ (6) ကို ရိုက်ထည့်မယ်ဆိုရင်  $n = 6$  ဖြစ်သွားပါပြီ။  $n$  ဟာ sort လုပ်မယ့် ဂဏန်းအရေအတွက်ပေါ့။

```
Ex708.cpp

// Listing 7.8: This program sorts through a list of
// numbers in ascending order using bubble sort method.

#include <iostream>

void bubbleSort (int x[], int n)
{
    for (int i=0; i < (n-2); i++)
        for (int j=0; j < (n-i-1); j++)
            if ( x[j] > x[j+1] )
            {
                int temp = x[j];
                x[j] = x[j+1];
                x[j+1] = temp;
            }
}

int main()
{
    int    n, num[25];

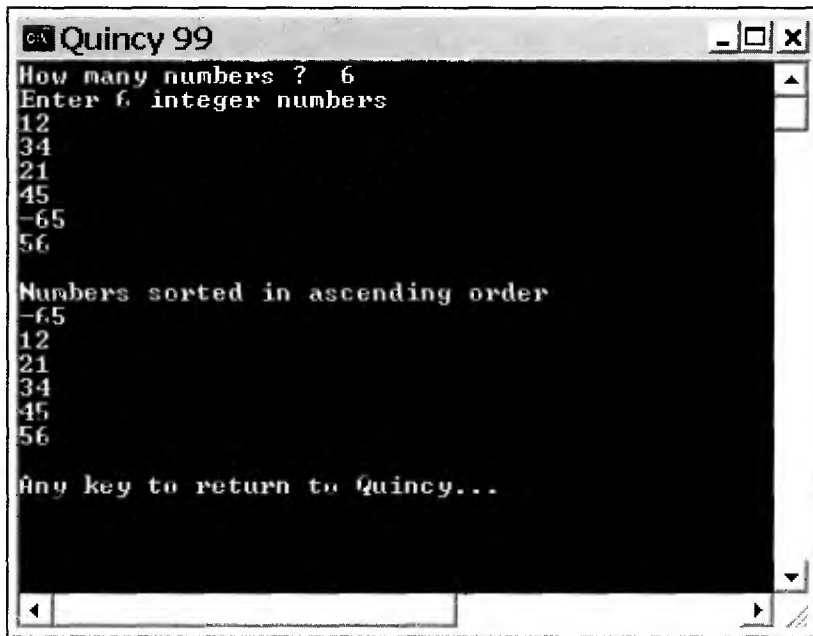
    cout << "How many numbers ? ";
    cin >> n;
    cout << "Enter " << n << " integer numbers\n";
    for (int i=0; i<n; i++)
        cin >> num[i];
    bubbleSort (num, n);
    cout << "\nNumbers sorted in ascending order";
    for (int j = 0; j < n; j++)
        cout << endl << num[j];
    cout << endl;
    return 0;
}
```

ပုံ (၇. ၁၁)

- အဲဒီဂဏန်း (6) လုံးအတွက် data တောင်းတဲ့အခါမှာ 12 34 21 45 -65 56 လို့ဆိုပြီး ဂဏန်းတွေထည့်ပေးလိုက်ပါမယ်။ ဒါဆိုရင် bubbleSort ( ) function ကို call ခေါ်ပါတယ်။ function argument တွေက num နဲ့ n ပါ။ num[ ] လို့ မရေးရဘူးဆိုတာကိုသတိပြုပါ။



- bubbleSort( ) function မှာကျတော့ argument တွေကို ( int x[ ], int n) စသည်ဖြင့် ရေးရပါတယ်။ array num[ ] အတွက် dummy argument ကို int x[ ] လို့ရေးထားတာကိုသတိပြုကြည့်ပါ။ function body ထဲကသင်္ချာတွေကတော့ numerical method ပါပဲ။ ရှင်းမပြတော့ပါဘူး။ Ex708.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၁၂) မှာပြထားတဲ့ အတိုင်းမြင်ရမှာပါ။



```
Quincy 99
How many numbers ? 6
Enter 6 integer numbers
12
34
21
45
-65
56

Numbers sorted in ascending order
-65
12
21
34
45
56

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

ပုံ (၇. ၁၂)

## Counting Frequencies

- ၁။ Ex708.cpp program ဟာဆိုရင် ကျွန်တော်တို့ keyboard ကနေရိုက်သည့်ပေးလိုက်တဲ့ character တွေထဲမှာ digit ဘယ်နှစ်လုံး၊ white space ဘယ်နှစ်ခု၊ non-white space ဘယ်နှစ်ခုပါလဲဆိုတာကို ရေတွက်ပေးမှာဖြစ်ပါတယ်။

// Listing 7.8: 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;
```

```
}
```

- စတင်ချင်း main( ) function ထဲမှာ digit[0] ကနေ digit[9] အထိ array variable (10) ခု လုံးကို သုညတွေလို့ initialize လုပ်ပေးပါတယ်။ နောက်ပြီး std C function getch( ) ကို အသုံးပြုပြီး ch တန်ဖိုးကိုဖတ်ပေးပါတယ်။ ဥပမာ ch=1 ဆိုပါစို့။ ဒါဆိုရင် ch=1 ဟာ ESC=27 နဲ့မညီတော့ဘူး။ while loop ထဲကိုဝင်လာပါပြီ။ ch=1 ဟာ 0 နဲ့ 9 ကြားကဖြစ်ရင် digit[1] = 0 + 1 လုပ်ပေးလိုက်ပါပြီ။ ပြီးရင် while loop အစကိုပြန်သွားပြီး နောက် ch တန်ဖိုးတစ်ခုကို ဖတ်ပါလိမ့်မယ်။ ဒီတစ်ခါလည်း ch=1 ပဲဆိုပါစို့။ ဒါဆိုရင် digit[1] = 1 + 1 = 2 ဖြစ်သွားပါလိမ့်မယ်။ တစ်ကယ်လို့ ch ဟာ white space တစ်ခုဖြစ်ခဲ့ရင် wh\_spcs အရေအတွက်ကို တစ်ခုတိုးပေးမှာပါ။ ch က digit လည်းမဟုတ် ၊ white space လည်းမဟုတ်တဲ့ character တစ်ခုခုဖြစ်မယ်ဆိုရင် nonwh\_spcs အရေအတွက်ကို တစ်ခုတိုးပေးမှာဖြစ်ပါတယ်။
- ကျွန်တော်တို့ data သွင်းပေးတာကိုရပ်ချင်ရင် Escape key ကိုနှိပ်ပေးပါ။ အခုနေ Ex708.cpp program ကို run လိုက်မယ်ဆိုရင် အဖြေကို ပုံ (၇. ၁၃) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။

```

Quincy 99
Enter a line of characters
1111 222 33333 577 aaa zzz+
digit[0] = 0
digit[1] = 4
digit[2] = 3
digit[3] = 5
digit[4] = 0
digit[5] = 1
digit[6] = 0
digit[7] = 1
digit[8] = 0
digit[9] = 1

White spaces = 5
Non-white spaces = 6

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

ပုံ (၇. ၁၃)

# Multidimensional Arrays

multidimensional array ကြေငြာပုံကြေငြာနည်းကတော့ one-dimensional array တစ်ခုကို ကြေငြာပုံနဲ့ ကွာပါဘူး။ ပိုလာတဲ့ dimension တွေအတွက် square bracket [ ] ထပ်တိုးပေးရုံပါပဲ။ two-dimensional array ဆိုရင် square bracket (2) ခု၊ three-dimensional array ဆိုရင် (3) ခု စသည်ဖြင့်ရေးသွားရပါမယ်။ အောက်မှာဖော်ပြထားတဲ့ ဥပမာတွေဟာ multidimensional array တွေကိုရေးတဲ့ပုံစံပါ။ လေ့လာကြည့်ပါ။

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

multidimensional array တစ်ခုရဲ့ element တွေကို initialize လုပ်မယ်ဆိုရင် အခုလိုလုပ်လို့ရပါတယ်။

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

ဒီပုံစံမှာဆိုရင် x ဟာ array ရဲ့ name ပါ။ (3) rows နဲ့ (4) columns ပါတဲ့ two-dimensional array တစ်ခုကိုဆိုလိုတာပါ။ ဒီ array element တွေကို ကွန်ပျူတာကနေ assign လုပ်သွားပုံကိုကြည့်ပါ။

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 နောက်ထပ်ပုံစံတစ်မျိုးနဲ့ ရေးလို့ရပါတယ်။ အခုလိုရေးရင် row နဲ့ column တွေကို ခွဲပြီးတော့ မြင်ရတာပေါ့။

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

```
Ex709.cpp

// Listing 7.9: This program displays students
// vs their marks chart.

#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},
                                                { 75,60,58,60,70},
                                                { 45,50,55,75,50},
                                                { 20,40,35,48,45}
    };

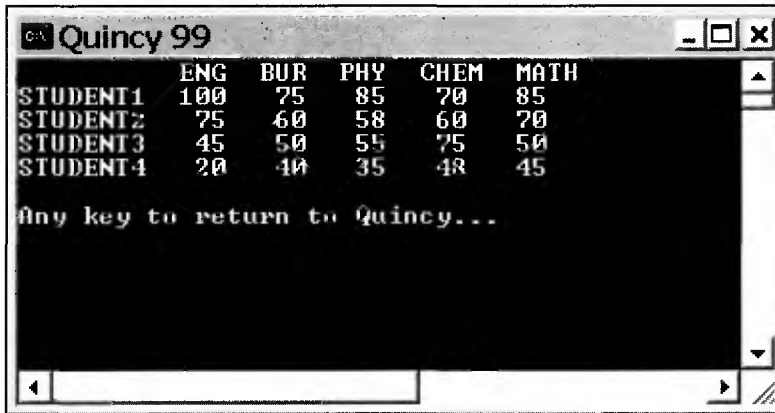
    cout << "      ENG BUR PHY CHEM MATH";
    for (int i=0; i<STUDENTS; i++)
    {
        cout << "\nSTUDENT" << i+1;
        for (int j=0; j<SUBJECTS; j++)
            cout << setw(5) << marks[i][j];
    }
    cout << endl;
    return 0;
}
```

ပုံ (၇.၁၄)

ပုံ (၇.၁၄) မှာဖော်ပြထားတဲ့ Ex709.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်း marks ဆိုတဲ့နာမည်နဲ့ two-dimensional array တစ်ခုကို initialize လုပ်ပါတယ်။ row နဲ့ column dimension တွေကိုလည်းကြေငြာထားပါတယ်။ နောက်ပြီး heading ကို display လုပ်ခိုင်းပါတယ်။ nested for loop ထဲမှာ row (4) ခု၊ column (5) ခုကိုရိုက်ခိုင်းပြီးတော့ program ပြီးသွားပါပြီ။

- Ex709.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၁၅) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



	ENG	BUR	PHY	CHEM	MATH
STUDENT1	100	75	85	70	85
STUDENT2	75	60	58	60	70
STUDENT3	45	50	55	75	50
STUDENT4	20	40	35	48	45

Any key to return to Quincy...

ပုံ (၇. ၁၅)

## Solving Simultaneous Equations

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

```
//          8x + 2y + 3z = 30  
//          x - 9y + 2z = 1  
//          2x + 3y + 6z = 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] = {
    { 8, 2, 3 },
    { 1, -9, 2 },
    { 2, 3, 6 },
};

// Right-hand side vector
double r[ ] = { 30, 1, 31 };

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

double temp;

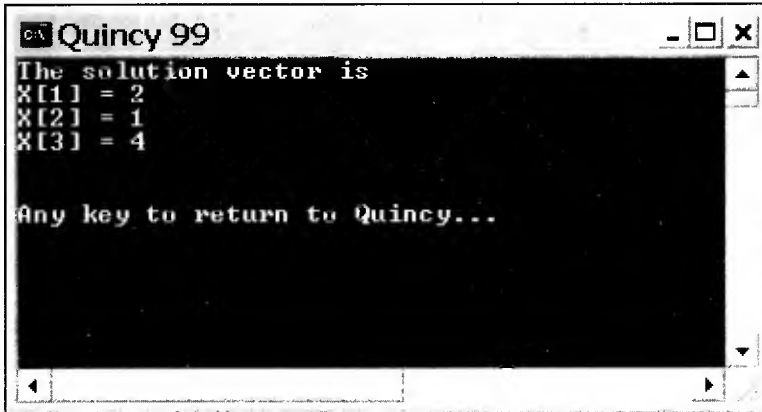
do {
    m = 0;
    for ( i=0; i<n; ++i )
    {
        *temp = r[i];
        for ( j=0; j<n; ++j )
            if ( i != j )
                temp -= x[j]* c[i][j];
        temp /= c[i][i];
        if (fabs(temp - x[i]) > 1e-7 ) ++m;
        x[i] = temp;
    }
} while (m != 0);

// Print output result
cout << "The solution vector is\n";
for ( i=0; i<n; ++i )
    cout << "X[" << i+1 << "] = "
        << x[i] << endl;
cout << endl;
return 0;

}

```

- Ex7010.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၁၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၇. ၁၆)

## 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;
```



```

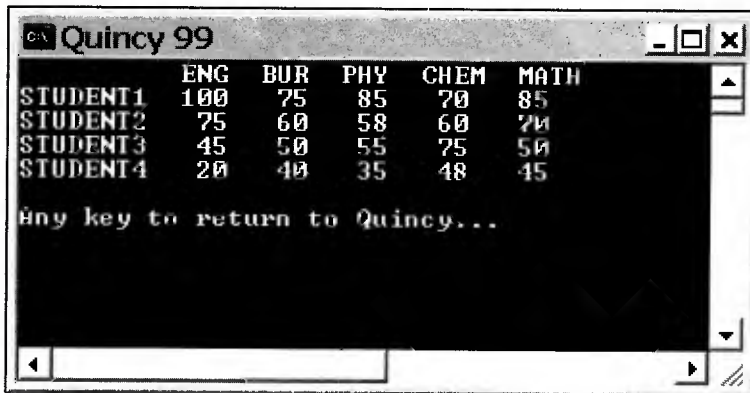
        for (int j =0; j < SUBJECTS; j++)
            cout << setw(5) << marks[i][j];
    }

}

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 လိုက်မယ်ဆိုရင် ပုံ (၇. ၁၇) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၇. ၁၇)

# 6 Arrays of Objects

မှီရိုး data type တွေကို array ထဲမှာ multiple data တွေအနေနဲ့ store လုပ်လို့ရသလို object တွေကိုလည်း array ထဲမှာ သိမ်းလို့ရပါတယ်။ အောက်မှာဖော်ပြထားတဲ့ Listing 7.12 ဟာ array ကိုအသုံးပြုပြီး object (15) ခုအတွက် Celsius ကနေ Fahrenheit ကို temperature conversion လုပ်ပေးထားတာပါ။

// Listing 7.12: Arrays of objects

```
# include <iostream>
const int N=15;

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

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

int main( )
{
    temprs 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(i);
    cout.width(11);
    cout << obj[i].getFah( );
    cout << endl;
}
return 0;
}

```

၂။ Ex7012.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်းမှာ temprs class object တွေဖြစ်တဲ့ array object (N) ခုကို construct လုပ်ပါတယ်။ ပြီးတော့ရင် for loop ထဲဝင်ပြီး obj[0] အတွက် celsius တန်ဖိုးကို setCel( ) call ခေါ်ပြီး set လုပ်ပေးပါတယ်။ ဒါဆိုရင်  $n=0$  အနေနဲ့ဝင်သွားမှာပါ။ ဒီတော့  $cel = -40 + n * 10 = -40 + 0 * 10 = -40$  လို့တွက်ချက်ပြီး main( ) function ကိုပြန်ရောက်လာမှာပါ။ obj[0].setCel(0) = -40 ဖြစ်ပြီပေါ့။
- obj [0] ရဲ့ celsius တန်ဖိုးကို fahrenheit ပြောင်းပေးဖို့အတွက် getFah ( ) function ကို call ခေါ်ပါတယ်။ ဒါဆိုရင်  $fah = 1.8 * cel + 32 = 1.8 * (-40) + 32 = -40$  ကိုတွက်ယူပါတယ်။ ပြီးရင် main ( ) function ဆီကို obj[0].getFah ( ) = -40 လို့ return ပြန်ရောက်လာပါတယ်။ for loop ကို (15) ကြိမ်တိတိပတ်ပြီးရင် Celsius vs Fahrenheit table ယောတစ်ခုဟာ ကွန်ပျူတာမှာပေါ်လာမှာဖြစ်ပါတယ်။ Ex7012.cpp program ကို run ကြည့်ပါ။ ပုံ (၇. ၁၈) မှာပြထားတဲ့အတိုင်းမြင်ရပါလိမ့်မယ်။

## More on Arrays of Objects

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

Quincy 99	
CELSIUS	FAHRENHEIT
-40	-40.00
-30	-22.00
-20	-4.00
-10	14.00
0	32.00
10	50.00
20	68.00
30	86.00
40	104.00
50	122.00
60	140.00
70	158.00
80	176.00
90	194.00
100	212.00
Any key to return to Quincy...	

0 (2.00)

```
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 << "\'"; }
```

```
};
```

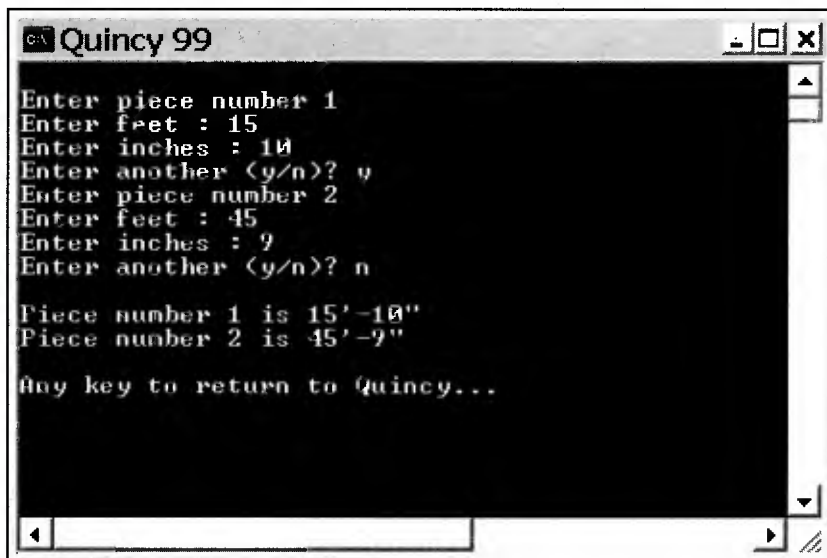
```

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 လိုက်မယ်ဆိုရင် ပုံ (၇. ၁၉) မှာပြထားတဲ့အတိုင်းမြင်ရပါလိမ့်မယ်။



```

Quincy 99
Enter piece number 1
Enter feet : 15
Enter inches : 10
Enter another (y/n)? y
Enter piece number 2
Enter feet : 45
Enter inches : 9
Enter another (y/n)? n

Piece number 1 is 15'-10"
Piece number 2 is 45'-9"

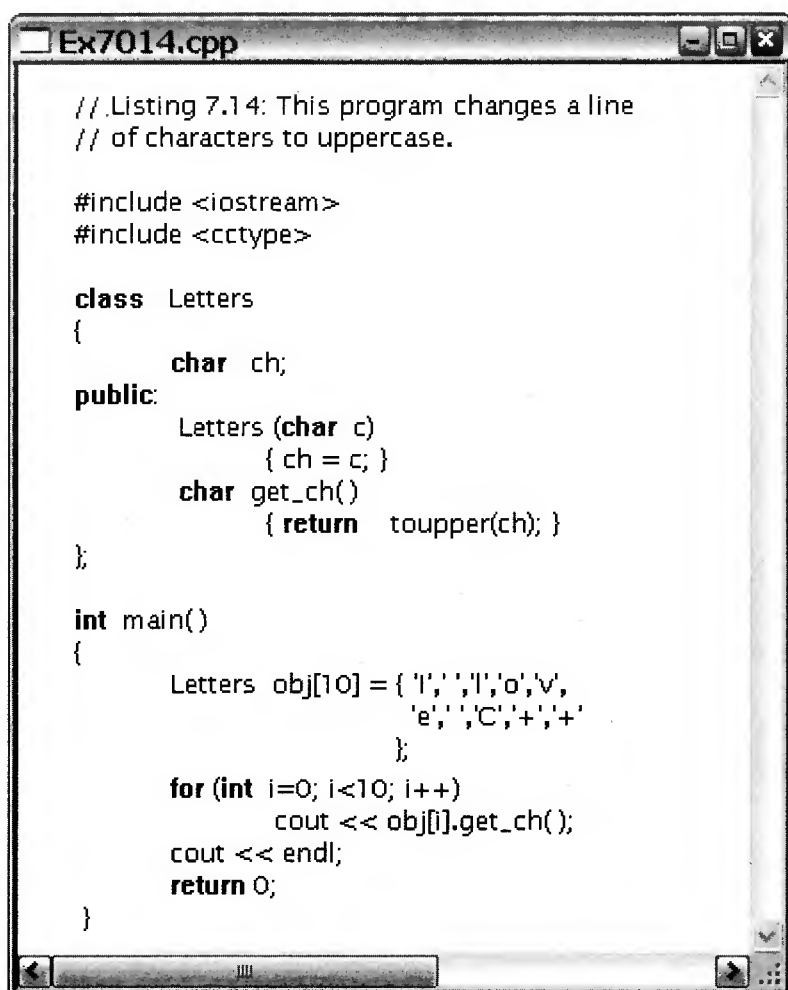
Any key to return to Quincy...

```

ပုံ (၇. ၁၉)

# Initializing Arrays of Objects

ပုံ (၇. ၂၀) မှာရေးထားတဲ့ Ex7014.cpp program ဟာ type class letters ဖြစ်တဲ့ array object တွေမှာ initialize လုပ်ထားတဲ့ character တွေကို uppercase letter ဖြစ်အောင်ပြောင်းပေးတဲ့ program ဖြစ်ပါတယ်။ object obj[ ] မှာ I love C++ ဆိုတဲ့ character (10) ခုကို ch နဲ့ initialize လုပ်ပေးထားပါတယ်။ ကြောင့် obj[i].get\_ch( ) ကို display လုပ်ခိုင်းတာနဲ့ array object တစ်ခုချင်းထဲကသက်ဆိုင်ရာ ch တစ်ခုစီကို uppercase ပြောင်းပြီးပြပေးမှာပါ။ toupper( ) function ကိုအသုံးပြုလို့ရအောင် <cctype> header နဲ့ကို program မှာ include လုပ်ပေးရပါမယ်။



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

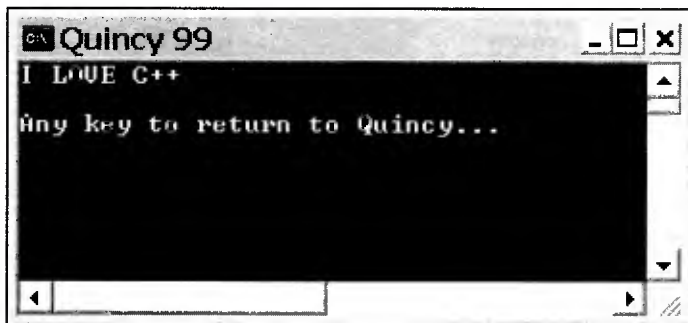
#include <iostream>
#include <cctype>

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

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

ပုံ (၇. ၂၀)

၂။ Ex7014.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၂၁) မှာပြထားတဲ့အတိုင်းမြင်ရပါလိမ့်မယ်။



ပုံ (၇. ၂၁)

## Initializing Multidimensional Arrays of Objects

၁။ Ex7015.cpp program မှာ two-dimensional array object obj[ ] ထဲမှာ initialize လုပ်ထားတဲ့ element တစ်ခုချင်းကိုနှစ်ထပ်ကိန်းတင်ပေးတဲ့ program ပါပဲ။ getSquare( ) function ကိုအသုံးပြုပြီး obj[0][0] နဲ့ obj[0][1] ထဲက {1, 5} element (2) ခုကိုနှစ်ထပ်ကိန်းတင်ပြီး display လုပ်ပြပါလိမ့်မယ်။ for loop ကို (၄) ကြိမ်ပတ်ခိုင်းထားတဲ့အတွက် ကျန်တဲ့ object (8) ခုက element တွေကိုလည်း square လုပ်ပေးမှာပါ ၊ လေ့လာကြည့်ပါ။

// 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, 6.8},
                        {3, 7.4},
                        {4, 8.1}
};
```

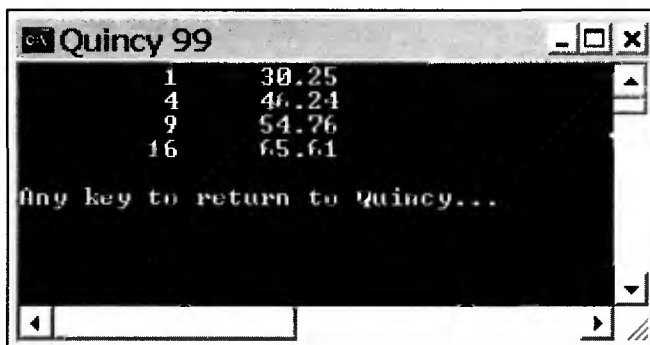
**for** (int i=0; i<ROW; i++)

```
cout << setw(10) << obj[i][0].getSquare( )
      << setw(10) << obj[i][1].getSquare( )
      << endl;
```

**return** 0;

}

Ex7015.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၂၂) မှာပြထားတဲ့အတိုင်းမြင်ရပါလိမ့်မယ်။



ပုံ (၇. ၂၂)



# 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;
}

};

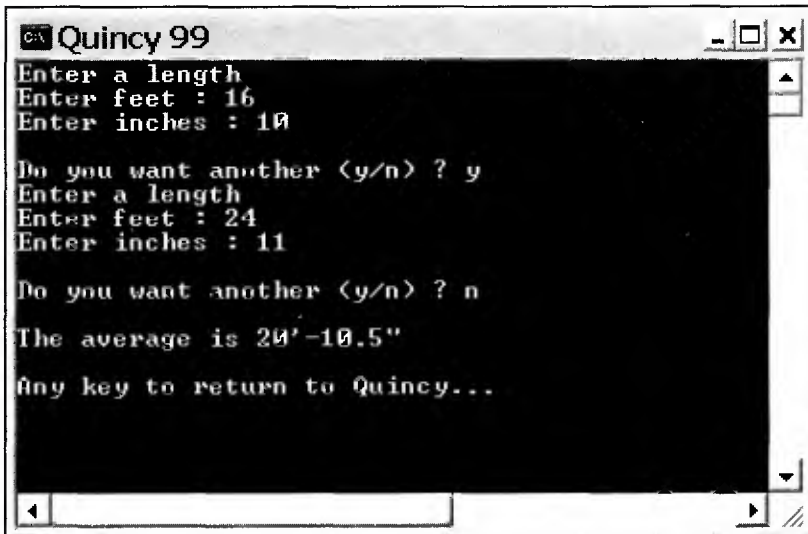
int main( )
{
    Length obj[MAX];
    Length total, ans;
    int count=0;
    char ch;

    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 ဟာ type class Length ဖြစ်တဲ့ array object (50) ခုထဲမှာ store လုပ်ထားတဲ့ ft-inch data အစုံတွေကိုပေါင်းပြီး average ရှာပေးမယ့် program ဖြစ်ပါတယ်။ 50 ဟာ maximum array element အရေအတွက်ပါ။ ကျွန်တော်တို့က object တစ်ခုအတွက် data တစ်စုံစီထည့်ပေးသွားရမှာဖြစ်ပါတယ်။ do-while loop ကိုကြည့်ပါ။ feet-inches data အစုံအရေ အတွက်ကို count ထဲမှာမှတ်သွားပါတယ်။ data ထည့်ဦးမယ်ဆိုရင် Do you want another (y/n) ? prompt ပေါ်လာတဲ့အခါမှာ y ကို ချိတ်ထည့်ပါ။ ရပ်ချင်ရင် n ကိုချိတ်ထည့်ရပါမယ်။ n ချိတ်ထည့်ပြီးတာနဲ့ average ရှာထားတဲ့အဖြေကို ကွန်ပျူတာက display လုပ်ပြမှာပါ။
- ဒီ program မှာ Length constructor function ကို သလနမူသမိ လုပ်ပြထားပါတယ်။ Ex7016.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၇. ၂၃) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



```

Quincy 99
Enter a length
Enter feet : 16
Enter inches : 10

Do you want another (y/n) ? y
Enter a length
Enter feet : 24
Enter inches : 11

Do you want another (y/n) ? n
The average is 24'-10.5"
Any key to return to Quincy...
```

ပုံ (၇. ၂၃)

# Chapter 8



operator overloading ဆိုတာ C++ ရဲ့ object-oriented programming (OOP) မှာ စိတ်ဝင်စားဖွယ် အစိတ်အပိုင်းတစ်ခုလို့ပြောရင် မမှားပါဘူး။ ဘာပြုလို့လဲဆိုတော့ ခက်ခဲပြီးအမြင်မှာရှုပ်ထွေးတဲ့ C++ program statement မျိုးတွေကို operator overloading နည်းအသုံးပြုပြီး လွယ်ကူရှင်းလင်းအောင်လုပ်လို့ရတာကိုး။ ဥပမာ integer သို့မဟုတ် floating-point variable (2) ခုကိုပေါင်းချင်ရင် C++ ရဲ့ ပုံမှန် operator တစ်ခုဖြစ်တဲ့ '+' operator ကို အသုံးပြုပြီး  $c = a + b$ ; လို့ရေးလို့ရပေမယ့်  $a$ ၊  $b$ ၊  $c$  တို့ဟာ class type object တွေ ဖြစ်နေမယ်ဆိုလို့ရှိရင် စောစောကလိုလွယ်လွယ်လေး တန်းပေါင်းလို့မရတော့ပါဘူး။ အခုလိုမျိုးရေးမှရမှာပါ။

`c.addObjs (a,b);`

ဒီရေးနည်းမျိုးကို စာဖတ်သူလည်းလေ့လာခဲ့ပြီးပါပြီ။ တစ်ကယ်လို့ '+' operator တစ်ခုဟာ ရိုးရိုး data type တွေကိုပေါင်းလို့ရသလို object တွေကိုလည်း အဲဒီပုံစံနဲ့ပဲ ပေါင်းလို့ရမယ်ဆိုရင် အစွမ်းတစ်ခုတိုးလာရမှာပါပဲ။ အဲဒါဆိုမကောင်းဘူးလား။  $c = a + b$ ; လို့တန်းရေးချနိုင်ပြီပေါ့။ တစ်ကယ်တော့ '+' operator ကို overload လုပ်ခြင်းအားဖြင့် သူ့ရဲ့ပင်ကိုယ်ဂုဏ်သတ္တိပျောက်မသွားပါဘူး။ define လုပ်လိုက်တဲ့ meaning အသစ်တောင်တိုးလာပြီပဲ။ operator overloading ဟာ C++ language ကိုပိုမိုကျယ်ပြန့်အောင် ပံ့ပိုးပေးမှာဖြစ်ပါတယ်။

## ၈.၁ Overloaded ++ Operator

၁။ ကောင်းပြီ ၊ operator တစ်ခုကို overload လုပ်ချင်ရင် ဘာအရင်လုပ်ရမလဲ။ operator function တစ်ခုကို create လုပ်ရမှာပါ။ operator overloading ဆိုတာ function overloading နဲ့ဆင်တူပါတယ်။ အောက်မှာရေးပြထားတဲ့ Ex801.cpp program မှာ counter ဆိုတဲ့ class တစ်ခုကို create လုပ်ပြီးတော့ objects x နဲ့ y တို့က variable count ကိုလည်း increment လုပ်ပြသွားပါတယ်။ member function ကို အခုလို call ခေါ်ပြထားတယ်လေ။

```
x.incrCount ( );  
y.incrCount ( );
```

တစ်ကယ်လို့ ဒီ statement (2) ခုကို အခုလို x++; y++; လို့ ပြင်ရေးလို့ရရင် ပိုမကောင်းဘူးလား။ ဒါဟာ '+' operator ကို overload လုပ်လိုက်တာပါ။ ဆိုလိုတာက compiler အနေနဲ့ ++ operator ကို တွေ့တိုင်း သူနဲ့သက်ဆိုင်ရာ member function ကို call ခေါ်ပြီး operand ကို increment လုပ်ပေးမှာပါ။ တစ်ခုပဲရှိတယ်၊ operand ဖြစ်တဲ့ x နဲ့ y တို့ဟာ 'counter' class type ဖြစ်ဖို့လိုပါတယ်။ x နဲ့ y တို့ဟာ integer variable တွေ ဖြစ်နေမယ်ဆိုရင် compiler က C++ built-in '+' operator ကို အသုံးပြုပြီး increment လုပ်ပေးမှာပါ။ '+' operator (2) ခုကို compiler ကခွဲခြားသိပါတယ်။ ကောင်းပြီ ၊ Ex801.cpp program ကိုအရင်လေ့လာကြည့်ပါ။

```
// Listing 8.1: Incrementing a counter variable  
#include <iostream>
```

```
class counter  
{  
    unsigned int count;  
public:  
    counter( )  
        { count = 100; }  
  
    int getCount( )  
        { return count; }  
  
    void incrCount( )
```

```

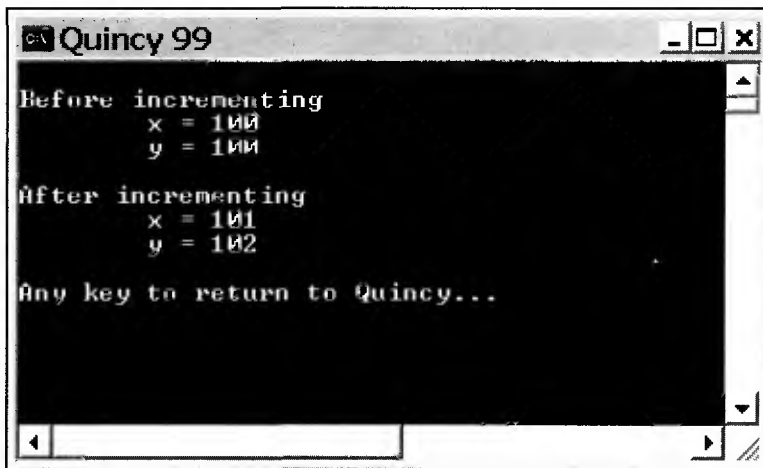
        { count++; }
};

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 လိုက်မယ်ဆိုရင် ပုံ (စ. ၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



```

Quincy 99
Before incrementing
  x = 100
  y = 100

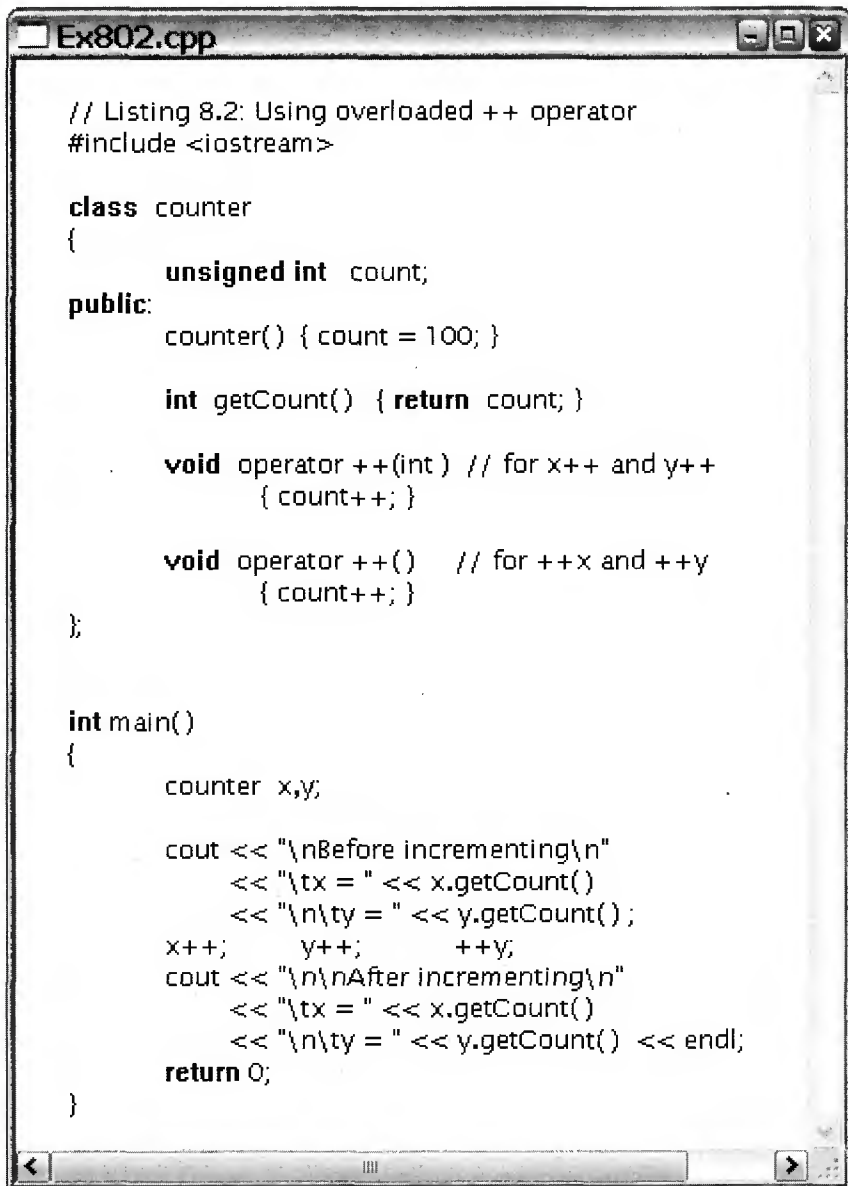
After incrementing
  x = 101
  y = 102

Any key to return to Quincy...

```

ပုံ (စ. ၁)

၃။ overloaded ++ operator ကိုအသုံးပြုပြီး Ex801.cpp program ကိုအခုလိုပြင်ရေးလိုရပါတယ်။  
ပုံ (၈.၂) မှာပြထားတဲ့ Ex802.cpp program ကိုလေ့လာကြည့်ပါ။



```
// 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 incrementing\n"
         << "tx = " << x.getCount()
         << "\nty = " << y.getCount();
    x++;    y++;    ++y;
    cout << "\nAfter incrementing\n"
         << "tx = " << x.getCount()
         << "\nty = " << y.getCount() << endl;
    return 0;
}
```

ပုံ (၈.၂)

Ex802.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- `main( )` function ထဲမှာ `counter class object (2)` ခုဖြစ်တဲ့ `x` နဲ့ `y` တို့ကို `define` လုပ်လိုက်တာနဲ့ `counter( )` constructor function ကအလုပ်စလုပ်ပါပြီ။ `x.count=100` နဲ့ `y.count=100` လို့ initialize လုပ်ပေးမှာပါ။ ဒီတော့ `x.getCount( )` နဲ့ `y.getCount( )` ကို call ခေါ်လိုက်တာနဲ့ `object x` နဲ့ `y` တို့ထဲက `count` တန်ဖိုးအသီးသီးကို return ပြန်ပေးပါလိမ့်မယ်။ `getCount( )` function က `count` တန်ဖိုးတွေကို သိပြီးသားဖြစ်နေတာကိုး။
- အခုတစ်ခါ `object x` နဲ့ `y` ကို `'++'` operator နဲ့ overload လုပ်လိုက်တဲ့အတွက် `x` နဲ့ `y` ထဲက `count data` တန်ဖိုးကို increment လုပ်တော့မှာပါ။ ဒီဥစ္စာဟာ `void operator ++(int )` ဆိုတဲ့ function member ကို call ခေါ်လိုက်ပြီး `x++` နဲ့ `y++` ကိုဖြေရှင်းပေးတာပါ။
- `++y` လို့ operator overloading လုပ်မယ်ဆိုရင် `void operator ++( )` ဆိုတဲ့ function member ကို call ခေါ်လိုက်ပြီး `++y` ကိုဖြေရှင်းပေးပါလိမ့်မယ်။ Ex802.cpp program ကို run မယ်ဆိုရင် ပုံ (၈. ၁) မှာပြထားတဲ့အဖြစ်ပဲရမှာဖြစ်ပါတယ်။

## ၈.၂ Overloaded + Operator

၁။ အခုတစ်ခါတင်ပြမယ့် Ex803.cpp program မှာ overloaded + operator အသုံးပြုနည်းကို ရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// 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;

```

```

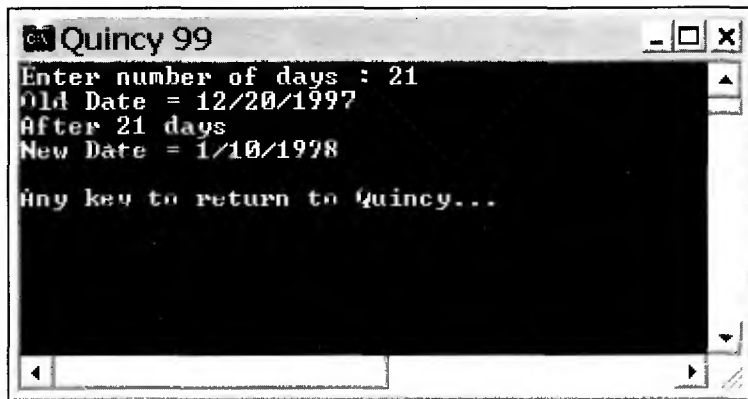
newDate = oldDate + x;           // three weeks hence
cout << "Old Date = " ;
oldDate.display( );
cout << "After " << x << " days\n";
cout << "New Date = " ;
newDate.display();

return 0;
}

```

### Ex803.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်း main( ) function မှာ Date object (2) ခုဖြစ်တဲ့ oldDate နဲ့ newDate တို့ကို construct လုပ်ပါတယ်။ oldDate အတွက် oldDate.month = 12 ၊ oldDate.day = 20 နဲ့ oldDate.year = 1997 တို့ကို assign လုပ်ပေးမှာပါ။
- Enter number of days : ဆိုတဲ့ prompt ပေါ်လာတဲ့အခါမှာ x = 21 ကိုရိုက်ထည့်ပါမယ်။ newDate = oldDate + x; ဆိုတဲ့ statement ရဲ့ဆိုလိုရင်းက newDate = oldDate.operator + (21); နဲ့အတူတူပါပဲ။ ဒီတော့ overloaded + operator function ထဲကို n = 21 အနေနဲ့ pass လုပ်ဝင်သွားပါပြီ။
- overloaded + operator function ထဲရောက်တဲ့အခါကျရင် 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] သို့မဟုတ် while (41 > 31) ကိုဆိုလိုပါတယ်။ ဒီတော့ while loop ထဲဝင်လာပြီး n -= days[x.month-1] = 41 - 31 = 10 ကိုတွက်ယူပါတယ်။
- ပြီးတော့ရင် if (++x.month == 13) လို့မေးပါတယ်။ ဒီအဓိပ္ပါယ်က if (12+1 == 13) နဲ့အတူတူပါပဲ။ ဒီတော့ if block statement ထဲဝင်လာပြီး x.month = 1 နဲ့ x.year++ = 1997 + 1 = 1998 လို့တွက်ယူပါတယ်။
- နောက်တစ်ကြိမ်မှာတော့ while loop ထဲကိုမဝင်တော့ပါဘူး။ x.day = n = 10 လို့ assign လုပ်ပေးပြီး Date object x ကို main( ) function ဆီကို return ပြန်ပို့ပေးပါတယ်။ ဒီတော့ main( ) ကိုပြန်ရောက်လာတဲ့အခါမှာ newDate = {1,10,1998} ဖြစ်နေပါပြီ။ Ex803.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈. ၃) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါပဲ။



ပုံ (၈.၃)

## More on Overloaded + and ++ Operators

၁။ Ex804.cpp program ဟာဆိုရင် overloaded +/++ operator တွေအားလုံးကို program တစ်ခုတည်းမှာပဲ စုပြီးအသုံးပြုပြထားတာဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။ program ကိုစာဖတ်သူကိုယ်တိုင် trace လုပ်ကြည့်စေချင်ပါတယ်။ ပိုနားလည်သွားတာပေါ့။

// Listing 8.4: More on overloaded + and ++ 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 << 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};
```

```

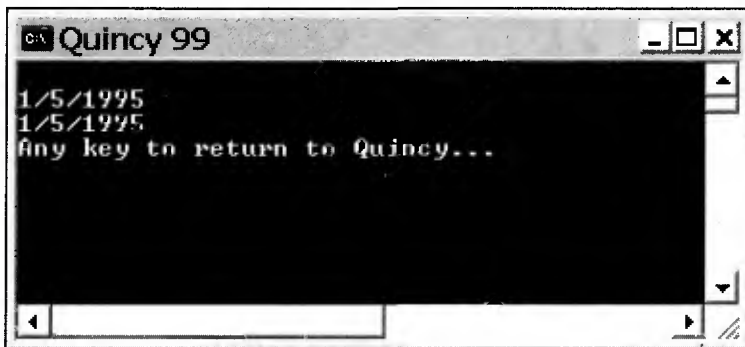
int main( )
{
    Date  newDate, oldDate(1,4,1995);

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

    return 0;
}

```

၂။ Ex804.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈. ၄) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈. ၄)

## Overloading Arithmetic + and - Operators

၁။ Ex805.cpp program မှာ Length object တွေဖြစ်တဲ့ L1 နဲ့ L2 တို့ကို define အရင်လုပ်ပြီး Length object L3 နဲ့ L4 တို့ကို overloaded +/- operator တွေနဲ့ အပေါင်းအနှုတ်လုပ်ထားတာဖြစ်ပါတယ်။ လေ့လာကြည့်ပါ။ program ကိုစာဖတ်သူကိုယ်တိုင် trace လုပ်ကြည့်စေချင်ပါတယ်။

```
// 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;

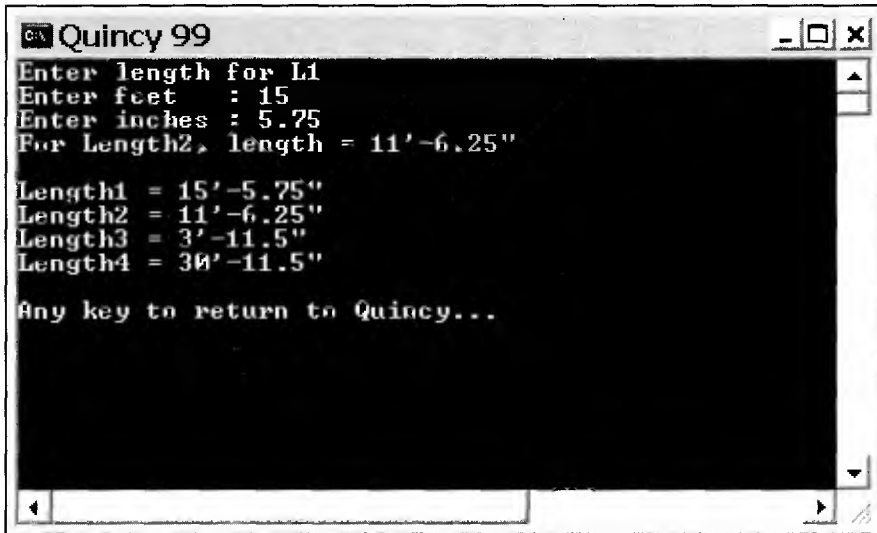
    cout << "Enter length for L1\n";
    L1.getLength( );

    Length    L2(11,6.25);
    cout << "For Length2, 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 လိုက်မယ်ဆိုရင် ပုံ (၈. ၅) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈. ၅)

## Adding Vectors

၁။ vector sum သို့မဟုတ် vector subtract လုပ်တာကိုမပြောခင် ကျွန်တော်အရင်ပြောချင်တာက အမှတ်တစ်ခုကို 2D မျက်နှာပြင်ပေါ်မှာ နေရာသတ်မှတ်ပုံသတ်မှတ်နည်းကို သိထားဖို့လိုပါတယ်။ 2D plane တစ်ခုပေါ်က အမှတ်တစ်ခုကို (x,y) rectangle coordinates နဲ့ဖော်ပြလို့ရသလို polar coordinates (radius, angle) နဲ့လည်းဖော်ပြလို့ရပါတယ်။ ကောင်းပြီ ၊ စတင်ချင်း vector အပေါင်းအနှုတ်လုပ်တာကို rectangular coordinates အသုံးပြုပြီး program ရေးကြည့်ရအောင်။ vector တစ်ခုက (x=2, y=3) နဲ့နောက် vector တစ်ခုက (x=5, y=2) ဖြစ်တယ်ဆိုပါစို့။ ဒီ vector (2) ခုကိုပေါင်းကြည့်မယ်လေ။

// Listing 8.6: Adding vectors

```
#include <iostream>
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(2,3), v2(5,2), vr;
```

```
int x, y;
```

```
cout << "First vector for v1 is\n\t(2, 3)\n\n";
```

```
cout << "Second vector for v2 is\n\t(5, 2)\n\n";
```

```
vr = v1 + v2;
```

```
vr.getXy (x, y);
```

```
cout << "Thus New vector for v1 + v2 is\n"
```

```
<< "\t(" << x << ", " << y << ")\n\n";
```

```
vr = v2 - v1;
```

```
vr.getXy (x, y);
```

```
cout << "New vector for v2 - v1 is\n"
```

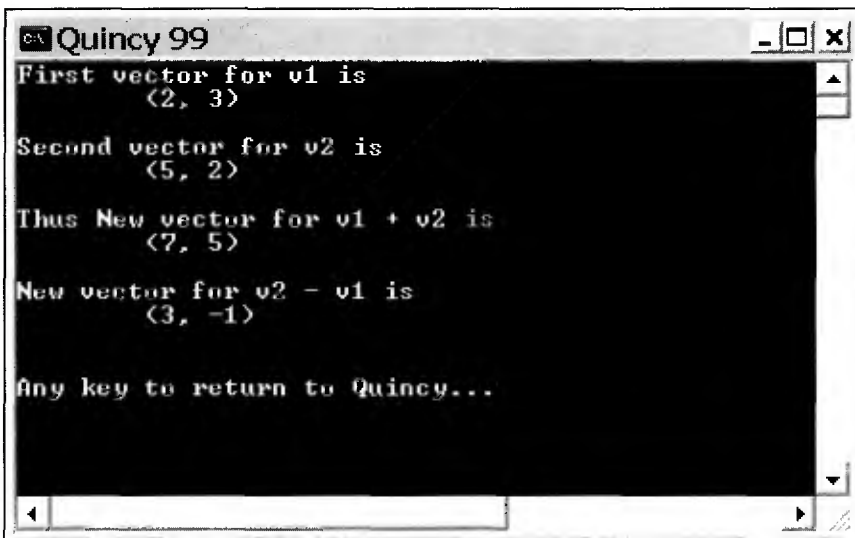
```
<< "\t(" << x << ", " << y << ")\n\n";
```

```
return 0;
```

```
}
```

## Ex806.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်း main( ) function ထဲမှာ class coord object တွေဖြစ်တဲ့ v1 ၊ v2 နဲ့ vr တို့ကို define လုပ်ပါတယ်။ ဒီတော့ coord constructor တွေထဲရောက်သွားပြီး (v1.xco = i = 2 ၊ v1.yco = j = 3) ၊ (v2.xco = i = 5 ၊ v2.yco = j = 2) နဲ့ (vr.xco = 0 ၊ vr.yco = 0) လို့ initialize လုပ်ပေးပါတယ်။ vr = v1 + v2 ဆိုတာ vr = v1.operator+(v2) ဖြစ်ပါတယ်။ vr = v2 - v1 ဆိုတာ vr = v2.operator-(v1) ဖြစ်ပါတယ်။
- Ex806.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈. ၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



```
Quincy 99
First vector for v1 is
    (2, 3)
Second vector for v2 is
    (5, 2)
Thus New vector for v1 + v2 is
    (7, 5)
New vector for v2 - v1 is
    (3, -1)
Any key to return to Quincy...
```

ပုံ (၈. ၆)

## Overloaded + Operators with Polar Coordinates

၁။ တစ်ကယ်လို့ rectangular coordinates အစား polar coordinates ကို အသုံးပြုချင်တယ်ဆိုရင် အခုလိုပုံသေနည်းတွေကိုအသုံးပြုရပါမယ်။

$$x = \text{radius} * \cos(\text{angle})$$

$$y = \text{radius} * \sin(\text{angle}) \text{ where angle is in radians}$$

$$\text{radius} = \sqrt{x^2 + y^2}$$

$$\text{angle} = \text{atan}(y / x)$$

// 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 = atan(y/x);
    return polar(r, a);
}

};

int main( )
{
    polar p1(5,0);
    polar p2(5,1.57096325);
    polar p3;

    p3 = p1 + p2;
    cout << "The given two polar vectors are\n";
    cout << "\tP1 = ";
    p1.display( );
    cout << "\n\tP2 = ";
    p2.display( );
    cout << endl;
    cout << "\n\nSum of the two polar vectors is\n";
    cout << "\tP3 = ";
    p3.display( );
    cout << endl;
    return 0;
}

```

Ex807.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်း main( ) function အတွင်းမှာ polar class type ဖြစ်တဲ့ object (3) ခုကို define လုပ်ပါတယ်။ ဒါကြောင့်မို့ (p1.radius = r = 5 ၊ p1.angle = a = 0) ၊ (p2.radius = r = 5 ၊ p2.angle = a = p/2) နဲ့ (p3.radius = 0 ၊ p3.angle = 0) လို့

initialize လုပ်ပေးပါတယ်။ နောက်ပြီး  $p3 = p1 + p2$ ; ဆိုတဲ့ statement ကြောင့် ကွန်ပျူတာက အခုလိုတွက်ယူပါတယ်။

```
x = getx( ) + p.getx( )
    = p1.getx( ) + p2.getx( )
    = p1.radius*cos(p1.angle) + p2.radius*cos(p2.angle)
    = 5*cos(0) + 5*cos( $\pi/2$ )
    = 5 + 0 = 5
```

```
y = gety( ) + p.gety( )
    = p1.gety( ) + p2.gety( )
    = p1.radius*sin(p1.angle) + p2.radius*sin(p2.angle)
    = 5*sin(0) + 5*sin( $\pi/2$ )
    = 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 လိုင်းနံပါတ် `main( )` ကို အဖြေပြန်ပို့ပေးပါတယ်။ ဒါကြောင့်မို့ `object p3` ထဲမှာ `p3.radius = 7.07107` နဲ့ `p3.angle = 0.7854` တို့ဖြစ်သွားပါပြီ။ ထိုနည်းတူ `object p1` ထဲက `radius` ၊ `angle` တန်ဖိုးတွေကို `display` လုပ်နဲ့ အတွက် `p1.display( )` function ကို call ခေါ်ပြီး အခုလိုတွက်ယူပြီးထုတ်ပေးပါတယ်။

`p1.radius = 5`

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

- ဒီနည်းအတိုင်း object p2 ထဲက radius ၊ angle တို့ကိုလည်းတွက်ယူပါတယ်။

`p2.radius = 5`

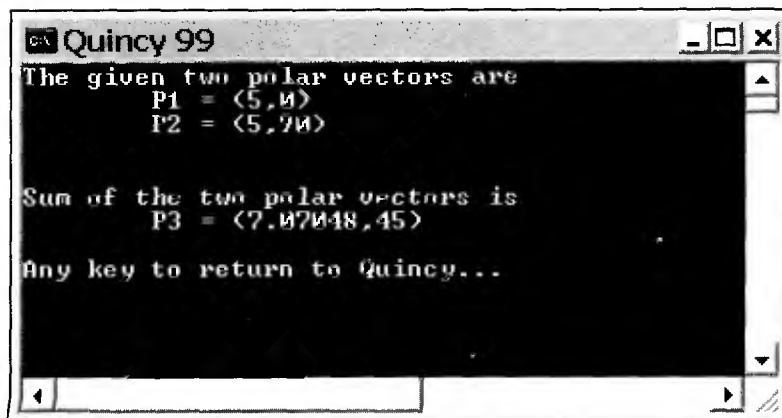
`angle (in degrees) = p2.angle*180/3.141593`  
`= 1.570796325 * 180/3.141593 = 90`

- object p3 ထဲက radius ၊ angle တန်ဖိုးတွေကို `p3.display( )` function ကိုအသုံးပြုပြီး တွက်ယူပါတယ်။ ပြီးတော့ display လုပ်ပြမှာပါ။

`p3.radius = 7.07107`

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

- Ex807.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈.၇) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈.၇)

# 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);
    }
};
```

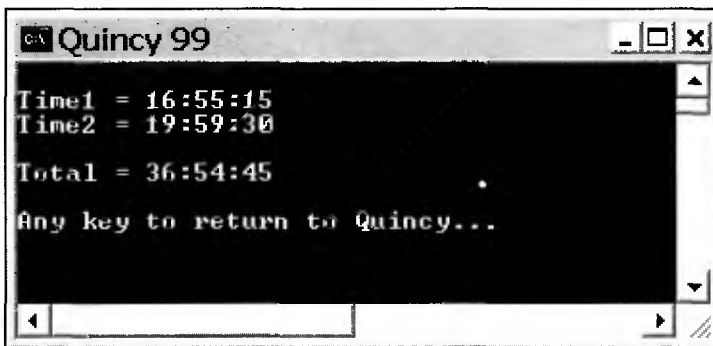
```

int main( )
{
    mytime    t1(16, 55, 15);
    mytime    t2(19, 59, 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 လိုက်မယ်ဆိုရင် ပုံ (၈. ၈) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈. ၈)

## ၈.၃ 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-1];
        if (++x.month == 13)
        {
            x.month = 1;
            x.year++;
        }
    }
    x.day = n;

    return x;
}

```

```

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

```

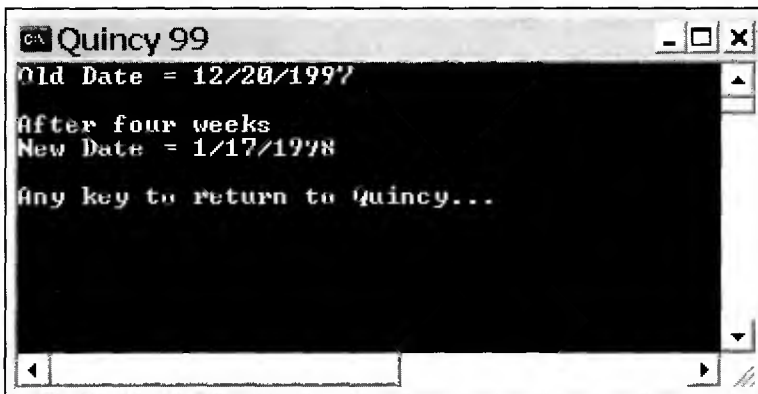
```

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;
}

```

Ex809.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈.၉) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈.၉)

## ၈.၄ Overloading the Assignment += Operator

၁။ ဒီတစ်ခါတင်ပြမှာကတော့ assignment operator တွေဖြစ်တဲ့ += , -= , <=< , >>= , ^= တွေထဲက += operator ကို overload လုပ်ပြထားတဲ့ Ex8010.cpp program ကိုတင်ပြပါမယ်။ လေ့လာကြည့်ပါ။

```
//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};
```

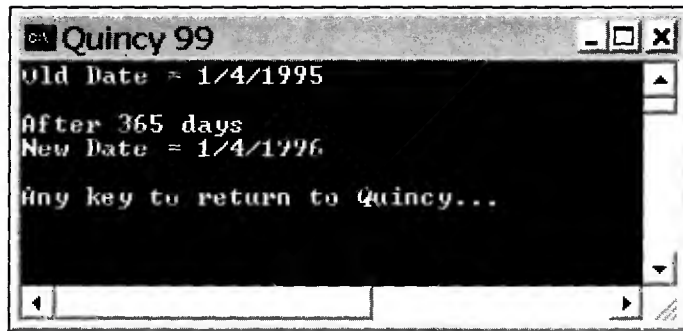
```
// Overloaded + operator definition.
Date Date::operator + (int n) const
{
    Date x = *this;
    n += x.day;
    while (n > days[x.month-1])
    {
        n -= days[x.month-1];
        if (++x.month == 13)
        {
            x.month = 1;
            x.year++;
        }
    }
    x.day = n;
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

Ex8010.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈. ၁၀) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈.၁၀)

## Concatenating Strings with Overloaded += Operators

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex8011.cpp program မှာ object s2 ထဲက string ကို object s1 ထဲက string နဲ့ဆက်ပေးပြီးတော့ s1 အနေနဲ့ assign လုပ်ပါတယ်။ နောက်ပြီး s1 ကို s3 နဲ့ ထပ်ပြီး assign ပေးတဲ့အတွက် s3 ကို display လုပ်တဲ့အခါမှာ ပုံ (၈.၁၁) မှာပြထားတဲ့အတိုင်းအဖြေရပါလိမ့်မယ်။

// Listing 8.11: 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.";
}

};

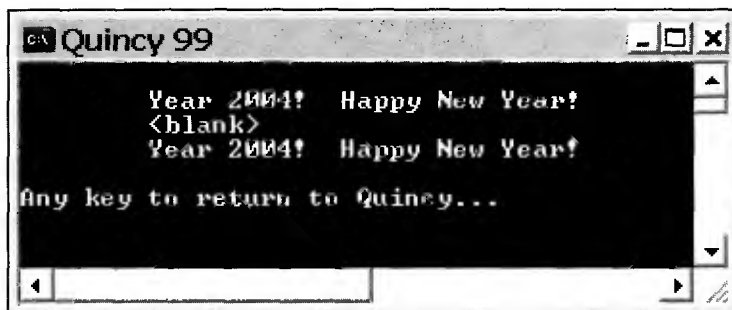
int main( )
{
    myString s1 = "\n\tYear 2004! ";
    myString s2 = "Happy New Year!";
    myString s3 = "\n\t<blank>";

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

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

```

Ex8011.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (စ. ၁၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (စ. ၁၁)

## ၈.၅ Overloaded Relational Operators

Listing 8.12 မှာ မြေပြထားတဲ့ Ex8012.cpp program ဟာ less than '<' operator ကို overload လုပ်ပြီး length class type ဖြစ်တဲ့ အလျားအရှည် (2) ခုကို ဘယ်ဟာကြီးလဲလို့ နှိုင်းယှဉ်ခိုင်းထားပါတယ်။ လေ့လာကြည့်ပါ။

// 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;
}

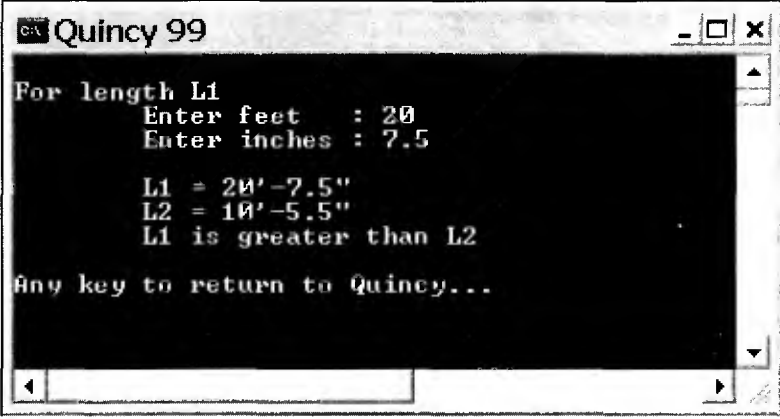
};

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

    Length L2(10,5.5);
    cout << "\n\tL1 = ";                 L1.showLength( );
    cout << "\n\tL2 = ";                 L2.showLength( );
    if (L1 < L2)
        cout << "\n\tL1 is less than L2\n";
    else
        cout << "\n\tL1 is greater than L2\n";
    return 0;
}

```

Ex8012.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈.၁၂) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



```

Quincy 99
For length L1
  Enter feet   : 20
  Enter inches : 7.5

  L1 = 20'-7.5"
  L2 = 10'-5.5"
  L1 is greater than L2

Any key to return to Quincy...

```

ပုံ (၈.၁၂)



# More on Overloaded Relational Operators

// Listing 8.13: More 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;
```

```

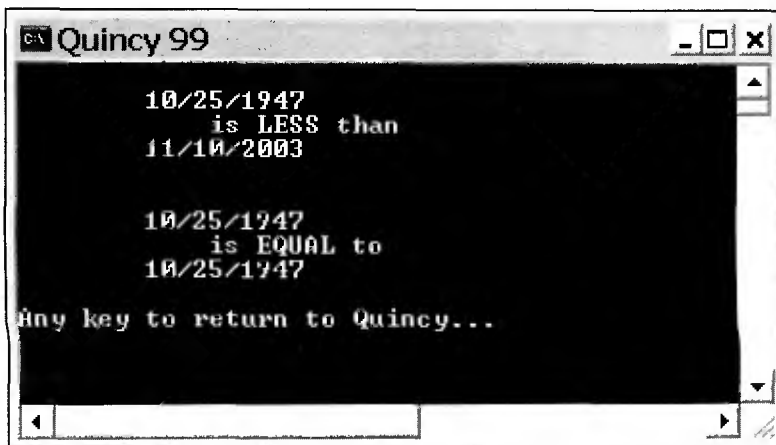
        return this->month < x.month;
    }
    return this->year < x.year;
}

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 လိုက်မယ်ဆိုရင် ပုံ (စ. ၁၃) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (စ. ၁၃)

## ၈.၆ Overloading == Operators

Listing 8.14 မှာ မြေပြထားတဲ့ Ex8014.cpp program ဟာ == operator ကို overloading လုပ်ပြီး myString class type ဖြစ်တဲ့ string (2) ခုကို တူမတူနိုင်ယှဉ်ခိုင်းထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 8.14: 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    getStr( )
```

```
    {
```

```
        cout << '\t';
```

```
        cin.get (str,MAX);
```

```
    }
```

```
    bool    operator == (myString ms)
```

```
    {
```

```
        return    strcmp(str,ms.str) ? FALSE:TRUE;
```

```
    }
```

```
};
```

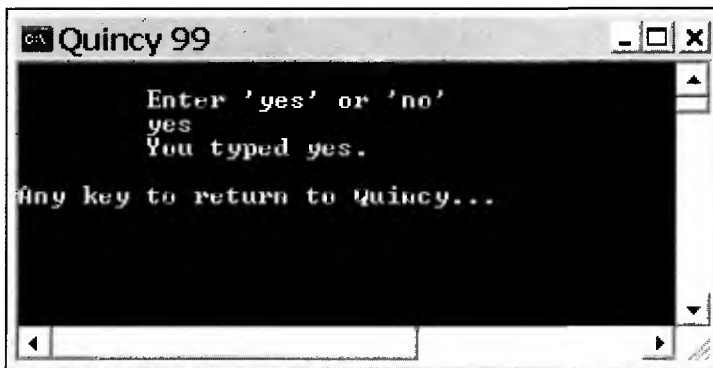
```

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

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

```

Ex8014.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈. ၁၄) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈. ၁၄)

## Overloading - Operators

Listing 8.15 မှာဖော်ပြထားတဲ့ Ex8015.cpp program ဟာဆိုရင် unary minus - operator ကို overload လုပ်ပြီး object (2) ခုရဲ့တန်ဖိုးတွေကို ပြောင်းပြန်ဖြစ်အောင်ပြောင်းပေးပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 8.15: 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); }

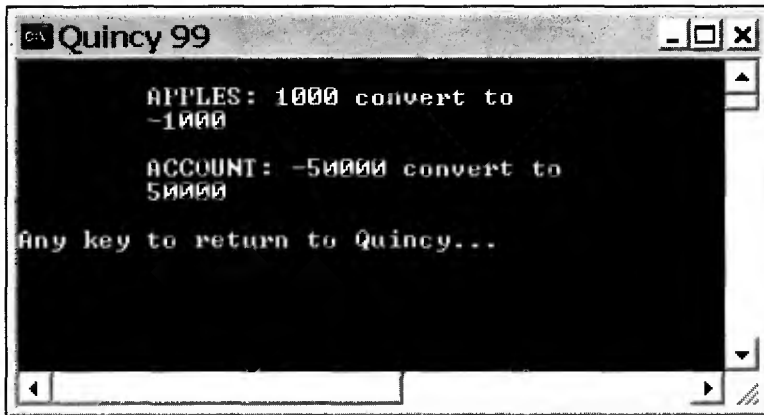
    void display( ) const
        {
            cout << "\n\t" << ch << ": " << x
                << " convert to";
        }

    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 (၈.၁၅)

## ၈.၈ Overloading [ ] Operators

Listing 8.16 မှာ ဖော်ပြထားတဲ့ Ex8016.cpp program ဟာ subscript [ ] operator အသုံးပြုပုံကို ဖော်ပြထားပါတယ်။ ဒီ program မှာ string တစ်ခုကို store လုပ်ထားတဲ့ myStr class ကနေ subscript operator ကို overload လုပ်ပြီး string value ထဲကကြိုက်တဲ့ character position ကို access လုပ်လို့ရတာကို တင်ပြထားပါတယ်။ ဒါပေမယ့် constant string တွေကိုတော့ modify လုပ်လို့မရပါဘူး။

// 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( );
}

```

```

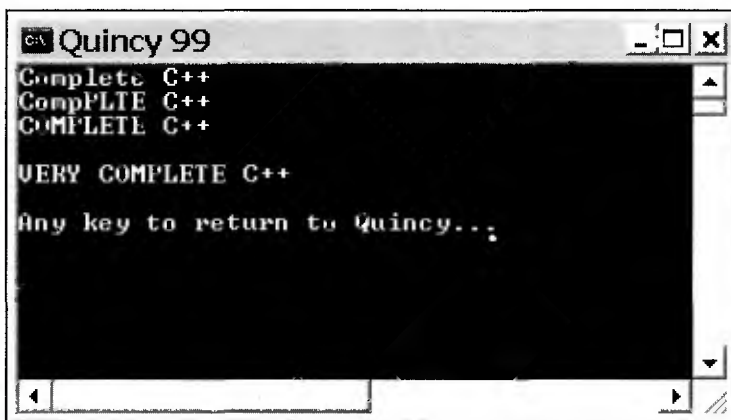
const myStr str2("\nVERY COMPLETE C++");

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

return 0;
}

```

Ex8016.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈. ၁၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၈. ၁၆)

## ၈.၉ Overloading -> Operators

Listing 8.17 မှာဖော်ပြထားတဲ့ Ex8017.cpp program ဟာဆိုရင် pointer-to-member -> operator ကို overload လုပ်နည်းရေးပြထားတာပါပဲ။ DatePtr class object ဟာ pointer တစ်ခုဖြစ်ပြီး သူ့ကို data value assign လုပ်ထား၊ မထားသိပါတယ်။ ကျွန်တော်တို့က date pointer ကို data assign



လုပ်မပေးဘူးဆိုရင် DatePtr constructor function က ဘာ data မှလက်ခံမရတော့ဘူးပေါ့။ ဒီတော့ DatePtr object ဖြစ်တဲ့ dp ဟာ zero value ကို point လုပ်နေပါလိမ့်မယ်။ dp -> display( ); ကနေ call လိုက်တဲ့အခါမှာ ပထမ Date\* operator ->( ) function ထဲဝင်သွားပြီး if (dp == 0) နဲ့တွေ့တဲ့အခါကျရင် null Date object ရဲ့ address (&nulldate) ကို return လုပ်ပေးမှာပါ။ ဒီခါမှာ 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 လို့ initialize လုပ်ပေးပါလိမ့်မယ်။ ဒါဆိုရင် month ၊ day နဲ့ year တို့ရဲ့တန်ဖိုးတွေကို display( ) function မှာ display လုပ်ပြလို့ရပါပြီ။ date object မှာ parameter တွေ assign ထားရင် ဘာဖြစ်မလဲဆိုတာကို စာဖတ်သူကိုယ်တိုင် trace လုပ်ကြည့်စေချင်ပါတယ်။

// 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.
    DatePtr dp;

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

    Date dt(3,15,2003);

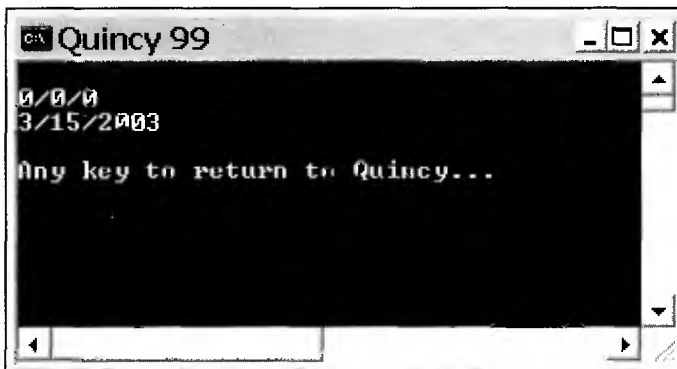
    // Put address of date in pointer.
    dp = &dt;

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

    return 0;
}

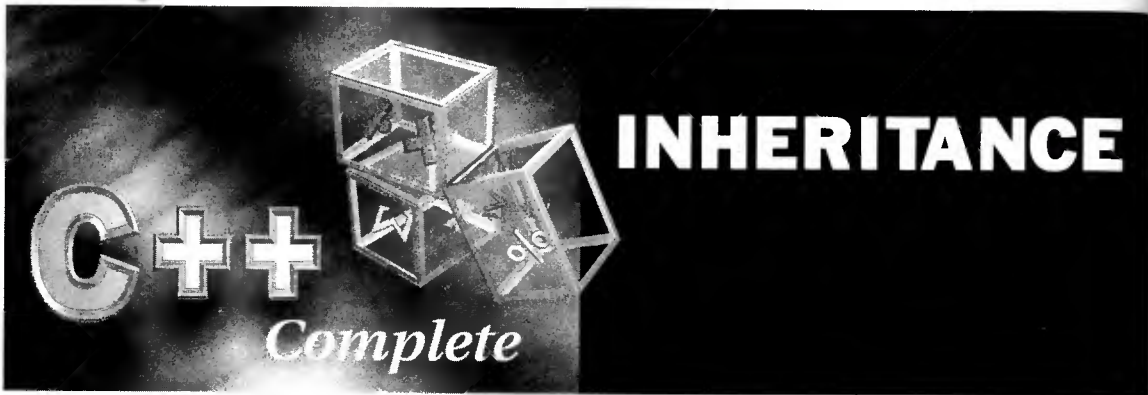
```

Ex8017.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၈. ၁၇) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။

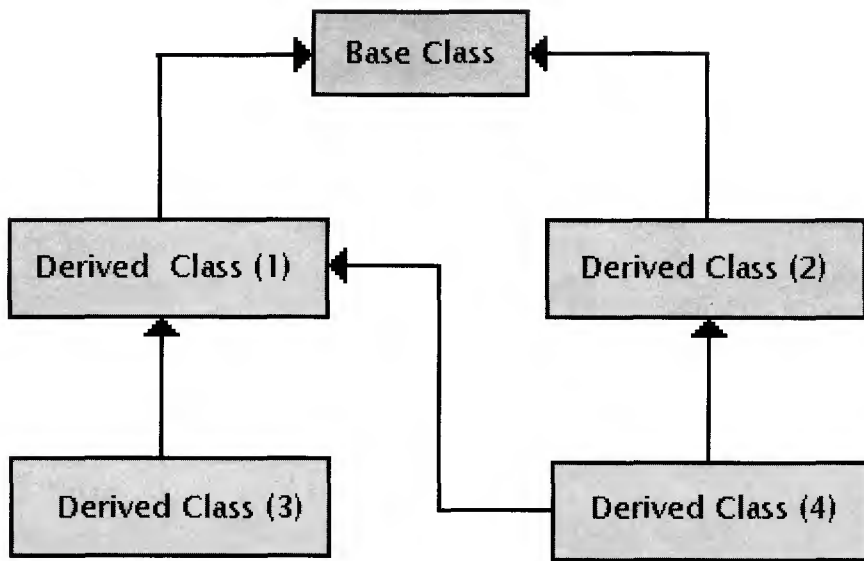


ပုံ (၈. ၁၇)

# Chapter 9



class ပြီးရင် C++ မှာအရေးပါဆုံး feature ကတော့ inheritance ပါပဲ။ inheritance ဟာ ဆိုရင် base class လို့ခေါ်တဲ့ ရေးပြီးသားမှန်ပြီးသား class တွေကနေတစ်ဆင့် derive လုပ်ယူထားတဲ့ မျိုးဆက်ခံပွား class အမျိုးအစားတွေပဲဖြစ်ပါတယ်။ base class ရဲ့အမွေခံဖြစ်တဲ့အတွက် ဒီ class မျိုးတွေကို derived classes လို့ခေါ်ကြပါတယ်။ derived class တစ်ခုရဲ့သဘောတရားက သူများရေးထားပြီးသား class တစ်ခုကိုတိုက်ရိုက် ပြင်တာမျိုးမဟုတ်ပဲ ကိုယ့်အခြေအနေနဲ့ကိုက်ညီမယ့် class ပုံစံအသစ်တစ်ခုကို ပြောင်းယူတဲ့သဘောပေါ့။ အဲဒါကြောင့် derived class တစ်ခုဟာ base class စွမ်းဆောင်နိုင်တာတွေအကုန်လုံးကို လုပ်နိုင်ပါတယ်။ သူ့ကိုပြုပြင်ချင် ရင်လည်း ပြင်လို့ရပါတယ်။ base class ကိုလုံးဝထိစရာမလိုပါဘူး။ သူ့ချည်း လွတ်လွတ်လပ်လပ်ပြင်ဆင်ပြီးတော့ အသုံးပြုလို့ရပါတယ်။ ဒီလိုပြန်လည်အသုံးချခွင့် (reusability) မျိုးကိုရရှိခြင်းဟာ inheritance ရဲ့အားသာချက်ပါပဲ။ ကောင်းပြီ ၊ inheritance ရဲ့သဘောတရားကို အလွယ်တကူ လေ့လာကြည့်လို့ရအောင် ပုံ (၉. ၁) ဖော်ပြထားပါတယ်။ မြားတွေဟာ derived class ကနေ base class ဆီကိုပြောင်းပြန်သွားနေတာ သတိပြုကြည့်ပါ။ ဒါဟာ inheritance ရဲ့သတ်မှတ်ချက်ဖြစ်ပါတယ်။ ပုံ (၉. ၁) ရဲ့ဆိုလိုရင်းကို ခြုံပြောမယ်ဆိုလို့ရှိရင်



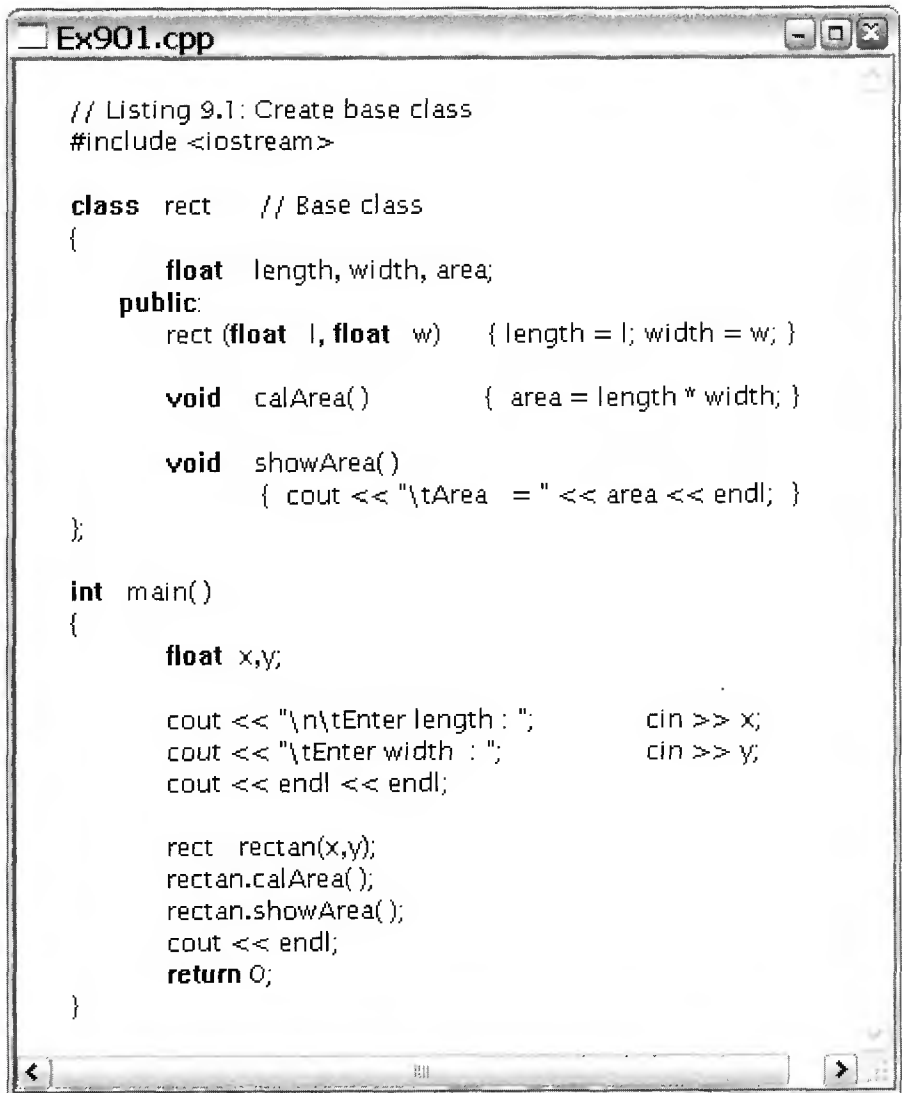
ပုံ (၉. ၁)

- derived class (1) နဲ့ (2) တို့ဟာ base class ကနေ derived ဖြစ်လာတာပါ။ သူတို့ဟာ base class ရဲ့ဂုဏ်သတ္တိအကုန်လုံးကို အလိုအလျောက်ပိုင်ဆိုင်ထားသလို သူတို့တစ်ခုချင်းမှာ အပိုဆောင်း property တစ်မျိုးစီရှိနေကြပါတယ်။ တစ်ခုနဲ့တစ်ခု တူတော့မတူကြပါဘူး။
- derived class (3) ဟာဆိုရင် base class ရော၊ derived class (1) ရော ပိုင်ဆိုင်ထားတဲ့ ဂုဏ်သတ္တိတွေကို အလိုအလျောက် ပိုင်ဆိုင်ထားသလို သူ့မှာလည်း အပိုဆောင်း property တစ်မျိုး ရှိနေတယ်လေ။
- derived class (4) ကတော့ base class ရော၊ derived class (1) နဲ့ (2) ရော ပိုင်ဆိုင် ထားတဲ့ ဂုဏ်သတ္တိတွေအားလုံးကို ပိုင်ဆိုင်ထားသလို သူ့မှာလည်း အပိုဆောင်း property တစ်မျိုး ရှိနေပါတယ်။ အဲဒီလို inheritance မျိုးကို multiple inheritance လို့ခေါ်ပါတယ်။

## ၉.၁ Create a Derived Class

၁။ ကောင်းပြီ ၊ ကျွန်တော်တို့ derived class ကို create မလုပ်ခင် base class တစ်ခုကို အရင် create လုပ်ကြည့်ရအောင်။ ဥပမာ အလျားနဲ့အနံတွေပေးထားတဲ့ rectangle တစ်ခုရဲ့ area ကိုရှာတဲ့ program တစ်ခုပဲ

ဆိုပါစို့။ ပုံ (၉.၂) မှာရေးပြထားတဲ့ Ex901.cpp program ကိုကြည့်ပါ။ ဒီ program နဲ့ပတ်သက်ပြီး အသေးစိတ်ရှင်းမပြောပါဘူး။ program ကို run လိုက်ပြီး Enter length နဲ့ Enter width ဆိုတဲ့ prompt ပေါ်လာတဲ့အခါမှာ 10 နဲ့ 20 ကိုအသီးသီးထည့်ပေးလိုက်တာနဲ့ Area = 200 လို့အဖြေရပါလိမ့်မယ်။ ဒီ program မှာကျွန်တော်တို့စိတ်ဝင်စားတာက class rect ပါပဲ။ ဒီ class ကို base class လုပ်ယူကြည့်ရအောင်။



```
// Listing 9.1: Create base class
#include <iostream>

class rect    // Base class
{
    float  length, width, area;
public:
    rect (float  l, float  w)    { length = l; width = w; }

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

    void  showArea()
        { cout << "\tArea  = " << area << endl; }
};

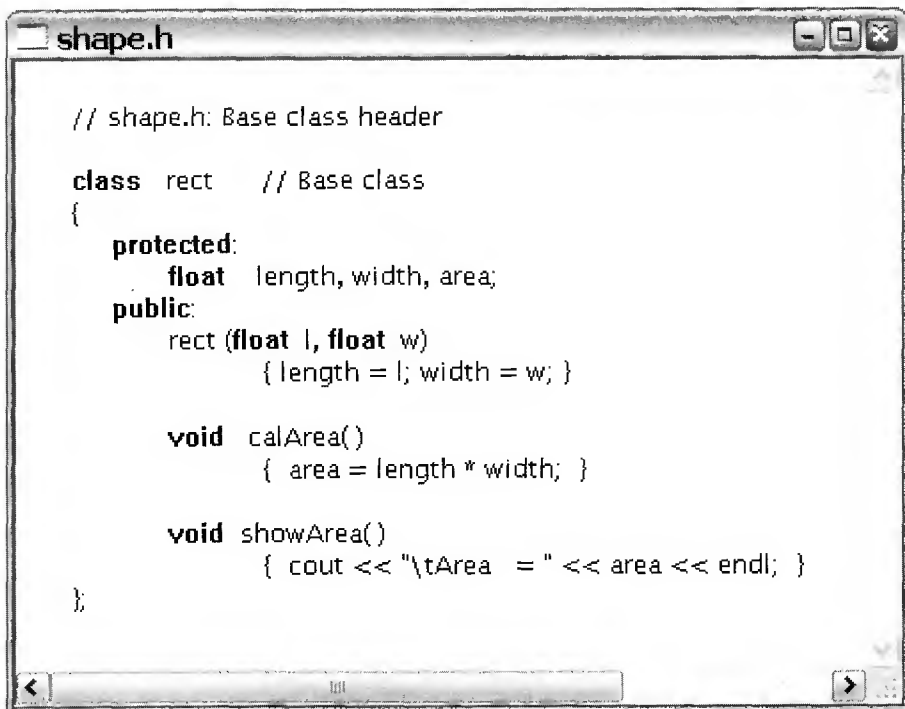
int  main()
{
    float  x,y;

    cout << "\n\tEnter length : ";      cin >> x;
    cout << "\tEnter width  : ";      cin >> y;
    cout << endl << endl;

    rect  rectan(x,y);
    rectan.calArea();
    rectan.showArea();
    cout << endl;
    return 0;
}
```

ပုံ (၉.၂)

၂။ class declaration က private နေရာမှာ protected လို့ပြောင်းရေးပြီး class rect ကို shape.h ဆိုတဲ့နာမည်နဲ့ header ဖိုင်ကို save လုပ်ပါမယ်။ private အစား protected လို့ ရေးထားတဲ့အကြောင်းက base class က data member တွေကို derived class ကနေ ဖတ်ယူခွင့်ပေးချင်လို့ပါ။ ဒီ data တွေကို base class/ derived class ပြင်ပနေရာတွေကနေ တိုက်ရိုက်ဖတ်ခွင့်မရှိပါဘူး။ shape header ဖိုင်အတွက် listing ကို ပုံ (၉. ၃) မှာဖော်ပြထားပါတယ်။



```
// shape.h: Base class header

class rect    // Base class
{
    protected:
        float  length, width, area;
    public:
        rect (float l, float w)
            { length = l; width = w; }

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

        void  showArea()
            { cout << "\tArea  = " << area << endl; }
};
```

ပုံ (၉. ၃)

၃။ class declaration ဒီတစ်ခါ ကျွန်တော်တို့ rect base class ကိုအမှီပြုပြီး volume ရှိတဲ့ derived class တစ်ခုကို create လုပ်မယ်လေ။ base class ကို shape.h ဖိုင်ထဲမှာ save လုပ်ထားတာကိုသတိရပါ။ base class ထဲမှာပါတဲ့ length နဲ့ width တို့ကိုမှီပြီး derived class ထဲမှာ depth data ကိုထပ်ထည့်ပေးလို့ ရအောင်လုပ်ပြီး volume တွက်တဲ့ function ကို call ခေါ်မယ်လေ။ ပုံ (၉. ၄) မှာဖော်ပြထားတဲ့ Ex902.cpp မှာ derived class နဲ့အတူ main( ) function ကိုပါဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// 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( );
```

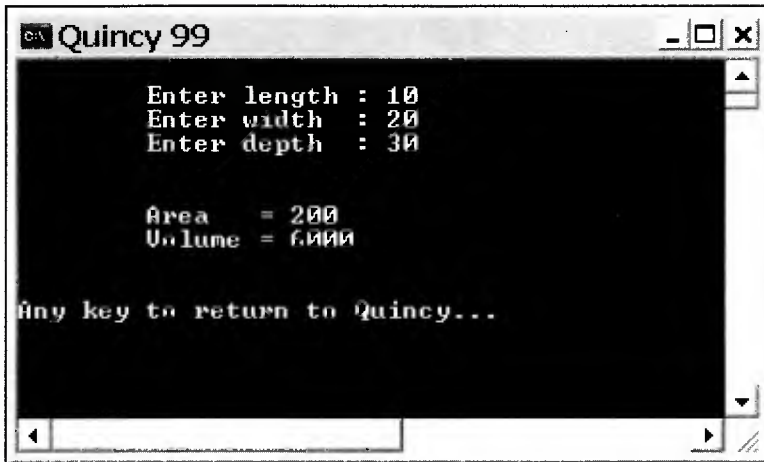
```
    box.showVol( );
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```

Ex902.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၉. ၄ မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၉. ၄)

Ex902.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်းမှာရေးထားတဲ့ `#include "shape.h"` ရဲ့အဓိပ္ပါယ်ဟာဆိုရင် Ex902.cpp program ကို compile လုပ်တဲ့အခါမှာ `shape.h` ဖိုင်ထဲမှာရှိတဲ့ code တွေကိုလည်းရောပြီး compile လုပ်ပေးပါလို့ဆိုလိုပါတယ်။ `x = 10`၊ `y = 20` နဲ့ `z = 30` ဆိုတဲ့တန်ဖိုးတွေကိုထည့်ပေးမယ်ဆိုပါစို့။ ဒါဆိုရင် derived class `cube` ဖြစ်တဲ့ object `box` ကို construct လုပ်ပြီး parameter (3) ခုကို pass လုပ်ပေးပါတယ်။
- ဒီတော့ `rect` class ရဲ့ derived class ဖြစ်တဲ့ `cube` ထဲကို `x = 10`၊ `y = 20` နဲ့ `depth = z = 30` ကို အရင်ထည့်ပေးပါတယ်။ ပြီးတော့ရင် `rect(x,y)` constructor ကနေတစ်ဆင့် `shape.h` ဖိုင်ထဲက `length = l = x = 10` နဲ့ `width = w = y = 20` လို့ assign လုပ်ပေးပါလိမ့်မယ်။
- `box.CalcVol()` ကို call ခေါ်လိုက်တဲ့အခါ `volume = length * height * depth = 10 * 20 * 30 = 6000` လို့တွက်ယူပါတယ်။ ဘာပြုလို့လဲဆိုတော့ derived class ကနေ base class ထဲက protected data member တွေဖြစ်တဲ့ `length` နဲ့ `width` တို့ကိုလှမ်းဖတ်ပြီး အသုံးပြုလို့ရပါတယ်။ `private` လို့ကြေငြာထားရင် အခုလိုဖတ်လို့ရမှာမဟုတ်ပါဘူး။



- ဒီတစ်ခါ `box.ShowVol( )` ကို `call` ခေါ်ပြီး `Volume = 6000` လို့ `display` လုပ်ခိုင်းပါတယ်။ အခုလေ့လာခဲ့တဲ့ `program` ကိုကြည့်ခြင်းအားဖြင့် `rect` ဆိုတဲ့ `base class` ကနေ `cube` ဆိုတဲ့ `derived class` တစ်ခုကို ကျွန်တော်တို့ကိုယ်တိုင် `create` လုပ်ပြီး `volume` ကို တွက်ယူလို့ရတယ်မဟုတ်လား။ ဒါဟာ `inheritance` ရဲ့သဘောတရားပါပဲ။

## Create a Derived Class 'triangle'

၁။ Listing 9.3 မှာရေးထားတဲ့ `program` ဟာ `base class rect` ကိုပဲအခြေခံပြီး `derive` လုပ်ယူထားတဲ့ `derived class triangle` ဖြစ်ပါတယ်။ ကြိုတင်တစ်ခုရဲ့ `area` ကိုရှာပေးမှာပါ။

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

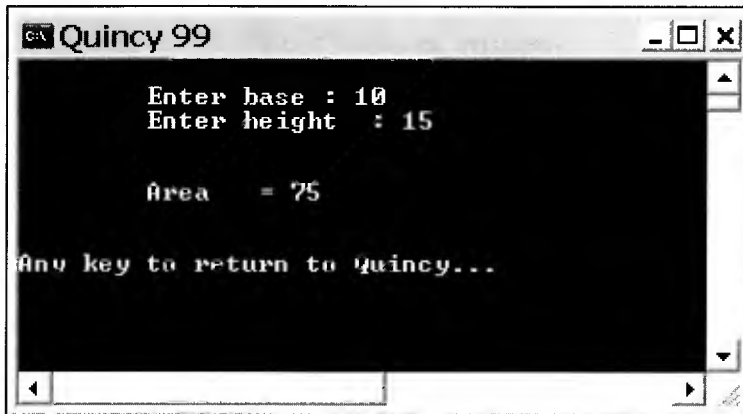
class triangle: public rect
{
    public:
        triangle (float x=0, float y=0) : rect (x, y)    { }

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

int main( )
{
    float x,y;

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

Ex903.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၉.၅) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



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 countDeriv : public counter // Derived class
{

```

```

    public:

```

```

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

```

```

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

```

```

};

```

```

int main( )
{

```

```

    countDeriv    myClass;                // myClass = 100
    cout << "\n\tmyClass = "
         << myClass.get_count( );

```

```

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

```

```

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

```

```

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

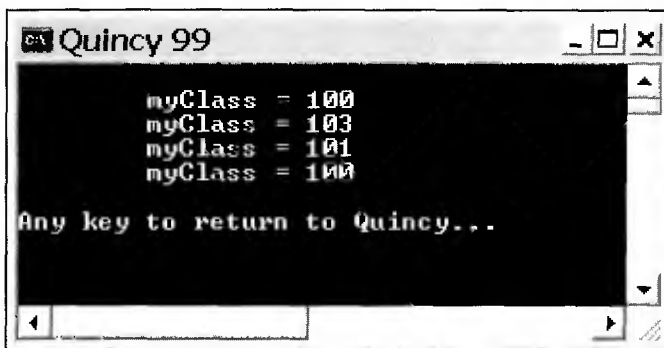
--myClass;                                // myClass = 100
cout << "\n\tmyClass = "
    << myClass.getCount( );

cout << endl;
return 0;
}

```

Ex904.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင်

- စတင်ချင်းမှာ countDeriv class object တစ်ခုဖြစ်တဲ့ myClass ကို define လုပ်လိုက်တဲ့ အတွက် base class ထဲရောက်သွားပြီး counter( ) constructor က အလုပ်စလုပ်ပါတယ်။ ဒါကြောင့် myClass.count = 100 လို့ initialize လုပ်ပေးလိုက်ပါပြီ။ myClass.count ကို display လုပ်ကြည့်ဖို့အတွက် myClass.getCount( ) function ကို call ခေါ်ပါတယ်။ ဒီ function ဟာ base class ထဲက function member တစ်ခုပါ။ count ကို protected လုပ်ထားတဲ့အတွက် base class ကနေတစ်ဆင့် ဖတ်ယူရတာပါပဲ။
- myClass++; statement ကြောင့် base class ထဲရောက်သွားပြီး counter operator ++( ) function ကနေ myClass.count ကို 1 တိုးပေးပါတယ်။ ဒီနည်းအတိုင်း (1) တွေ ထပ်တိုးပေးတဲ့အတွက် နောက်ဆုံး myClass.count ဟာ 103 ဖြစ်သွားပါပြီ။ myClass ကို decrement လည်းလုပ်ခိုင်းထားပါတယ် ၊ လေ့လာကြည့်ပါ။
- Ex904.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၉. ၆) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၉. ၆)

# 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( );
    }

};

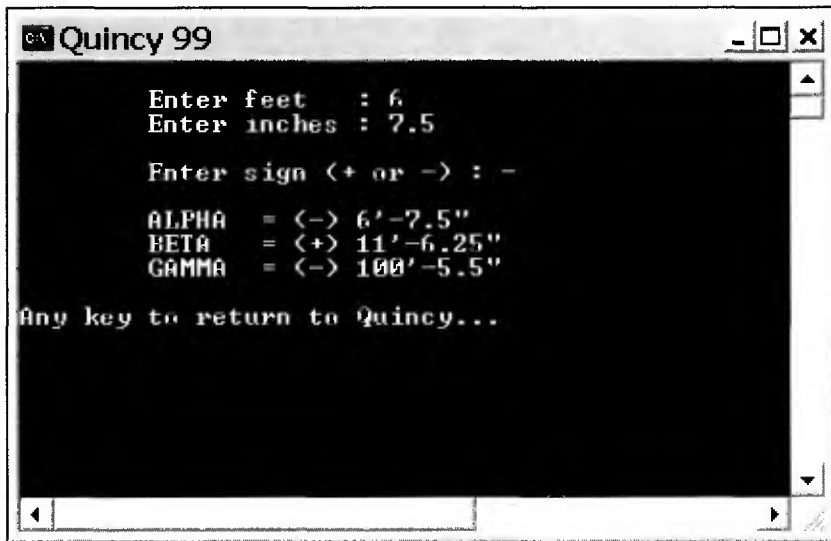
int main( )
{
    LengthSign alpha;
    alpha.getLength( );
    cout << "\n\tALPHA = ";          alpha.showLength( );

    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 လိုက်မယ်ဆိုရင် ပုံ (၉. ၇) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



```
Quincy 99
Enter feet : 6
Enter inches : 7.5

Enter sign (<+ or ->) : -

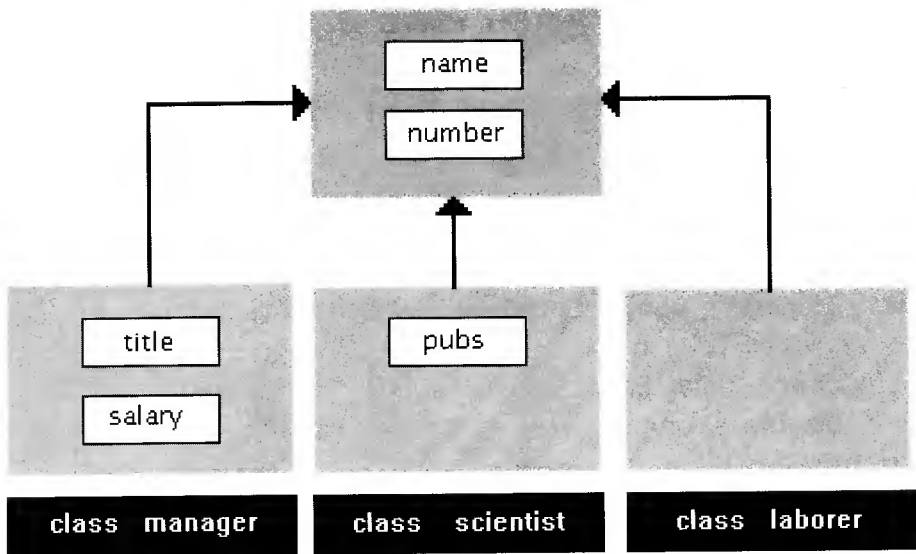
ALPHA = (<-) 6'-7.5"
BETA = (<+) 11'-6.25"
GAMMA = (<-) 100'-5.5"

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

ပုံ (၉. ၇)

## ၉.၃ Class Heirarchies

၁။ derived class တစ်ခုဟာ base class တစ်ခုရဲ့လုပ်ဆောင်မှုမှာ စွမ်းရည်အပိုဆောင်းတွေဖြည့်ပေးနိုင်သလို base class တစ်ခုရဲ့အစိတ်အပိုင်းတစ်ခုအနေနဲ့လည်း ချိတ်ဆက်ပေးနိုင်ပါတယ်။ ဥပမာ ပုံ (၉. ၈) မှာပြထားတဲ့ employee database တစ်ခုမှာဆိုရင် အလုပ်သမားအမျိုးအစား (3) မျိုးပါဝင်နေပါတယ်။ မူလ class ထဲမှာ အလုပ်သမားအားလုံးနဲ့ဆိုင်တဲ့ name , identification number တွေကို store လုပ်ထားပါတယ်။ employee ရဲ့ derived class ဖြစ်တဲ့ manager ထဲမှာတော့ title နဲ့ salary ဆိုတဲ့ data (2) ခုကိုထပ်ပြီး store လုပ်ထားပါတယ်။ ဒီတော့ manager category မှာ data (4) မျိုးကို သိမ်းထားနိုင်တဲ့သဘောပေါ့။ scientist category မှာတော့ pubs သို့မဟုတ် number of publications အတွက် data တစ်ခုကိုထပ် store လုပ်ထားပါတယ်။ laborer category မှာတော့ ဘာ data မှမသွင်းရသေးပါဘူး။ ဒီအတွက် program ကို Ex906.cpp မှာရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



୦ (୧. ୦)

## 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    m1;
    scientist   s1;
    laborer    l1;

    cout << "\n\n\tEnter data for manager1";
    m1.getData( );
    cout << "\n\tEnter data for scientist1";
    s1.getData( );
    cout << "\n\tEnter data for laborer1";
    l1.getData( );

    cout << "\n\n\tData on manager1";
    m1.putData( );
    cout << "\n\n\tData on scientist1";
    s1.putData( );
    cout << "\n\n\tData on laborer1";
    l1.putData( );
    cout << endl;

    return 0;
}

```

- Ex906.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် သူ့ထဲမှာ class employee ကိုထည့်ထားပေမယ့် အဲဒီ base class ကိုသဘောလောက်ပဲ အသုံးပြုထားပါတယ်။ အဓိကက manager၊ scientist နဲ့ laborer ဆိုတဲ့ derived class object ကိုပဲ ပိုအသုံးပြုထားပါတယ်။ တစ်ကယ်တော့ employee နဲ့ laborer တို့ဟာအတူတူပါပဲ။ class heirarchies မှာ derived class တွေဟာ base class ကနေ ဘယ်လိုဆင်းသက်လာတယ်ဆိုတာကို ဖော်ပြချင်လို့ဖြစ်ပါတယ်။ Ex906.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၉.၉) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။

```

Quincy 99

Enter data for manager1
Enter name      : ArKar
Enter number    : 12345
Enter title     : President
Enter salary    : 25000000

Enter data for scientist1
Enter name      : ZarNi
Enter number    : 34567
Enter number of publications : 999

Enter data for laborer1
Enter name      : PoZaw
Enter number    : 45678

Data on manager1
Name           : ArKar
Number         : 12345
Title          : President
Salary         : 2.5e+06

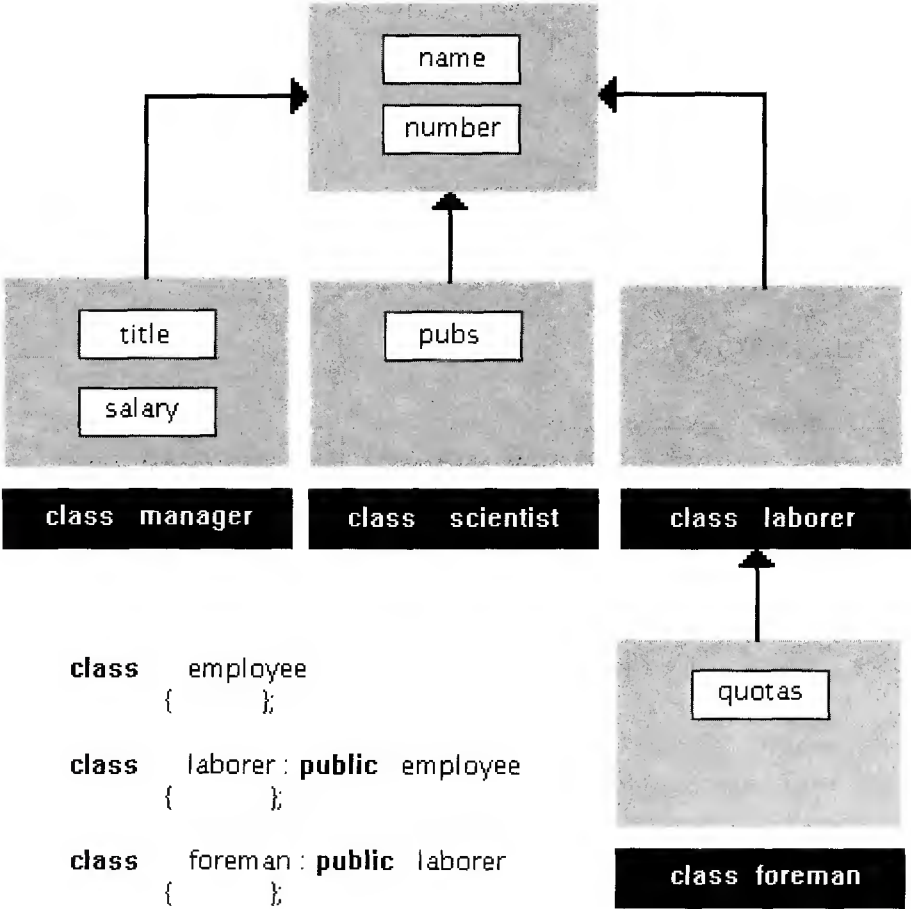
Data on scientist1
Name           : ZarNi
Number         : 34567
Number of publications : 999

Data on laborer1
Name           : PoZaw
Number         : 45678

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

ပုံ (၉.၉)

# Multiple Levels of Inheritance



ပုံ (၉. ၁၀)

ပုံ (၉. ၁၀) ကိုကြည့်မယ်ဆိုရင် class laborer ဟာ class employee ရဲ့ derived class တစ်ခုဖြစ်ပါတယ်။ foreman class က laborer class ကနေ derive ဆက်လုပ်ထားပါတယ်။ labore class ထဲမှာ data မရှိပေမယ့် foreman class ထဲမှာ quotas ဆိုတဲ့ data ရှိပါတယ်။ employee ၊ laborer နဲ့ foreman class တို့တွေ ဆက်သွယ်နေကြပုံကို Ex907.cpp program မှာဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// 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 << "\n\tNumber 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 << "\n\tQuotas : " << quotas;
    }
};

```

```

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

```

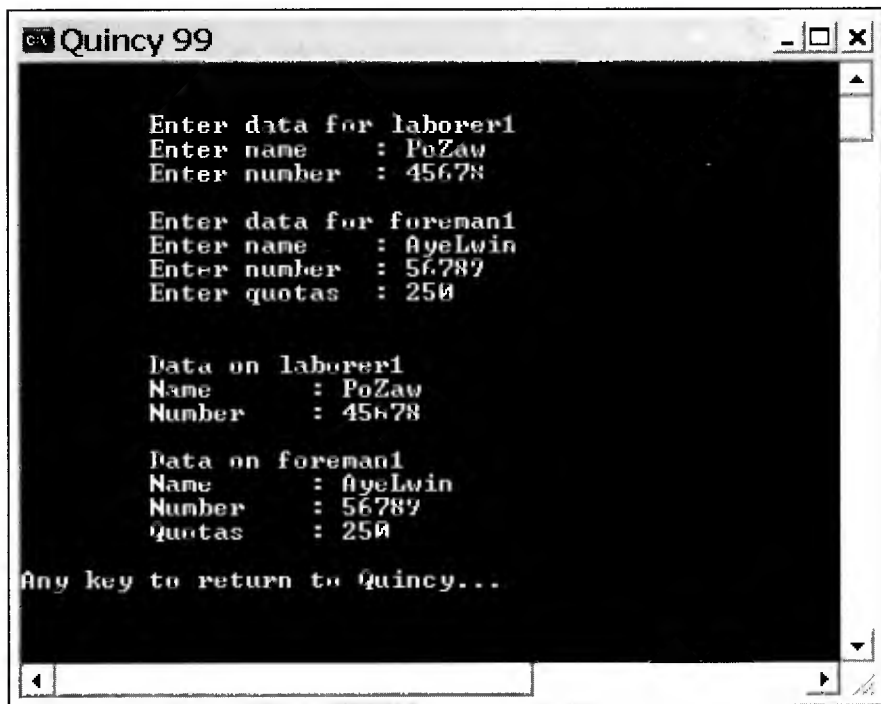
```

cout << "\n\n\tEnter data for laborer1";
l1.getData( );
cout << "\n\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;
}

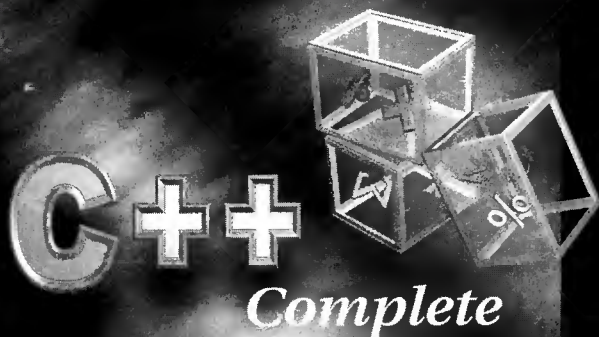
```

- Ex907.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၉. ၁၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၉. ၁၁)

# Chapter 10



## LIBRARY FUNCTIONS

Standard C function တွေဟာ Standard C++ library မှာပါဝင်ပါတယ်။ ရှေ့ပိုင်းမှာတုန်းက ကျွန်တော်တို့ရဲ့ program တွေမှာ standard C function တစ်ချို့ကိုအသုံးပြုခဲ့ပါတယ်။ ကျွန်တော်အနေနဲ့ Standard C++ မှာပါဝင်တဲ့ library function အကုန်လုံးကိုတင်ပြဖို့မရည်ရွယ်ပါဘူး။ အသုံးများတဲ့ library function တွေကိုပဲတင်ပြမှာဖြစ်ပါတယ်။ ဥပမာ ကျွန်တော်တို့ရေးတဲ့ C++ program တွေမှာ debugging code ကိုထည့်ပေးချင်ရင် `<cassert>` header ကိုထည့်ပေးခြင်းအားဖြင့် `assert( )` function ကိုအသုံးပြုလို့ရပါမယ်။ `main( )` function ကနေ `pass` လုပ်လိုက်တဲ့ data ကို `assert( )` function မှာစစ်လိုက်လို့ `false` ဖြစ်ရင် အဲဒီ argument ကို display လုပ်ပြပြီး program လည်းရပ်သွားပါလိမ့်မယ်။ program မှာအသုံးပြုထားတဲ့ `assert` macro တွေကို အသုံးမလိုတော့လို့ပြန်ဖျက်ချင်ရင် တစ်ခုချင်းလိုက်ဖျက်စရာမလိုပါဘူး။ `NDEBUG` macro ကို program မှာထည့်ပြီး `compile` လုပ်ပေးရင် `assert` macro ဟာ `disable` ဖြစ်သွားပါပြီ။ ပုံ (၁၀. ၁) မှာ ဖော်ပြထားတဲ့ `Ex1001.cpp` program မှာ `<cassert>` header ကိုအသုံးပြုထားပါတယ် ၊ လေ့လာကြည့်ပါ။



```
Ex1001.cpp

// Listing 10.1: Using <cassert> library function

#include <iostream>
#include <cassert>

void showMsg (char);

int main()
{
    char msg = 'a';
    showMsg (msg);

    return 0;
}

void showMsg (char msg)
{
    assert (msg != 'a');
    cout << msg << endl;
}
```

ပုံ (၁၀. ၁)

Ex1001.cpp program ကိုလေ့လာကြည့်ရင် main( ) ထဲက showMsg( ) function ကနေ pass လုပ်လိုက်တဲ့ data ဟာ character 'a' ဖြစ်ပါတယ်။ ဒီ 'a' ကို assert( ) function ထဲမှာ (msg != 'a') လား လို့စစ်လိုက်တဲ့အခါမှာ true ဆိုရင် message prompt မလုပ်ပါဘူး။ false ဆိုရင် assert( ) function ရဲ့ argument ကို prompt လုပ်ပြပြီး program stop ဖြစ်သွားမှာပါ။ အခုနေ Ex1001.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၀. ၂) မှာပြထားတဲ့အတိုင်းပေါ်လာမှာဖြစ်ပါတယ်။

```
Assertion failed: msg != 'a',
file c:\quincy99\programs\chap01\ex1001.cpp, line 19
abnormal program termination
Any key to return to Quincy...
```

ပုံ (၁၀. ၂)

၁၀.၁ <cerrno>

၁။ <cerrno> header ဟာဆိုရင် program မှာ error တစ်ခုတွေ့တာနဲ့ errno variable ကို define လုပ်ပြီးဖြစ်နေပါပြီ။ errno value ကိုတန်ဖိုးတစ်ခုနဲ့ assign လုပ်ပေးပါ။ အများအားဖြင့် 0 ဝဲ assign လုပ်ပါတယ်။ ပုံ (၁၀. ၃) မှာဖော်ပြထားတဲ့ Ex1002.cpp program မှာ num ဆိုတဲ့ကိန်းတစ်ခုကို နှစ်ထပ်ကိန်းရင်းရှာပြီး x နဲ့

```
Ex1002.cpp

// Listing 10.2: Using <cerrno> library function
#include <iostream>
#include <cmath>
#include <cerrno>

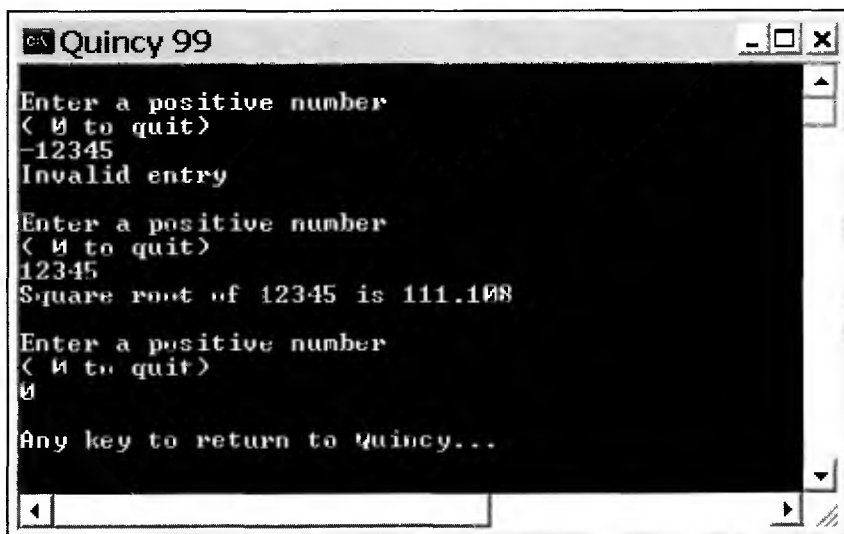
int main ()
{
    double num;
    do {
        errno = 0;
        cout << "\nEnter a positive number\n"
              << "(0 to quit)\n";
        cin >> num;
        if (num != 0)
        {
            double x = sqrt(num);

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

    return 0;
}
```

ပုံ (၁၀. ၃)

ညီပေးတဲ့ statement ကိုထည့်ရေးထားပါတယ်။ num value ကအနှုတ်တန်ဖိုးဖြစ်လို့မရပါဘူး။ ဖြစ်ခဲ့ရင် ပုံမှန် အတိုင်းကတော့ program stop ဖြစ်သွားမှာပဲ။ အခုတော့ errno variable ကိုအသုံးပြုပြီး အလုပ်ဆက်လုပ် သွားနိုင်သလို error check လုပ်ပြတာကိုလည်းတွေ့ရပါလိမ့်မယ်။ ပုံ (၁၀. ၄) မှာ program ကို run ပြထားပါတယ်။ program stop လုပ်ချင်ရင် 0 ကိုနှိပ်ပါ။



ပုံ (၁၀. ၄)

## ၁၀.၂ <cmath>

၁။ Ex1002.cpp program မှာဆိုရင် <cmath> header ကိုအသုံးပြုပြီးနှစ်ထပ်ကိန်းရင်းရှာတဲ့ sqrt( ) function ကိုအသုံးပြုနိုင်ခဲ့ပါတယ်။ အဲဒီလိုမျိုး math header တွေကို ဇယား (၁၀. ၁) မှာဖော်ပြထားပါတယ်။

ဇယား (၁၀. ၁) <cmath> Functions

Function	Returns
----------	---------

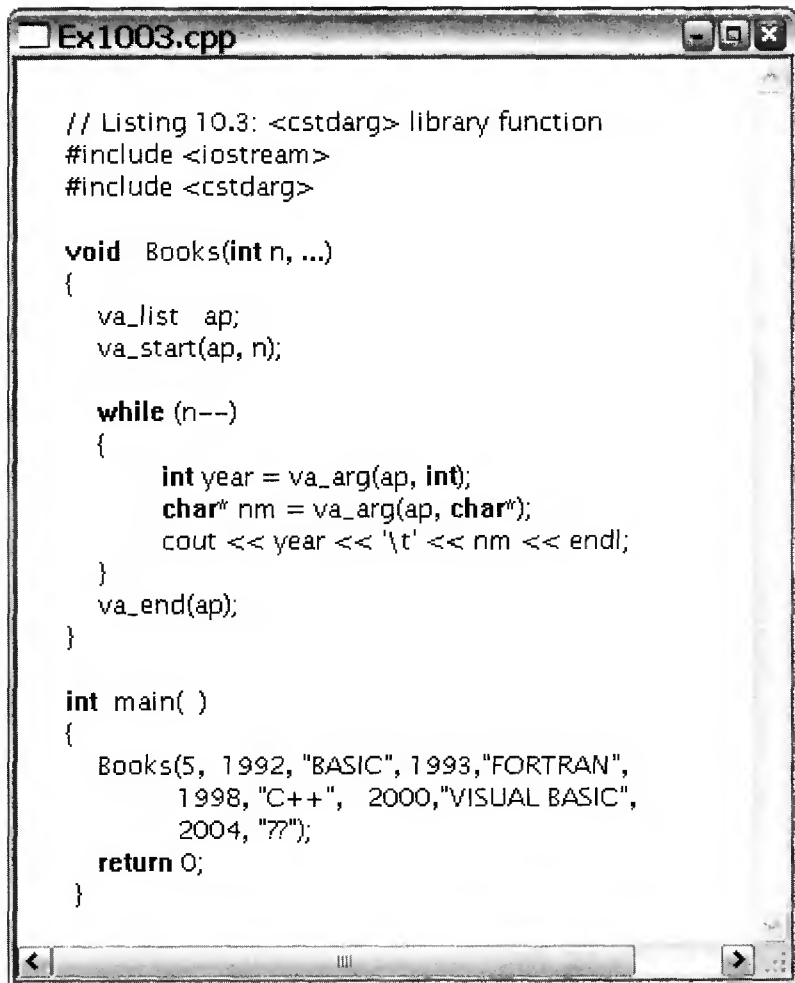
double acos (double x);

Arc cosine of x

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 <cstdlibarg>

C++ program တစ်ခုမှာ အရေအတွက်သိတဲ့ variable argument list တစ်ခုကို function မှာ pass လုပ်ပေးချင်ရင် <cstdlibarg> header ကိုအသုံးပြုရပါမယ်။ ပုံ (၁၀.၅) မှာရေးထားတဲ့ Ex1003.cpp program မှာဆိုရင် Books( ) function ကို call ခေါ်ပြီး fixed argument ဖြစ်တဲ့ n ကို n = 5 လို့ သတ်မှတ်ပြီး ကျန်တဲ့ variable list ကို called function ထဲမှာ print လုပ်ခိုင်းထားပါတယ်။ ဒီ program ရဲ့ လိုအပ်ချက်က variable list type (va\_list cp) ၊ starting variable argument macro (va\_start(ap, n) ၊ variable argument list ကိုဖတ်ယူဖို့အတွက် အသုံးပြုရမယ့် va\_arg(ap, char\*) နဲ့ va\_arg(ap, int) တို့ဖြစ်ပါတယ်။ data တွေဖတ်ယူပြီးကြောင်းကို ကြေငြာပေးမယ့် macro U va\_end(ap) ဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။



```
// Listing 10.3: <cstdlib> library function
#include <iostream>
#include <cstdlib>

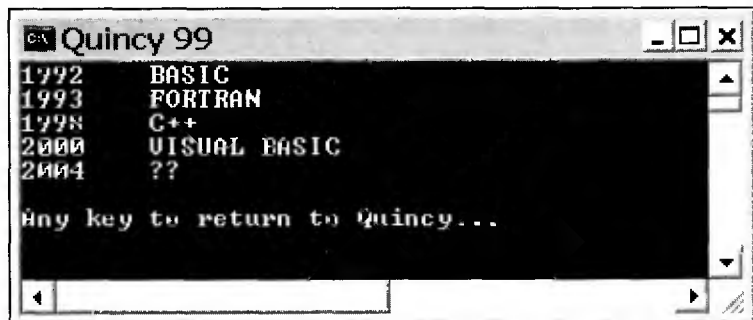
void Books(int n, ...)
{
    va_list ap;
    va_start(ap, n);

    while (n-->0)
    {
        int year = va_arg(ap, int);
        char* nm = va_arg(ap, char*);
        cout << year << '\t' << nm << endl;
    }
    va_end(ap);
}

int main( )
{
    Books(5, 1992, "BASIC", 1993, "FORTRAN",
        1998, "C++", 2000, "VISUAL BASIC",
        2004, "??");
    return 0;
}
```

ပုံ (၁၀. ၅)

Ex1003.cpp program ကို ပုံ (၁၀. ၆) မှာ run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



```
Quincey 99
1992    BASIC
1993    FORTRAN
1998    C++
2000    VISUAL BASIC
2004    ??

Any key to return to Quincey...
```

ပုံ (၁၀. ၆)

၁၀.၄ <cstdlib>

<cstdlib> header မှာ standard library function အများအပြား ပါဝင်ပါတယ်။ အဲဒါကိုအုပ်စု (4) ခုခွဲထားတာကတော့ (၁) Numerical functions (၂) Memory allocation functions (၃) system function နဲ့ (၄) random number generation function တွေ ဖြစ်ပါတယ်။ ဇယား (၁၀. ၂) မှာ <cstdlib> Numerical functions နဲ့ ဇယား (၁၀. ၃) မှာ <cstdlib> Memory Allocation function တွေကို ဖော်ပြထား ပါတယ် ၊ လေ့လာကြည့်ပါ။

ဇယား (၁၀. ၂) <cstdlib> Numerical Functions

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

ဇယား (၁၀. ၃) <cstdlib> Memory Allocation Functions

Function	Returns
<b>void</b> *calloc (int sz, int n);	Address of buffer or 0
<b>void</b> *malloc (int sz);	Address of buffer or 0
<b>void</b> 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( ) % 32; // Choose a secret number.
        do {
            cout << "Guess my secret number (0 - 32) ";
            cin >> num;

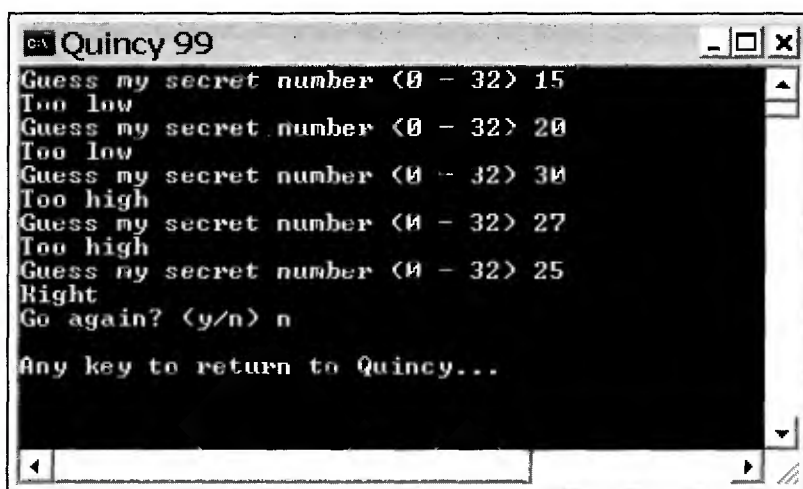
            // Report the status of the guess.
            cout << (num < assNum ? "Too low" :
                     num > assNum ? "Too high" : "Right") << endl;
        } while (num != assNum);

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

    return 0;
}

```

Ex1004.cpp program ကို ပုံ (၁၀.၇) မှာ run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



```

Quincy 99
Guess my secret number <0 - 32> 15
Too low
Guess my secret number <0 - 32> 20
Too low
Guess my secret number <0 - 32> 30
Too high
Guess my secret number <0 - 32> 27
Too high
Guess my secret number <0 - 32> 25
Right
Go again? <y/n> n
Any key to return to Quincy...

```

ပုံ (၁၀.၇)

## ၁၀.၅ <cstring>

<cstring> header တွေ့မှာဆိုရင် null-terminated character array တွေနဲ့တွဲပြီးအသုံးပြုလို့ရတဲ့ function တွေပါပါတယ်။ အဲဒါတွေက (၁) comparison functions (၂) copy functions (၃) concatenation functions (၄) strlen( ) function နဲ့ (၅) memset( ) function တွေဖြစ်ပါတယ်။ ဒီအတွက် function prototype တွေကအခုလိုပါ။

```
int      strcmp (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);
```

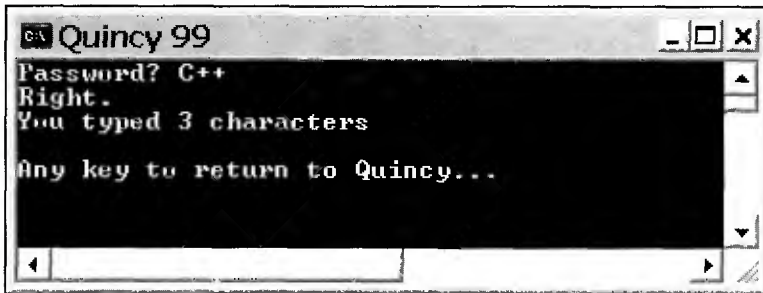


```

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

```

Ex1005.cpp program ကို ပုံ (၁၀. ၈) မှာ run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



ပုံ (၁၀. ၈)

## ၁၀.၆ <ctime>

<ctime> header ဟာဆိုရင် time နဲ့ date ကိုဆက်သွယ်ပေးတဲ့ structure တစ်ခု ၊ data type တစ်ခုနဲ့ function အများအပြားကို declare လုပ်ပေးနိုင်ပါတယ်။ structure ကိုအောက်မှာဖော်ပြထားပါတယ်။

```

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
};

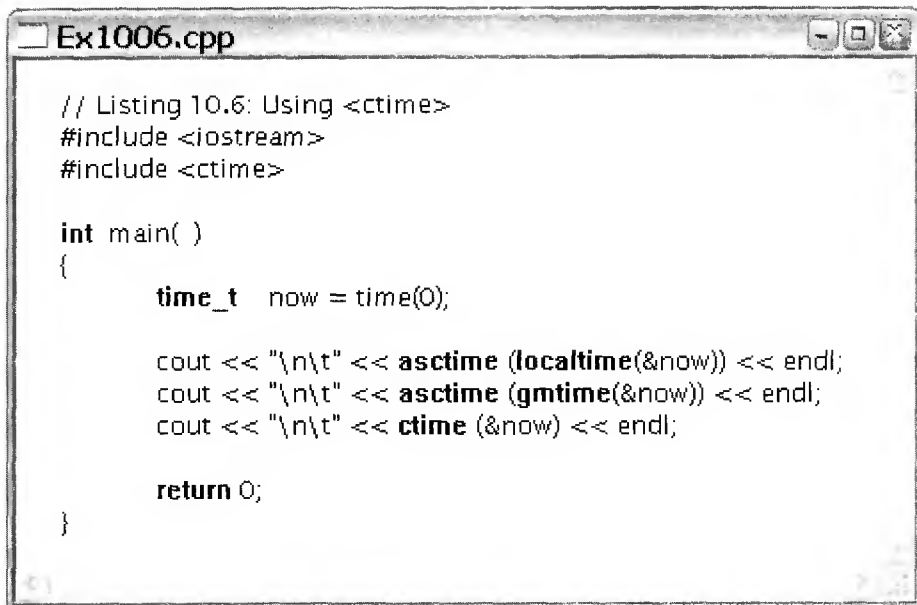
```

<ctime> function တွေအတွက် function prototype တွေက အခုလိုမျိုးဖြစ်ပါတယ်။

```

char      *asctime(const struct tm *tim);
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 *tim);
time_t    time(time_t *t);

```



```

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

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

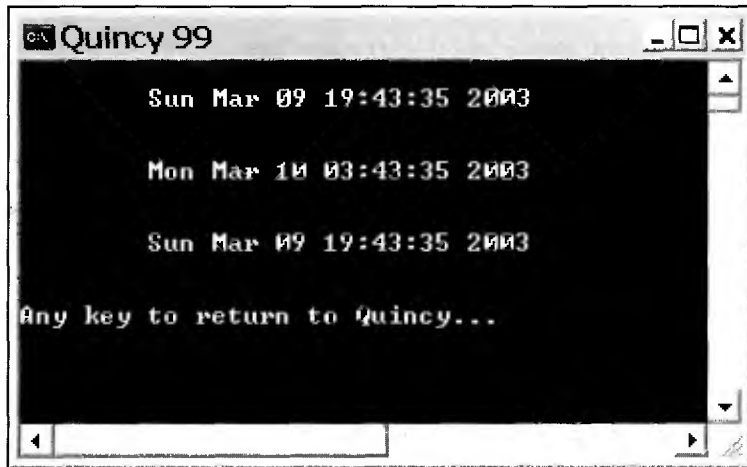
    cout << "\n\t" << asctime(localtime(&now)) << endl;
    cout << "\n\t" << asctime(gmtime(&now)) << endl;
    cout << "\n\t" << ctime(&now) << endl;

    return 0;
}

```

ပုံ (၁၀.၉)

Ex1006.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၀. ၁၀) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၁၀. ၁၀)