



```
1: // This program asks the user for information about an object of
2: // class history, and then writes this object to HISTORY.DAT file.
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:3**

# Complete

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

## **CHAPTER 11**

# **CLASS TEMPLATES**

11.1. Create a Generic Function	10
11.2. Search an Array Using a Generic Function	14
11.3. A Simple Class Template	16
11.4. Default Values for Parameters of a Specific Type	17
11.5. Using a Linked-list Template	19
11.6. Partial Template Specialization	27

## **CHAPTER 12**

# **STANDARD C++ LIBRARY**

12.1. The string Class	29
------------------------	----



**Complete**

12.2. The string Member Functions	37
12.3. Formatted Output	39
12.4. Output Member Functions	48
12.5. Input Member Functions	50

## **CHAPTER 13**

# **FILE I/O STREAMS**

13.1. The fstream Class	58
13.2. Appending to an Output File	59
13.3. Avoiding to Open an Existing File	60
13.4. The ofstream( ) Function	61
13.5. Reading Strings from a File	63
13.6. Reading until End-of-File	64



**Complete**

13.7. The seekg( ) Member Function	65
13.8. The tellg( ) Member Function	66
13.9. Read and Write a Stream File	67
13.10. Opening and Closing a Stream File	69
13.11. Objects I/O	70

## CHAPTER 14

# SEQUENCES

14.1. The vector Class Template	75
14.2. Sorting a Vector of Integers	85
14.3. The deque Class Template	87
14.4. The list Class Template	94
14.5. The stack Container Adaptor	102



**Complete**

14.6. The queue Container Adaptor	104
14.7. The priority_queue Container Adaptor	106

## **CHAPTER 15**

# **ASSOCIATIVE CONTAINERS**

15.1. The set Class Template	108
15.2. The multiset Class Template	116
15.3. The map Class Template	124
15.4. The multimap Class Template	133
15.5. User-Defined Predicates	141



**Complete**



## CHAPTER 16

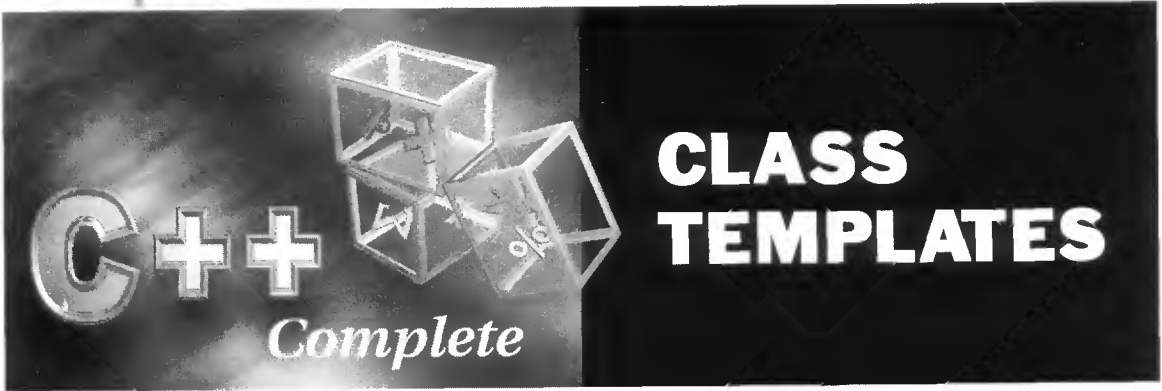
# GENERIC ALGORITHMS

16.1. Using the <code>adjacent_find( )</code> Function	144
16.2. Using the <code>count( )</code> Function	146
16.3. Using the <code>for_each( )</code> Function	147
16.4. Using the <code>fill( )</code> Function	149
16.5. Using the <code>random_shuffle( )</code> Function	150
16.6. Using the <code>partition( )</code> Function	152
16.7. Using the <code>rotate( )</code> Function	154
16.8. Using the <code>sort( )</code> Function	156
16.9. Using the <code>partial_sort( )</code> Function	158
16.10. Using the <code>merge( )</code> Function	160
16.11. Using the <code>accumulate( )</code> Function	164
16.12. Using the <code>inner_product( )</code> Function	165
16.13. Using the <code>partial_sum( )</code> Function	167
16.14. Using the <code>adjacent_difference( )</code> Function	168



## Complete

# Chapter 11



template ဆိုတာကို generic function တွေကို create လုပ်ရင်အသုံးပြုတဲ့ container တစ်ခုပါပဲ။ ဒါဖြင့် generic function ဆိုတာကရောဘာလဲ။ ဘယ်လို data မျိုးနဲ့မဆိုတွဲဖက်ပြီး အသုံးပြုလို့ရအောင် ရေးထားတဲ့ function တစ်ခုဖြစ်ပါတယ်။ generic function တစ်ခုကို execute လုပ်မယ်ဆိုလို့ရှိရင် compiler ကနေသင့်တော်တဲ့ data type ကိုရွေးချယ်ပြီး အသုံးပြုသွားမှာပါ။ ဒီဥစ္စာဟာ generic function တွေကနေ သူ့ အလိုအလျောက် overload လုပ်သွားတဲ့သဘောပေါ့ ၊ မဟုတ်ဘူးလား။ generic function တစ်ခုကို create လုပ်ရင် template ဆိုတဲ့ keyword ကိုထည့်ရေးရပါမယ်။ တစ်နည်းအားဖြင့် template ဆိုတာ generic function တစ်ခု ဘာလုပ်ရမယ်ဆိုတာကိုဖော်ပြတဲ့ framework တစ်ခုလို့ပြောရင် မမှားပါဘူး။ program တစ်ခုလုံးအတွက် အသေးစိတ်လုပ်ရမယ့် အလုပ်တာဝန်တွေက compiler ရဲ့တာဝန်ဖြစ်မှာပါပဲ။ template တစ်ခုရဲ့ပုံစံက အခုလိုပါ။

```
template    <class X>
return type  function name (parameter list)
{
    // body of function
}
```

# Create a Generic Function

// Listing 11.1: Creating a generic function

```
#include <iostream>
```

```
template <class X>
```

```
void swap (X& a, X& b)
```

```
{
```

```
    X temp = a;
```

```
    a = b;
```

```
    b = temp;
```

```
}
```

```
int main( )
```

```
{
```

```
    int i1= 100, i2= 500;
```

```
    float f1= 25.55, f2= 75.84;
```

```
    cout << "\n\tBEFORE SWAP:"
```

```
        << "\n\t\tFirst integer number    = " << i1
```

```
        << "\n\t\tSecond integer number = " << i2
```

```
        << "\n\t\tFirst float number       = " << f1
```

```
        << "\n\t\tSecond float number    = " << f2 << endl;
```

```
    swap (i1, i2);           // swapping integers
```

```
    swap(f1, f2);           // swapping floats
```

```
    cout << "\n\tAFTER SWAP:"
```

```
        << "\n\t\tFirst integer number    = " << i1
```

```
        << "\n\t\tSecond integer number = " << i2
```

```
        << "\n\t\tFirst float number       = " << f1
```

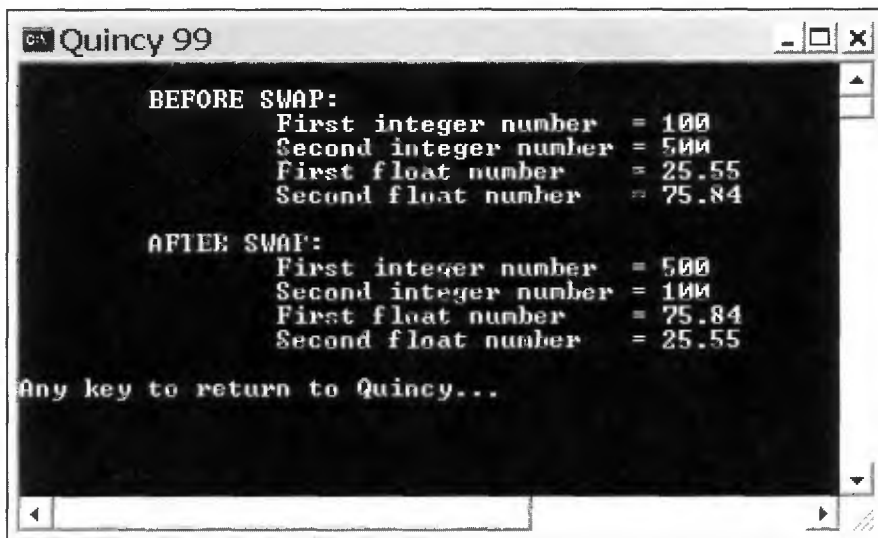
```
        << "\n\t\tSecond float number    = " << f2 << endl;
```

```
    return 0;
```

```
}
```



Ex1101.cpp ကို လေ့လာကြည့်မယ်ဆိုရင် X ဟာ data type အတွက် generic function ကနေ အသုံးပြုတဲ့ placeholder ပါ။ ဒီနာမည်ကို function body ထဲမှာလည်း အသုံးပြုလို့ရပါတယ်။ program ကို run တဲ့အခါကျရင် compiler ကနေ actual data type တစ်ခုကို placeholder နေရာမှာ အစားထည့်ပေးမှာပါပဲ။ class ဆိုတာက keyword ပါ။ Ex1101.cpp program ဟာ variable (2) ခုရဲ့တန်ဖိုးတွေ နေရာချင်းပြောင်းပေးတဲ့ generic function တစ်ခုကို create လုပ်ပေးတာဖြစ်ပါတယ်။ program ရဲ့လုပ်နည်းလုပ်ဟန်ဟာ အသုံးပြုတဲ့ variable အမျိုးအစားတွေပေါ်မှာမူမတည်ပါဘူး။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁. ၁) မှာ ပြထားတဲ့ အတိုင်းမြင်ရမှာပါ။



ပုံ (၁၁. ၁)

## Create a Generic Function 'Larger'

// Listing 11.2: Creating a generic function 'Larger'  
#include <iostream>

template <class X>

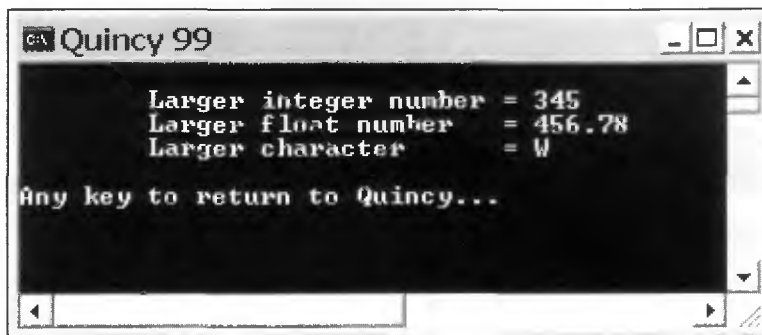
```

X Larger (X a,X b)
{
    return a > b ? a : b;
}

int main( )
{
    cout << "\n\tLarger integer number = " << Larger(12,345)
        << "\n\tLarger float number    = " << Larger(12.34, 456.78)
        << "\n\tLarger character        = " << Larger('W','V')
        << endl;
    return 0;
}

```

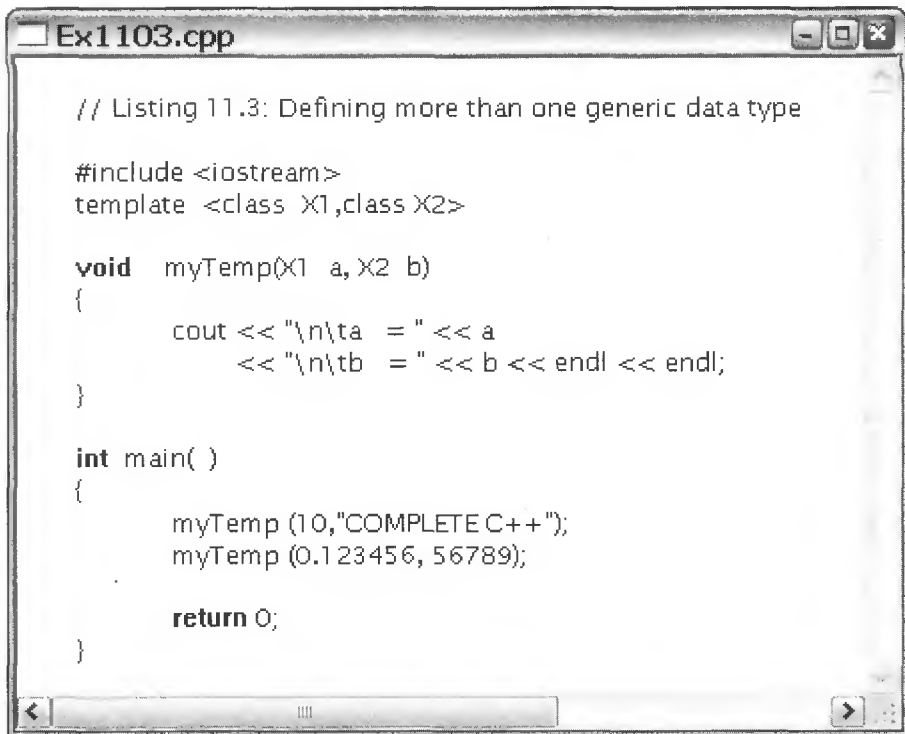
Ex1102.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁. ၂) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၁. ၂)

## Defining More than One Generic Data Type

၁။ generic function တစ်ခုမှာ data type တစ်မျိုးထက်ပိုပြီး အသုံးပြုချင်တယ်ဆိုလို့ရှိရင် data type တွေကို comma ခံပြီး ရေးထည့်ရင်ရပါတယ်။ ပုံ (၁၁. ၃) မှာဖော်ပြထားတဲ့ Ex1103.cpp program ဟာဆိုရင် generic data type (2) မျိုးကိုအသုံးပြုပြီး create လုပ်ထားတဲ့ function တစ်ခုပါပဲ။



```
// Listing 11.3: Defining more than one generic data type

#include <iostream>
template <class X1,class X2>

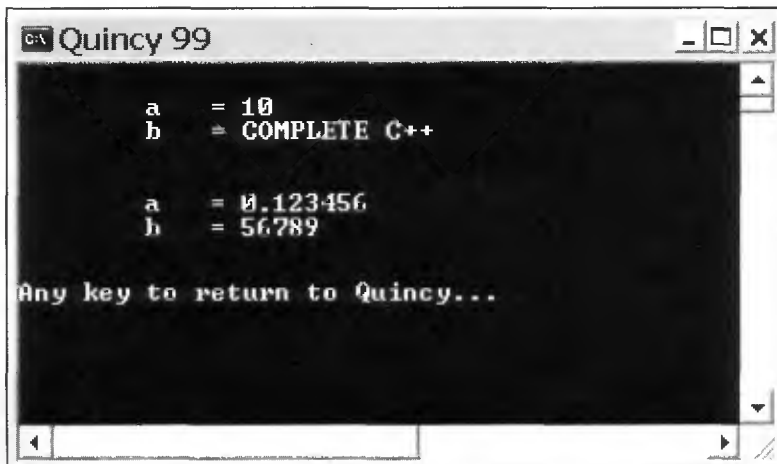
void myTemp(X1 a,X2 b)
{
    cout << "\n\ta = " << a
        << "\n\tb = " << b << endl << endl;
}

int main( )
{
    myTemp (10,"COMPLETE C++");
    myTemp (0.123456, 56789);

    return 0;
}
```

ပုံ (၁၁. ၃)

၂။ Ex1103.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁. ၄) မှာပြထားတဲ့အတိုင်း မြင်ရမှာဖြစ်ပါတယ်။



```
a      = 10
b      = COMPLETE C++

a      = 0.123456
b      = 56789

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

ပုံ (၁၁. ၄)

Ex1103.cpp program ကို လေ့လာကြည့်ရင် myTemp( ) function အတွက် instance တွေကို compiler က generate လုပ်တဲ့အခါမှာ placeholder type (2) ခုဖြစ်တဲ့ X1 နဲ့ X2 တို့နေရာမှာ (int, char) နဲ့ (double, int) data type တွေနဲ့ အစားထိုးအသုံးပြုသွားပါလိမ့်မယ်။ ဒါဆိုရင် generic function တွေဟာ overloaded function တွေနဲ့ ဆင်တူမနေဘူးလား။ ဆင်တော့ဆင်ပါတယ် ၊ ဒါပေမယ့်မတူပါဘူး။ ဘာပြုလို့လဲ ဆိုတော့ function တွေကို overload လုပ်တဲ့အခါကျရင် function တစ်ခုချင်းစီရဲ့ body ထဲမှာ မတူတဲ့အလုပ်တွေကို လုပ်ဆောင်ခိုင်းလို့ရပါတယ်။ ဒါပေမယ့် generic function ကျတော့ ပုံစံတူ action တွေကိုပဲ လုပ်ခိုင်းလို့ရမှာပါ။

# ၁၁.၂ Search an Array Using a Generic Function

// Listing 11.4: Searching an array for an object  
#include <iostream>  
#include <cstring>

```
template <class X>  
  
void find(X obj, X *list, int size)  
{  
    for (int i=0; i<size; i++)  
        if (list[i] == obj)  
        {  
            cout << "\n\t" << obj << " is PRESENT.\n";  
            return;  
        }  
    cout << "\n\t" << obj << " is ABSENT.\n";  
}  
  
int main( )  
{  
    int a[ ] = {10,14,18,25,56};
```

```

char    *c  = "This is a test";
double  d[ ] = {1.1,5.5,10.6,25.9};

find(14,a,5);

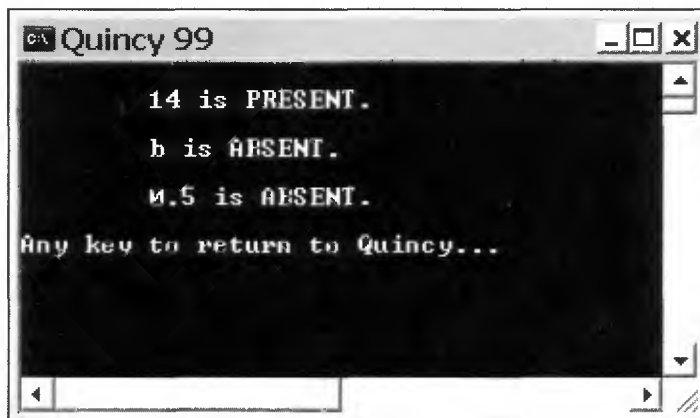
find('b',c,(int) strlen(c));

find(0.5,d,4);

return 0;
}

```

ကျွန်တော်တို့ Ex1104.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် generic function find( ) ထဲက parameter list မှာ argument (3) မျိုးကို pass လုပ်လို့ရအောင်လုပ်ထားပါတယ်။ ဥပမာ find(14,a,5) လို့ ရေးလိုက်ခြင်းအားဖြင့် array a[ ] ထဲက element (5) ခုထဲမှာ 14 ဆိုတဲ့နံပါတ်တစ်ခုပါလားလို့ စစ်ဆေးခိုင်းပြီး ပါခဲ့ရင် find( ) ကနေ 14 is PRESENT. ဆိုတဲ့ message ကို prompt လုပ်ပြမှာပါ။ မတွေ့ဘူးဆိုရင် 14 is ABSENT. လို့ display လုပ်ပြပါလိမ့်မယ်။ ဒီနည်းအတိုင်း နောက်ထပ် find( ) function တွေအလုပ်လုပ်သွားပုံ ကိုလေ့လာကြည့်ပါ။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁. ၅) မှာပြထားတဲ့အတိုင်း မြင်ရမှာဖြစ်ပါတယ်။



ပုံ (၁၁. ၅)



## ၁၁.၃ A Simple Class Template

generic function တွေကို define လုပ်ပြီး အသုံးပြုပုံကို နားလည်သွားပြီဆိုရင် generic class တွေကို ဆက်လေ့လာကြည့်ရအောင်။ generic class တစ်ခုရဲ့ general form က အခုလိုပါ။

```
template <class dtype>
class className
{
    // .....
    public:
    // .....
};
```

// Listing 11.5: Creating a class template  
#include <iostream>

```
template <class L1, class L2>
class Area
{
    L1    length;
    L2    width;

    public:
        Area (L1 l, L2 w)
            { length = l; width = w; }

        float display( )
        {
            cout << "\n\tLength  = " << length << " inches";
            cout << "\n\tWidth   = " << width  << " feet";
            return length*width/12 ;
        }
};
```

```

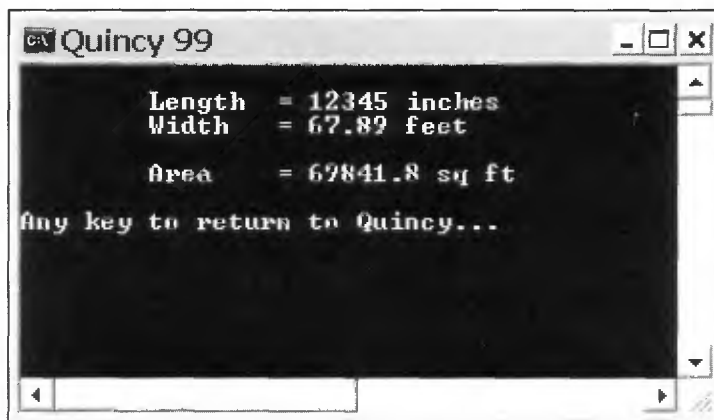
int main( )
{
    float    a = 12345;    // inches
    float    b = 67.89;    // feet

    Area <float, float> mt(a, b);
    cout << "\n\n\tArea    = " << mt.display( ) << " sq ft\n";

    return 0;
}

```

Ex1105.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁.၆) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၁.၆)

## ၁၁.၄ Default Values for Parameters of a Specific Type

ကျွန်တော်တို့ class template တစ်ခုကို create လုပ်တဲ့အခါမှာ template parameter list ထဲက default argument တစ်ခုကို တန်ဖိုးသတ်မှတ်ပေးထားလို့ရပါတယ်။ ဥပမာ Ex1106.cpp program ဟာဆိုရင်

Ex1105.cpp program ကို ပြုပြင်ထားတာပါ။ template parameter list မှာ တတိယမြောက် argument ကို integer လို့ type သတ်မှတ်ပြီး value ကို 10 လို့သတ်မှတ်ပေးထားပါတယ်။

```
// Listing 11.6: Default values for parameters of a specific type
#include <iostream>
```

```
template <class L1, class L2, int add = 10>
```

```
class Area
```

```
{
```

```
    L1    length;
```

```
    L2    width;
```

```
    public:
```

```
        Area (L1 l, L2 w)
```

```
        { length = l + add; width = w + add; }
```

```
        float display( )
```

```
        {
```

```
            cout << "\n\tLength = " << length << " inches";
```

```
            cout << "\n\tWidth  = " << width  << " feet";
```

```
            return ( length*width/12 );
```

```
        }
```

```
};
```

```
int main( )
```

```
{
```

```
    float a = 12345;    // inches
```

```
    float b = 67.89;    // feet
```

```
    Area <float, float> mt1(a, b);
```

```
    cout << "\n\tArea    = " << mt1.display( ) << " sq ft\n";
```

```
    Area <float, float, 100> mt2(a, b);
```

```
    cout << "\n\tArea    = " << mt2.display( ) << " sq ft\n";
```

```
    return 0;
```

```
}
```

Ex1106.cpp program ကို လေ့လာကြည့်ရင် main( ) function ထဲမှာ  $a = 12345$  နဲ့  $b = 67.89$  ကို initialize လုပ်ပေးထားပြီး mt1(a, b) ဆိုတဲ့ template function ကို call ခေါ်ပါတယ်။ ဒါဆိုရင် class Area ထဲဝင်လာပြီး  $length = l + add<default> = 12345 + 10 = 12355$  နဲ့  $width = w + add<default> = 67.89 + 10 = 77.89$  လို့ assign လုပ်ပေးမှာပါ။ default value ကို အသုံးမပြုချင်ဘူးဆိုရင် mt2(a, b) ကို call ခေါ်ရပါမယ်။ ဒါဆိုရင်  $length = l + add = 12345 + 100 = 12445$  နဲ့  $width = w + add = 67.89 + 100 = 167.89$  လို့နောက်တစ်မျိုး assign လုပ်ပေးမှာပါ။ template definition က default value 10 ကိုအသုံးမပြုတော့ပါဘူး။ Ex1106.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁. ၇) မှာ ပြထားတဲ့အတိုင်း မြင်ရမှာဖြစ်ပါတယ်။

```

Quincy 99

Length = 12355 inches
Width  = 77.89 feet
Area   = 80194.2 sq ft

Length = 12445 inches
Width  = 167.89 feet
Area   = 174116 sq ft

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

ပုံ (၁၁. ၇)

# ၁၁.၇ Using a Linked-list Template

```

// Listing 11.7: A Linked list of integers
#include <iostream>
#include "linklist.h"

int main( )
{
  
```

```

LinkedList <int> IntList;
for (int i = 0; i < 10; i++)
    IntList.AppendEntry(i);

int* ip = IntList.FirstEntry();
cout << "\n\t";

while (ip)
{
    cout << *ip << ' ';
    if (*ip == 0 || *ip == 5 || *ip == 8)
        IntList.RemoveEntry();
    ip = IntList.NextEntry();
}

cout << "\n\n\t";
while ((ip = IntList.NextEntry()) != 0)
    cout << *ip << ' ';

cout << endl;
return 0;
}

```

Ex1107.cpp program ကို လေ့လာကြည့်မယ်ဆိုလို့ရှိရင် LinkedList object တစ်ခုကို type (int) parameter အသုံးပြုပြီး declare လုပ်ပါတယ်။ စတင်ချင်း for loop ထဲမှာ AppendEntry( ) ကို call ခေါ်ပြီး list ထဲကို integer (10) ခုကို ထည့်ပေးပါတယ်။ list ထဲက first integer ကို FirstEntry( ) function နဲ့ point လုပ်ပြီး while loop ထဲမှာ တစ်ခုချင်း display လုပ်ခိုင်းပါတယ်။ 0 ၊ 5 နဲ့ 8 တို့ကိုတွေ့ရင် RemoveEntry() call ခေါ်ပြီး ဖျက်ပစ်ခဲ့ပါတယ်။ NextEntry( ) call ဟာ list ထဲကနောက်ဆုံး integer အထိလှမ်းခေါ်ပေးသွားမှာပါ။ အားလုံးပြီးသွားလို့ နောက်တစ်ကြိမ် list ထဲက integer တွေကို display လုပ်ခိုင်းရင် integer (3) ခုက ပါမှာမဟုတ်တော့ပါဘူး။ LinkedList.h header ကို တစ်ဘက်စာမျက်နှာမှာ ဖော်ပြထားပါတယ်။ Ex1107.cpp program မှာ #include "LinkedList.h" ကိုရေးထည့်ပေးရင် program ကို run လို့ရပါပြီ။



```
// linklist.h
```

```
#ifndef LINKLIST_H  
#define LINKLIST_H
```

```
template <class T> class LinkedList;
```

```
template <class T>
```

```
// The linked-list entry.
```

```
class ListEntry
```

```
{
```

```
    T thisentry;
```

```
    ListEntry* nextentry;
```

```
    ListEntry* preentry;
```

```
    ListEntry(T& entry);
```

```
    friend class LinkedList<T>;
```

```
};
```

```
template <class T>
```

```
// Construct a linked-list entry.
```

```
ListEntry<T> :: ListEntry(T &entry)
```

```
{
```

```
    thisentry = entry;
```

```
    nextentry = 0;
```

```
    preentry = 0;
```

```
}
```

```
template <class T>
```

```
// The linked list.
```

```
class LinkedList
```

```
{
```

```
    // The list head.
```

```
    ListEntry<T>* firstentry;
```

```
    ListEntry<T>* lastentry;
```

```
    ListEntry<T>* iterator;
```

```
    void RemoveEntry(ListEntry<T> *lentry);
```

```
    void InsertEntry(T& entry, ListEntry<T> *lentry);
```

**public:**

```
    LinkedList();  
    ~LinkedList();  
    void AppendEntry(T& entry);  
    void RemoveEntry(int pos = -1);  
    void InsertEntry(T& entry, int pos = -1);  
    T* FindEntry(int pos);  
    T* CurrentEntry();  
    T* FirstEntry();  
    T* LastEntry();  
    T* NextEntry();  
    T* PrevEntry();  
};
```

**template** <class T>

// Construct a linked list.

LinkedList<T> :: LinkedList( )

```
{  
    iterator = 0;  
    firstentry = 0;  
    lastentry = 0;  
}
```

**template** <class T>

// Destroy a linked list

LinkedList<T> :: ~LinkedList()

```
{  
    while (firstentry) RemoveEntry(firstentry);  
}
```

**template** <class T>

// Append an entry to the linked list.

**void** LinkedList<T> :: AppendEntry(T& entry)

```
{  
    ListEntry<T>* newentry = new ListEntry<T>(entry);  
    newentry->preventry = lastentry;  
    if (lastentry)
```

```

        lastentry->nextentry = newentry;
    if (firstentry == 0)
        firstentry = newentry;
    lastentry = newentry;
}

```

```

template <class T>
// Remove an entry from the linked list.
void    LinkedList<T> :: RemoveEntry(ListEntry<T>* lentry)
{
    if (lentry == 0)    return;
    if (lentry == iterator)
        iterator = lentry->preentry;

    // Repair any break made by this removal.
    if (lentry->nextentry)
        lentry->nextentry->preentry = lentry->preentry;
    if (lentry->preentry)
        lentry->preentry->nextentry = lentry->nextentry;

    // Maintain list head if this is last and/or first.
    if (lentry == lastentry)
        lastentry = lentry->preentry;
    if (lentry == firstentry)
        firstentry = lentry->nextentry;

    delete lentry;
}

```

```

template <class T>
// Insert an entry into the linked list.
void    LinkedList<T> :: InsertEntry(T& entry, ListEntry<T>* lentry)
{
    ListEntry<T>* newentry = new ListEntry<T>(entry);
    newentry->nextentry = lentry;
}

```

```

    if (lentry)
    {
        newentry->preventry = lentry->preventry;
        lentry->preventry = newentry;
    }

    if (newentry->preventry)
        newentry->preventry->nextentry = newentry;
    if (lentry == firstentry)
        firstentry = newentry;
}

```

```

template <class T>
// Remove an entry from the linked list.
void LinkedList<T> :: RemoveEntry(int pos)
{
    FindEntry(pos);
    RemoveEntry(iterator);
}

```

```

template <class T>
// Insert an entry into the linked list.
void LinkedList<T> :: InsertEntry(T& entry, int pos)
{
    FindEntry(pos);
    InsertEntry(entry, iterator);
}

```

```

template <class T>
// Return the current linked-list entry.
T* LinkedList<T>::CurrentEntry()
{
    return iterator ? &(iterator->>thisentry) : 0;
}

```

```

template <class T>
// Return a specific linked-list entry.

```

```

template<class T> LinkedList<T>::FindEntry(int pos)
{
    if (pos != -1)
    {
        iterator = firstentry;

        if (iterator)
        {
            while (pos--)
                iterator = iterator->nextentry;
        }
    }

    return CurrentEntry();
}

```

```

template<class T>
// Return the first entry in the linked list.
T* LinkedList<T> :: FirstEntry()
{
    iterator = firstentry;
    return CurrentEntry();
}

```

```

template<class T>
// Return the last entry in the linked list.
T* LinkedList<T> :: LastEntry()
{
    iterator = lastentry;
    return CurrentEntry( );
}

```

```

template<class T>
// Return the next entry in the linked list.
T* LinkedList<T>::NextEntry( )

```



```

{
    if (iterator == 0)
        iterator = firstentry;
    else
        iterator = iterator->nextentry;

    return CurrentEntry( );
}

```

```

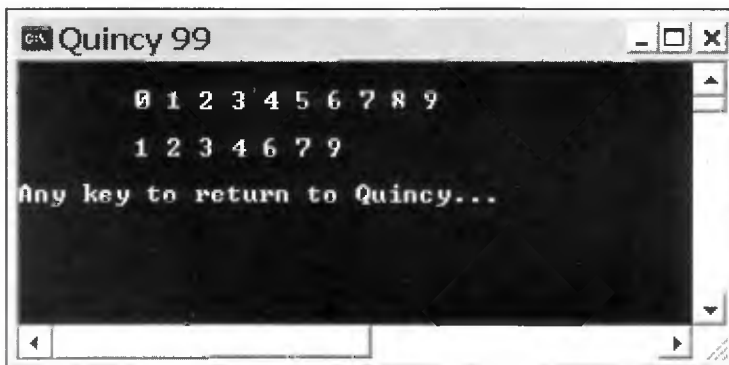
template <class T>
// Return the previous entry in the linked list.
T* LinkedList<T> :: PrevEntry( )
{
    if (iterator == 0)
        iterator = lastentry;
    else
        iterator = iterator->preventry;

    return CurrentEntry();
}

#endif

```

Ex1107.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁. ၈) မှာပြထားတဲ့အတိုင်း မြင်ရမှာဖြစ်ပါတယ်။



ပုံ (၁၁. ၈)

## ၁၁.၆ Partial Template Specialization

template partial specialization ဆိုတာ primary class template ရဲ့ အမည်ကိုယူသုံးသော်လည်း parameter တွေဟာ အမျိုးမတူတဲ့ special parameter တွေကိုအသုံးပြုထားတဲ့ template မျိုးကိုဆိုလိုပါတယ်။ ဥပမာ primary class template တစ်ခုမှာ object (2) ခုကို parameter တွေအနေနဲ့ pass လုပ်ခွင့်ပြုထားပြီး အဲဒီ object တွေကိုကွန်ပျူတာမှာ display လုပ်လို့ရတယ်ဆိုပါစို့။ တစ်ကယ်လို့ ဒုတိယ object နေရာမှာ char value တစ်ခုကိုပြောင်းထားပြီး integer value အနေနဲ့ ပထမ object မှာ ထည့်ပေါင်းပေးနိုင်တဲ့ template မျိုးကို create လုပ်ခဲ့မယ်ဆိုရင် အဲဒီဥပမာကို template partial specialization လို့ခေါ်ပါတယ်။ Ex1108. cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 11.8: Template partial specialization

```
#include <iostream>
```

```
template <class T1, class T2>
```

```
class MyTemp
```

```
{
```

```
    T1 obj1;    T2 obj2;
```

```
public:
```

```
    MyTemp (T1 o1, T2 o2) : obj1(o1), obj2(o2){    }
```

```
    void display( )
```

```
{
```

```
        cout << "\n\tOBJECT DISPLAY\n\t-----\n"
             << "\tObject 1: " << obj1 << "\n\tObject 2: "
             << obj2 << endl << endl;
```

```
}
```

```
};
```

```
template <class T>
```

```
class MyTemp <T, char>
```

```
{
```

```
    T obj1, obj2;
```

```
public:
```

```
    MyTemp(T o1, char c) : obj1(o1), obj2(o1)
```

```
        {obj2 += (int) c;}
```

```

void display( )
{
    cout << "\n\tOBJECT DISPLAY\n"
         << "\t-----\n"
         << "\tObject 1: " << obj1 << endl
         << "\tObject 2: " << obj2 << endl
         << endl;
}

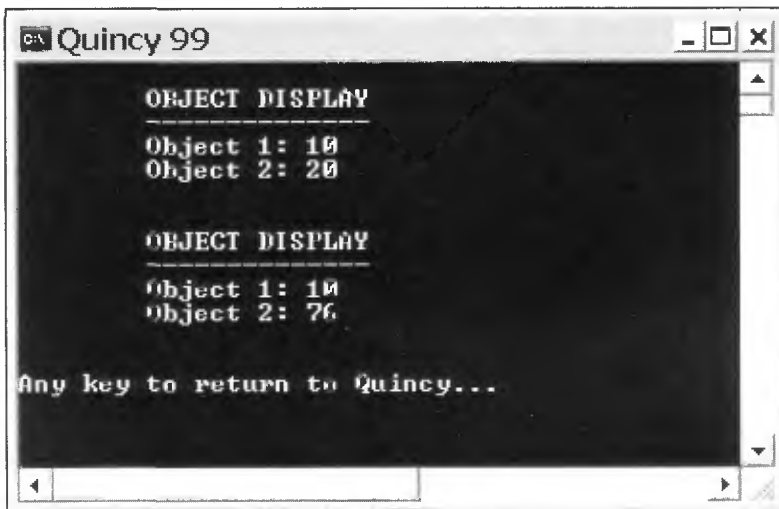
};

int main( )
{
    MyTemp <int, int>    mt1(10, 20);
    MyTemp <int, char>  mt2(10, 'B');    // 'B' = 66

    mt1.display( );
    mt2.display( );
    return 0;
}

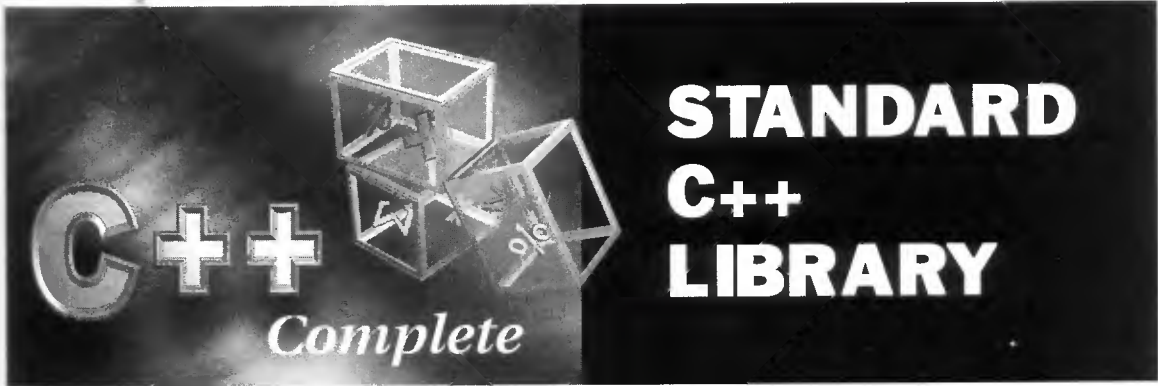
```

Ex1108.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၁.၉) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၁.၉)

# Chapter 12



BASIC language ကိုစတင်လေ့လာခဲ့သူများအနေနဲ့ C language ကို ဆက်လက်လေ့လာကြတဲ့ အခါ C မှာ string operator တွေအဖြစ်ပါဝင်တဲ့ strcpy( ) နဲ့ strcmp( ) function အုပ်စု (2) ခုကိုပဲတွေ့ရတဲ့အတွက် အားမလိုအားမရဖြစ်ခဲ့ကြပါတယ်။ သို့သော်လည်း C ကနေ C++ ကို ဆက်လက်လေ့လာကြတဲ့အခါမှာ ဒီအားနည်းချက်တွေအားလုံး ပျောက်ကွယ်သွားတာကိုတွေ့ရမှာပါ။ C++ မှာပါဝင်တဲ့ string class ဟာဆိုရင် string object တွေနဲ့ပတ်သက်ပြီး အပြည့်စုံဆုံးလုပ်ပေးနိုင်ပြီလေ။ ဥပမာ string object တွေကို construct လုပ်မယ်၊ assign လုပ်မယ်၊ concatenate လုပ်မယ်၊ compare လုပ်မယ်၊ search လုပ်မယ်ဆိုရင် <string> header ကို program မှာ include လုပ်ပေးတာနဲ့ အကုန်လုပ်လို့ရပါလိမ့်မယ်။

## ၁၂.၁ The string Class

၁။ ပုံ (၁၂. ၁) မှာဖော်ပြထားတဲ့ Ex1201.cpp program ဟာဆိုရင် string တွေ construct လုပ်နည်း အမျိုးမျိုးကိုရေးပြထားတဲ့ program တစ်ခုဖြစ်ပါတယ်။ စတင်ချင်းမှာ string s1; လို့ရေးထားတဲ့ statement ကနေ empty string တစ်ခုကို construct လုပ်ပေးမှာပါ။ နောက်တစ်နည်းက string s2("This is a string");

လို့ရေးခြင်းအားဖြင့် ကွင်းထဲက string object ကနေ string object s2 ကို construct လုပ်ပေးပါလိမ့်မယ်။ တစ်ကယ်လို့ ch[ ] ဟာ character array တစ်ခုဖြစ်မယ်ဆိုရင် string s3(ch); လို့ရေးလိုက်တာနဲ့ string object s3 ဟာ construct ဖြစ်သွားပြီး content က ch[ ] ခဲ့ character array ပဲဖြစ်မှာပါ။

```

Ex1201.cpp

// Listing 12.1: Constructing strings
#include <iostream>
#include <string>

int main()
{
    string s1;

    string s2("This is a string");
    cout << "\n\ts2 = " << s2 << endl;

    // Construct a string object from a character array
    char ch[] = "This is a character array";
    string s3(ch);
    cout << "\n\ts3 = " << s3 << endl;

    return 0;
}

```

ပုံ (၁၂. ၁)

၂။ Ex1201.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၂) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။

```

Quincy 99

s2 = This is a string
s3 = This is a character array
Any key to return to Quincy...

```

ပုံ (၁၂. ၂)



# Assigning Strings

၁။ string object တွေကို construct လုပ်ပြီးပြီဆိုရင် တစ်ခုနဲ့တစ်ခု assign ပြန်လုပ်ပေးလို့ရပါတယ်။ ဥပမာ string s2("This is a string"); ကို construct လုပ်ပြီးရင် string s1 = s2; အနေနဲ့ assign လုပ်ပေးလို့ရတယ်လေ။ Ex1202.cpp program ကိုလေ့လာကြည့်ပါ။

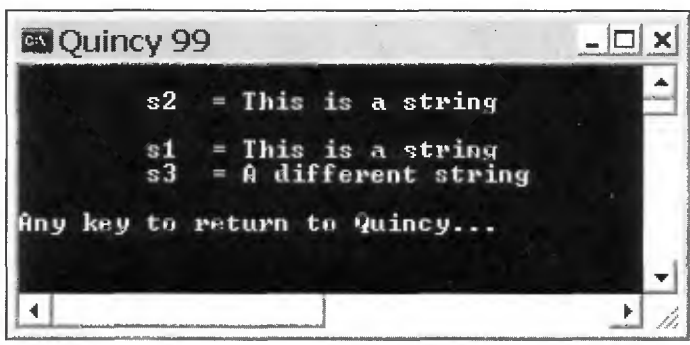
```
// Listing 12.2: Assigning strings
#include <iostream>
#include <string>

int main( )
{
    string s1;
    string s2("This is a string");
    cout << "\n\t s2 = " << s2 << endl;

    s1 = s2;
    cout << "\n\t s1 = " << s1 << endl;
    string s3 = "A different string";
    cout << "\t s3 = " << s3 << endl;

    return 0;
}
```

၂။ Ex1202.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၃) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂.၃)

# Concatenating Strings

၁။ string object တွေကို concatenation operator + နဲ့ += တို့ကိုအသုံးပြုပြီး တစ်ခုနဲ့တစ်ခုဆက်ပေးလို့ရပါတယ်။ Ex1203.cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 12.3: Concatenating strings

```
#include <iostream>
```

```
#include <string>
```

```
int main( )
```

```
{
```

```
    string s4("Hello ");
```

```
    string s5("Complete C++");
```

```
    string s6 = s4 + s5;
```

```
    cout << "\n\ts4 = " << s4 << "\n\ts5 = " << s5
```

```
        << "\n\ts6 = " << s6 << endl;
```

```
    s4 += s5;
```

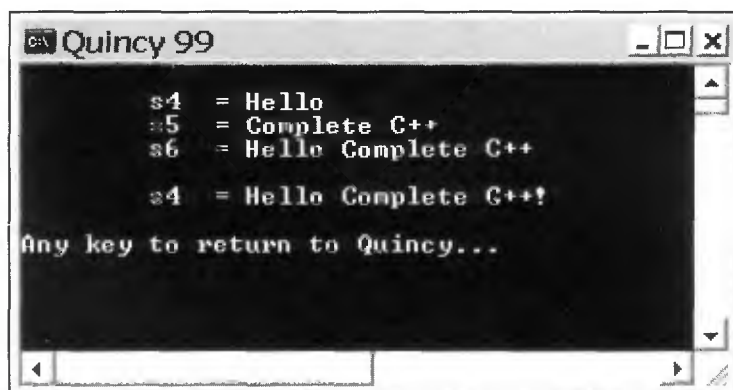
```
    s4 += '!';
```

```
    cout << "\n\ts4 = " << s4 << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1203.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၄) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



```
C:\ Quincy 99
s4 = Hello
s5 = Complete C++
s6 = Hello Complete C++
s4 = Hello Complete C++!
Any key to return to Quincy...
```

ပုံ (၁၂.၄)

# Subscripting Strings

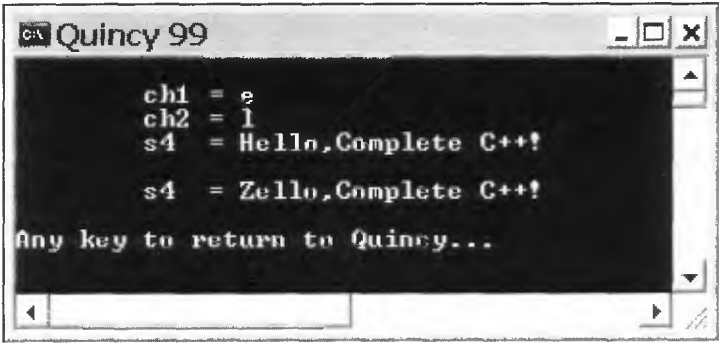
၁။ string object တစ်ခုကနေ single character တစ်ခုကိုထုတ်ယူမယ်။ ဒါမှမဟုတ် character တစ်ခုကိုနောက် character တစ်ခုနဲ့ အစားထိုးပေးချင်ရင် subscript operator [ ] သို့မဟုတ် at(int) member function ကိုအသုံးပြုရပါမယ်။ Ex1204.cpp program ကိုလေ့လာကြည့်ပါ။

```
// Listing 12.4: Subscripting strings
#include <iostream>
#include <string>

int main( )
{
    string s4("Hello Complete C++! ");
    char ch1 = s4[1];
    char ch2 = s4.at(2);

    s4[5] = ',';
    cout << "\n\tch1 = " << ch1 << "\n\tch2 = " << ch2
         << "\n\tts4 = " << s4 << endl;
    s4.at(0) = 'Z';
    cout << "\n\tts4 = " << s4 << endl;
    return 0;
}
```

၂။ Ex1204.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၅) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂.၅)

# Substrings

၁။ string object တစ်ခုကနေ substring တစ်ခုကို ခွဲထုတ်ယူချင်ရင် substr ( ) function ကိုအသုံးပြုရပါမယ်။ Ex1205.cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 12.5: Substrings

```
#include <iostream>
```

```
#include <string>
```

```
int main( )
```

```
{
```

```
    string s4("Hello,Complete C++!");
```

```
    cout << "\n\ts4.substr(6,8) = " << s4.substr(6,8);
```

```
    string s7(s4.substr(0,5));
```

```
    cout << "\n\ts7 = " << s7 << endl;
```

```
    cout << "\n\tLength of string s4 = "
```

```
        << s4.length( ) << endl;
```

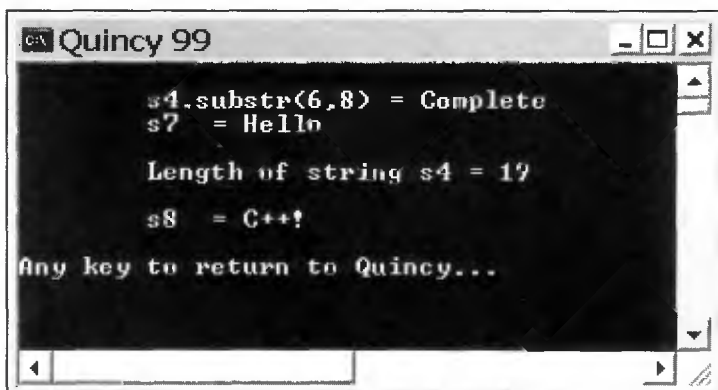
```
    string s8(s4.substr(s4.length( )-4,4));
```

```
    cout << "\n\ts8 = " << s8 << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1205.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၆) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂.၆)

# Searching Strings

၁။ standard C++ library ထဲက string class မှာ find( ) ၊ rfind( ) ဆိုတဲ့ overloaded function တွေကို ထည့်ပေးထားပါတယ်။ find( ) function ကိုအသုံးပြုပြီး string object ထဲမှာ matching substring၊ single character သို့မဟုတ် character array တို့ကို string အစကနေစပြီး forward search လုပ်နိုင်ပါတယ်။ တစ်ကယ်လို့ search argument ကို ရှာမတွေ့ဘူးဆိုရင် -1 value ကို return လုပ်ပေးပါလိမ့်မယ်။ repeated search argument ရှာချင်ရင် rfind( ) function ကိုအသုံးပြုရင်ရပါတယ်။ Ex1206.cpp program ကို လေ့လာကြည့်ပါ။

// Listing 12.6: Searching strings

```
#include <iostream>
```

```
#include <string>
```

```
int main( )
```

```
{
```

```
    string s4("Hello ");
```

```
    string s5("Complete C++");
```

```
    s4 += s5;
```

```
    s4 += '!';
```

```
    s4[5] = ',';
```

```
    s4.at(0) = 'Z';
```

```
    // Searching strings
```

```
    int n = s4.find("C++");
```

```
    cout << "\n\tn = " << n << endl;
```

```
    n = s4.find('!');
```

```
    cout << "\n\tn = " << n << endl;
```

```
    n = s4.rfind('!');
```

```
    cout << "\n\tn = " << n << endl;
```

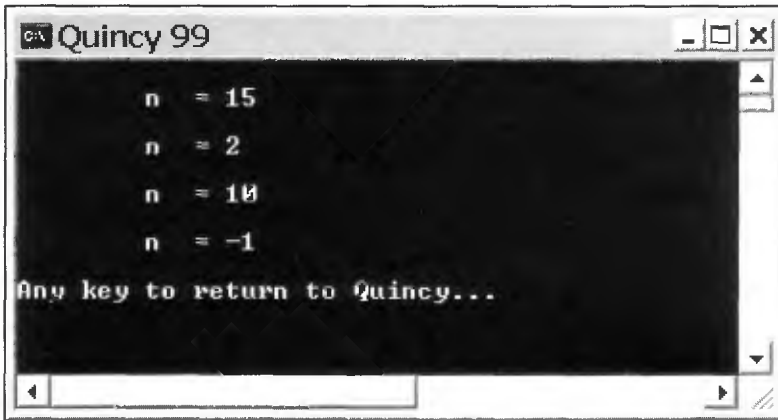
```
    n = s4.find("bye");
```

```
    cout << "\n\tn = " << n << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1206.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၇) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၇)

## Comparing Strings

၁။ Ex1207.cpp program မှာ string object တွေ နှိုင်းယှဉ်နည်းကို program တစ်ခု ရေးပြထားပါတယ်၊ လေ့လာကြည့်ပါ။

// Listing 12.7: Comparing strings

```
#include <iostream>
#include <string>

int main( )
{
    string s1("C++ ");
    string s2("Programming ");

    if ("Bye " < s1)
        if (s2 == "Programming ")
```

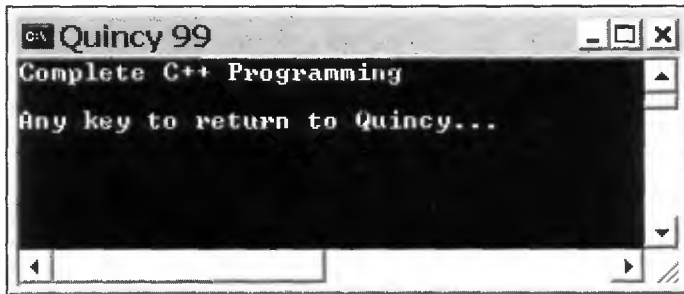
```

        if (s2 > s1)
        {
            string s3("Complete ");
            string s4 = s3 + s1 + s2;
            cout << s4 << endl;
        }

    return 0;
}

```

၂။ Ex1207.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၈) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂.၈)

## ၁၂.၂ The string Member Functions

၁။ standard C++ string class မှာ member function (4) မျိုးပါရှိပါတယ်။ (၁) clear( ) (၂) empty( ) (၃) length( ) နဲ့ (၄) data( ) function တို့ဖြစ်ကြပါတယ်။ clear( ) function ဟာ string object တွေကို zero length တန်ဖိုးဖြစ်အောင် clear လုပ်ပေးနိုင်ပါတယ်။ empty( ) function ကတော့ string object တစ်ခုဟာ empty ဖြစ်မဖြစ်ကို check လုပ်ပေးပါမယ်။ empty ဖြစ်ရင် true bool data ကို return လုပ်ပေးမှာပါ။ empty မဟုတ်ရင် false ပေါ့။ length( ) function က string object မှာပါဝင်တဲ့ character အရေအတွက်ကို ရေတွက်ပေးမှာပါ။ string data buffer တစ်ခုကို point လုပ်နေတဲ့ const pointer တစ်ခုကို return လုပ်ပေးချင်ရင် data( ) member function ကို အသုံးပြုလို့ရပါတယ်။ function တွေ အသုံးပြုနည်းကို Ex1208.cpp program မှာဖော်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 12.8: Using the string member functions

```
#include <iostream>
```

```
#include <string>
```

```
void test(const string& str)
{
    if (str.empty( ))
        cout << "\n\tThe string is empty\n";
    else
    {
        int len = str.length( );

        cout << "\n\tThe string has "
              << len << " characters.\n\t\""
              << str << "\"\n";
    }
}
```

```
int main( )
{
    string str;
    test(str);

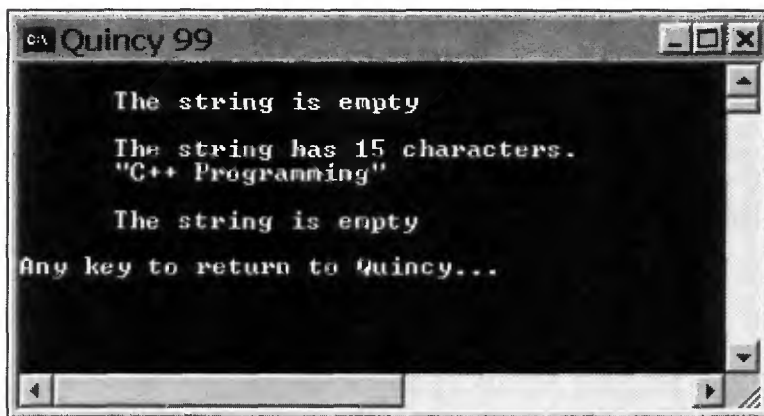
    str = "C++ Programming";
    test(str);

    str.clear( );
    test(str);

    return 0;
}
```

၂။ Ex1208.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၉) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။





ပုံ (၁၂.၉)

## ၁၂.၃ Formatted Output

### The `ios::width( )` Function

၁၂.၃ C++ program တစ်ခုကို run လိုက်ပြီး output ကို fixed column width တစ်ခုထဲမှာ ပေါ်လာစေ ချင်ရင် `width( )` member function ကိုအသုံးပြုရပါမယ်။ Ex1209.cpp program မှာ array `x[ ]` value တွေကို column width (10) ရှိတဲ့ column ထဲမှာ fixed format သို့မဟုတ် scientific format နဲ့ right-align လုပ်ပြီး display လုပ်ပြခိုင်းထားပါတယ်။

// Listing 12.9: Using `width( )` member function  
#include <iostream>

```
int main( )
{
    static double x[ ] =
        { 0.00000012, 1.23, 345.678, 56789012.34 };

    cout.setf(ios::fixed, ios::scientific );
    for (int i = 0; i < 4; i++)
```

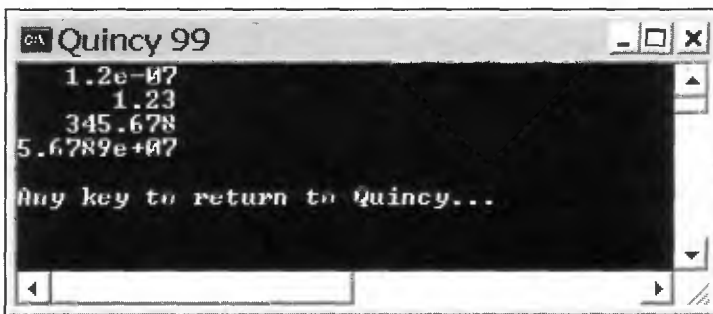
```

{
    cout.width(10);
    cout << x[i] << endl;
}

return 0;
}

```

၂။ Ex1209.cpp program ကို run လိုက်မယ်ဆိုရင်ပုံ (၁၂. ၁၀) မှာပြထားတဲ့အတိုင်းမြင်ရပါလိမ့်မယ်။ argument value ဟာ အလွန်ငယ်မယ် သို့မဟုတ် သိပ်ကြီးတဲ့အခါကျရင် scientific format နဲ့ display လုပ်ပြမှာပါ။ argument value ကပုံမှန်ဆိုရင် column width ထက်မကျော်တဲ့အခါမှာ fixed format နဲ့ display လုပ်ပြပါလိမ့်မယ်။ column width ထက်ကျော်သွားရင်တော့ argument value ကို round (6 decimal places) လုပ်ပေးမှာပါ။



ပုံ (၁၂. ၁၀)

## The setw( ) Manipulator

၁။ data display လုပ်တာကို table form အနေနဲ့ဖော်ပြချင်ရင် setw( ) manipulator ကိုအသုံးပြုပြီး column width ကိုတစ်နေရာချင်း adjust လုပ်ပေးလို့ရပါတယ်။ setw( ) function ကိုအသုံးပြုနိုင်ဖို့အတွက် <iomanip> header ကို program မှာ include လုပ်ပေးရပါမယ်။ cout.setf(ios::fixed) ကိုအသုံးပြုခဲ့ရင် argument value ဟာ သိပ်ငယ်နေတဲ့အခါမှာ သုညတွေပဲဖော်ပြပေးပါလိမ့်မယ်။ Ex1210.cpp program ကို လေ့လာကြည့်ပါ။

// Listing 12.10: The setw( ) manipulator

```
#include <iostream>
```

```
#include <iomanip>
```

```
int main( )
```

```
{
```

```
    static double x[ ] =
```

```
        { 0.00000012, 1.23, 345.678, 56789012.34 };
```

```
    static char *ch[ ] =
```

```
        {"ZARNI", "ARKAR", "AYELWIN", "POZAW"};
```

```
    cout.setf(ios::fixed);
```

```
    cout << endl;
```

```
    for (int i = 0; i < 4; i++)
```

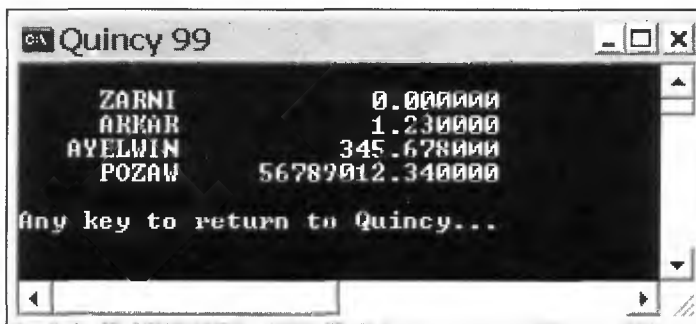
```
        cout << setw(10) << ch[i]
```

```
            << setw(20) << x[i] << endl;
```

```
    return 0;
```

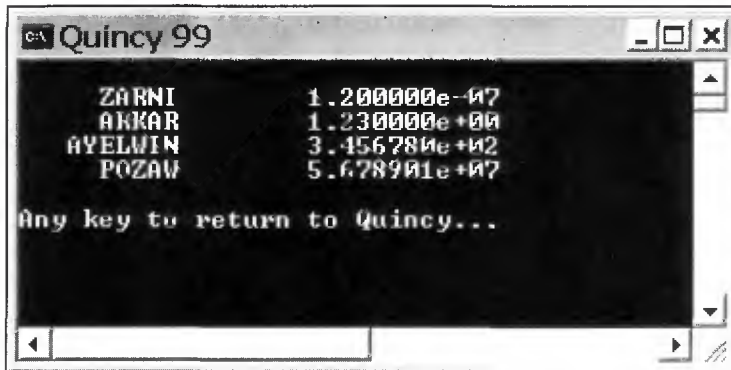
```
}
```

၂။ Ex1210.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၁) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၁၁)

၃။ cout.setf(ios::fixed) အစား cout.setf(ios::scientific) ကိုအသုံးပြုမယ်ဆိုရင် ပုံ (၁၂. ၁၂) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၁၂.၁၂)

## The ios::fill( ) Function

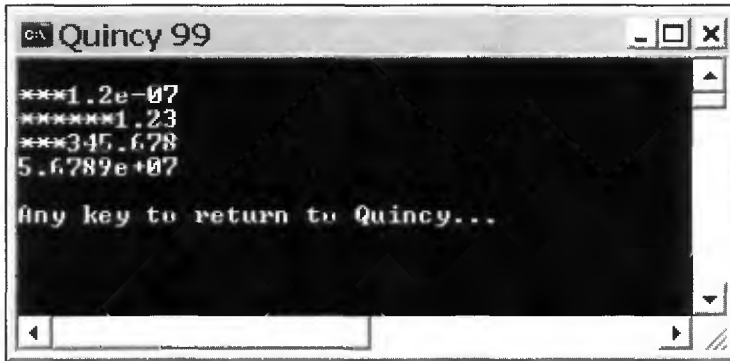
၁။ program မှာ fill( ) function ကိုအသုံးပြုမယ်ဆိုရင် output မှာ display လုပ်ပြမယ့် value တွေရဲ့ ရှေ့ကလွတ်တဲ့ နေရာတွေမှာ fill( ) function argument နဲ့အစားထိုး ပြောင်းပေးသွားမှာဖြစ်ပါတယ်။ Ex1211.cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 12.11: The fill( ) function  
#include <iostream>

```
int main( )
{
    static double x[ ] =
        { 0.00000012, 1.23, 345.678, 56789012.34 };

    cout << endl;
    for (int i = 0; i < 4; i++)
    {
        cout.width(10);
        cout.fill('*');
        cout << x[i] << endl;
    }
    return 0;
}
```

၂။ Ex1211.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၃) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၁၃)

## Output Justification

၁။ ပုံ (၁၂. ၁၂) မှာပြထားတဲ့ program output မှာနာမည်တွေကို left-justified လုပ်ပေးချင်ရင် `setiosflags (ios::left)` manipulator ကို ထည့်ရေးပေးရင်ရပါတယ်။ Ex1212.cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 12.12: The output justification

```
#include <iostream>
```

```
#include <iomanip>
```

```
int main( )
```

```
{
```

```
    static double x[ ] =
```

```
        { 0.00000012, 1.23, 345.678, 56789012.34 };
```

```
    static char* ch[ ] =
```

```
        {"ZARNI", "ARKAR", "AYELWIN", "POZAW"};
```

```
    cout << endl;
```

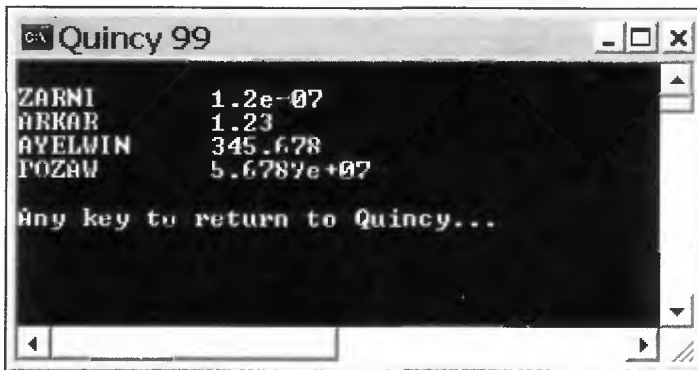
```
    for (int i = 0; i < 4; i++)
```

```

{
    cout << setiosflags(ios::left)
         << setw(12) << ch[i]
         << setw(15) << x[i] << endl;
}
return 0;
}

```

၂။ Ex1212.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၄) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၁၄)

## The setprecision Manipulator

၁။ ပုံ (၁၂. ၁၄) မှာပြထားတဲ့ program output မှာ ဒုတိယကော်လံက value တွေကို right-justified ၊ setprecision (1 decimal place) နဲ့ဖော်ပြပေးချင်ရင် Ex1213.cpp program အတိုင်း ရေးပေးရပါမယ်။

// Listing 12.13: The setprecision manipulator

```
#include <iostream>
```

```
#include <iomanip>
```

```
int main( )
```

```
{
```

```
    static double x[ ] = { 0.00000012, 1.23, 345.678, 56789012.34 };
```

```

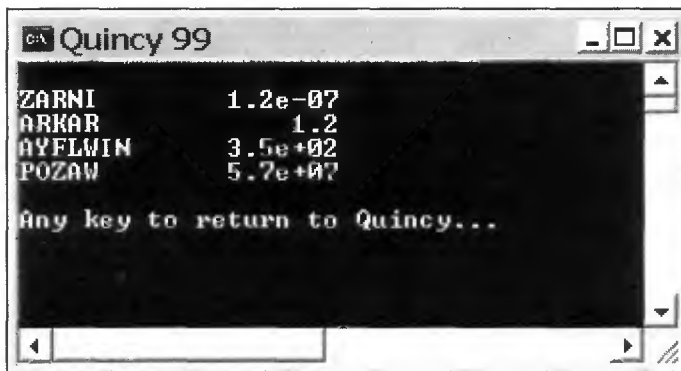
static char* ch[ ] =
    {"ZARNI", "ARKAR", "AYELWIN", "POZAW"};

cout << endl;
for (int i = 0; i < 4; i++)
{
    cout << setiosflags(ios::left) << setw(10)
        << ch[i]
        << setiosflags(ios::right)
        << setw(10) << setprecision(2)
        << x[i] << endl;
}

return 0;
}

```

၂။ Ex1213.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၅) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၁၅)

## Aligning Decimal Points

၁။ ပုံ (၁၂. ၁၅) မှာဖော်ပြထားတဲ့ program output မှာ ဒုတိယကော်လံက floating-point value တွေကို decimal-aligned လုပ်ပေးချင်ရင် Ex1214.cpp program မှာ ရေးထားတဲ့အတိုင်းရေးပါ။

// Listing 12.14: Scientific and fixed notation

```
#include <iostream>
```

```
#include <iomanip>
```

```
int main( )
```

```
{
```

```
    static double x[ ] =
```

```
        { 0.00000012, 1.23, 345.678, 56789012.34 };
```

```
    static char* ch[ ] =
```

```
        {"ZARNI", "ARKAR", "AYELWIN", "POZAW"};
```

```
    cout << endl;
```

```
    for (int i = 0; i < 4; i++)
```

```
    {
```

```
        cout << setiosflags(ios::left) << setw(10)
```

```
            << ch[i]
```

```
            << resetiosflags(ios::left) << setiosflags(ios::fixed)
```

```
            << setiosflags(ios::right) << setw(12)
```

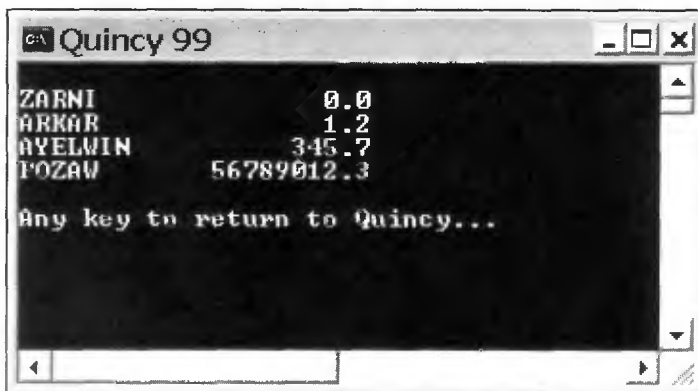
```
            << setprecision(1) << x[i] << endl;
```

```
    }
```

```
    return 0;
```

```
}
```

Ex1214.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၆) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၁၆)



# The Table of Square Roots and Squares

၁။ Ex1215.cpp program ဟာဆိုရင် 2 ကနေ 10 အထိ ဂဏန်း (9) လုံးကို square root နဲ့ square တွေရှာပြီး ဇယားပုံစံနဲ့ display လုပ်ပြခိုင်းတဲ့ program ဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

```
// Listing 12.15: A table of square roots and squares
```

```
#include <iostream>
```

```
#include <iomanip>
```

```
#include <cmath>
```

```
int main( )
```

```
{
```

```
    double x;
```

```
    cout << "\n\t X\t\t sqrt(X)\t\t X*X\n\n";
```

```
    for (x = 2; x <= 10; x++)
```

```
    {
```

```
        cout << resetiosflags(ios::left)
```

```
            << setw(12) << setprecision(1)
```

```
            << x
```

```
            << setiosflags(ios::fixed)
```

```
            << setw(12) << setprecision(4)
```

```
            << sqrt(x)
```

```
            << setiosflags(ios::fixed)
```

```
            << setw(12) << setprecision(1)
```

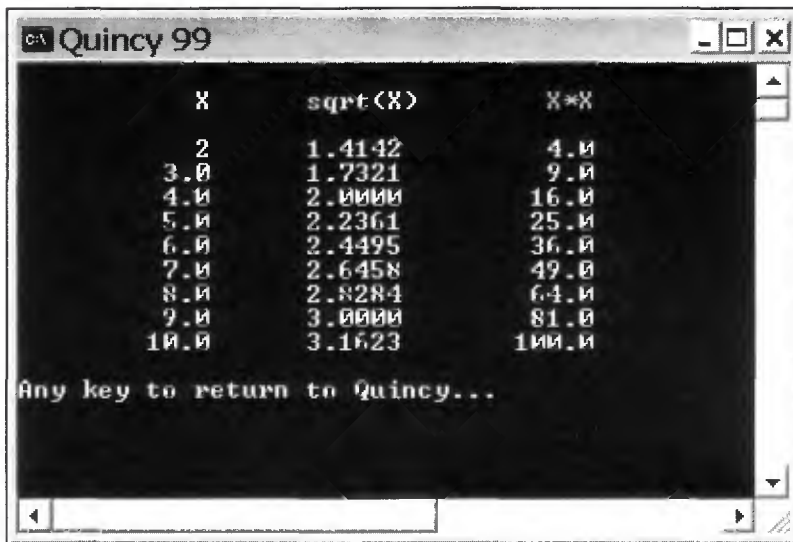
```
            << x*x << endl;
```

```
    }
```

```
    return 0;
```

```
}
```

၂။ Ex1215.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၇) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂.၁၇)

## ၁၂.၃ Output Member Functions

၁။ `std::ostream` class ထဲမှာ single character ကို display လုပ်ပြတဲ့ `put( )` member function နဲ့ output object stream တွေကို display လုပ်ပြမယ့် overloaded `<<` insertion operator တို့ပါဝင်ကြပါတယ်။ Ex1216.cpp program ကိုလေ့လာကြည့်ပါ။

```
// Listing 12.16: Output member function
#include <iostream>
```

```
int main( )
{
    cout.put('C');
    cout.put('+');
    cout.put('+');
    cout << endl;
```

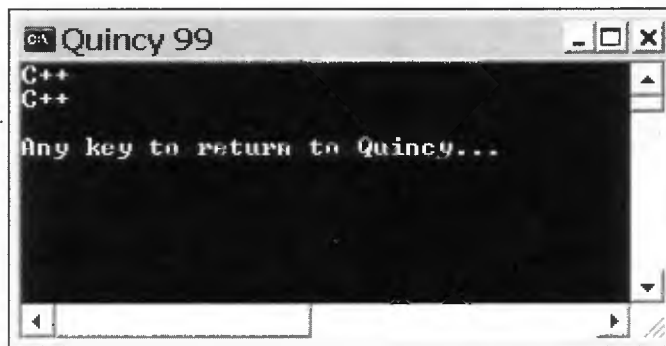
```

cout << 'C';
cout << '+';
cout << '+';
cout << endl;

return 0;
}

```

၂။ Ex1216.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၈) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၁၈)

၃။ write( ) member function တစ်ခုဟာ output data ကို binary format အနေနဲ့ stream ထဲ ကိုပို့ပေးပါတယ်။ Ex1217.cpp program ကို run လိုက်ရင် message ကို display အရင်လုပ်ပြီး alarm မြည်သံကိုကြားရမှာပါ။ နောက်ပြီး next line ကို အလိုလိုကူးသွားပါလိမ့်မယ်။

```

// Listing 12.17: Output member function
#include <iostream>

```

```

int main( )
{
    static struct
    {
        char ch[40];
        int alarm;
        int eol;
    }
}

```

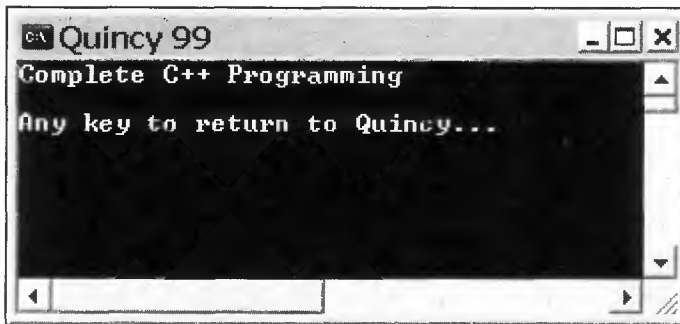
```

} msg = {"Complete C++ Programming", '\a', '\n'};

cout.write(reinterpret_cast <char*>(&msg), sizeof msg);
return 0;
}

```

၂။ Ex1217.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၁၉) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၁၉)

## ၁၂.၄ Input Member Functions

၁။ C++ program တွေမှာ >> extraction operator ကိုအသုံးပြုပြီး data input လုပ်မယ်ဆိုလို့ရှိရင် အားနည်းချက်တစ်ခုက data input မှာ white space တွေပါလာရင် data လို့မယူဆပဲ ကျော်သွားပါလိမ့်မယ်။ အဲဒီလိုမကျော်စေချင်ဘူးဆိုရင် get( ) member function ကိုအသုံးပြုရပါမယ်။ Ex1218.cpp program မှာ >> extraction operator နဲ့ get( ) function တို့ကိုနှိုင်းယှဉ်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

```

// Listing 12.18: Input member function
#include <iostream>

```

```

int main( )
{

```

```

char line[25], ch = 0, *lp;
cout << "\n Type a line terminated by 'x'" << "\n > ";
lp = line;

while (ch != 'x')
{
    cin >> ch;    *lp++ = ch;
}
*lp = '\0';
cout << ' ' << line;
cout << "\n\n Type another one\n > ";
lp = line;
ch = 0;
while (ch != 'x')
{
    cin.get(ch);    *lp++ = ch;
}
*lp = '\0';
cout << line << endl;;
return 0;
}

```

Ex1218.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၂၀) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။

```

C:\ Quincy 99
Type a line terminated by 'x'
> C++ PROGRAMMING x
C++PROGRAMMINGx

Type another one
> C++ PROGRAMMING x
C++ PROGRAMMING x
Any key to return to Quincy...

```

ပုံ (၁၂.၂၀)

၃။ တစ်ကယ်လို့ C++ program ကို run လိုက်ပြီး input data ထည့် ၊ ENTER နှိပ်လိုက်တာနဲ့ output display လုပ်ပြစေချင်ရင် Ex1219.cpp program မှာ ရေးထားတဲ့အတိုင်းရေးရပါမယ်။

```
// Listing 12.19: Input member function
#include <iostream>
```

```
int main( )
{
    char line[40];

    cout << "\n Type a line terminated by carriage return\n > ";
    cin.get(line, 40);
    cout << "\n " << line << endl;

    return 0;
}
```

၄။ နောက်တစ်မျိုးက input data မှာ password တစ်ခုကိုရိုက်ထည့်ပေးမှ output display လုပ်ပြတာမျိုး ဖြစ်စေချင်ရင် Ex1220.cpp program မှာပြထားတဲ့အတိုင်းရေးပါ။

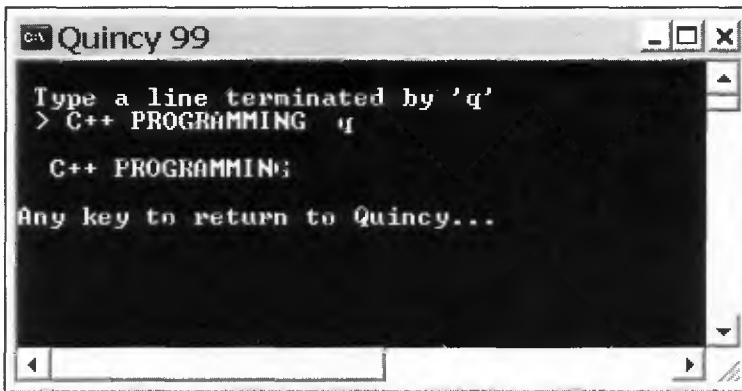
```
// Listing 12.20: Input member function
#include <iostream>
```

```
int main( )
{
    char line[40];

    cout << "\n Type a line terminated by 'q'\n > ";
    cin.getline(line, 25, 'q');
    cout << " " << line << endl;

    return 0;
}
```

Ex1220.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၂၁) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂. ၂၁)

## More on the get( ) Function

// Listing 12.21: More on the get( ) function

```
#include <iostream>
```

```
int main( )
```

```
{
```

```
    char ch;
```

```
    while ((ch = cin.get( )) != EOF)
```

```
        cout << "ch = " << ch << endl;
```

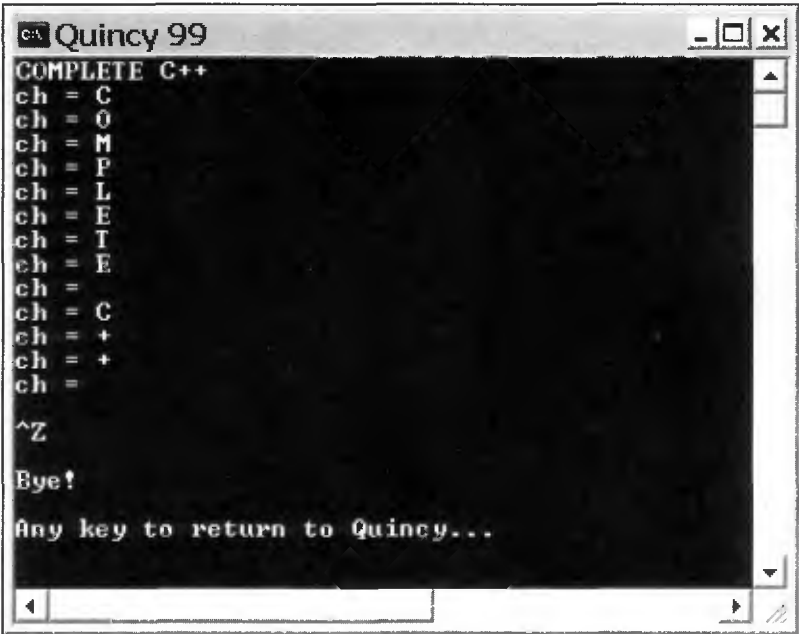
```
    cout << endl << "Bye!" << endl;
```

```
    return 0;
```

```
}
```

Ex1221.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၂၂) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ ဒီ program ကိုလေ့လာကြည့်ရင် cin.get( ) ကိုအသုံးပြုပြီး string object အတွက် character တစ်လုံးချင်းကို display လုပ်ပြသွားပါတယ်။ ENTER key ကိုနှိပ်တဲ့အတွက် ch = <blank> ဟာ ဆက်ပေါ်လာတာပါ။ data

input ကိုလက်ခံပါသေးတယ်။ CTRL + Z +ENTER key တွေကိုနှိပ်ရင်တော့ Bye! ဆိုတဲ့ message ကို display လုပ်ပြပြီး program stop ဖြစ်သွားပါပြီ။



ပုံ (၁၂.၂၂)

## Using the get( ) Function with Parameters

// Listing 12.22: Using the get( ) function with parameters  
#include <iostream>

```
int main( )  
{  
    char  ch1, ch2, ch3;  
  
    cout << "\n\tEnter three letters : ";  
    cin.get(ch1).get(ch2).get(ch3);
```



```

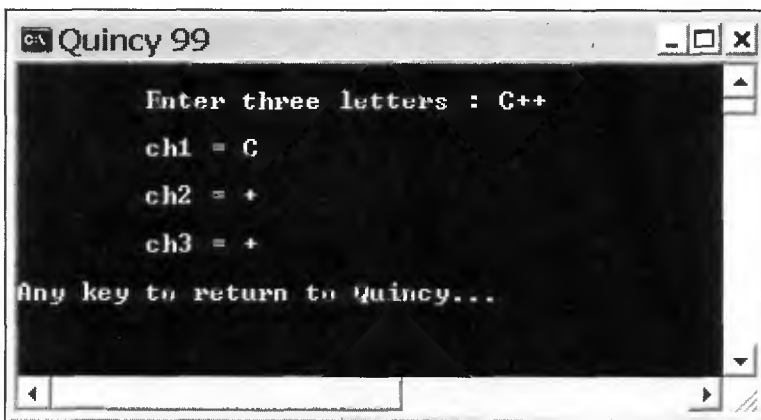
cout << "\n\tch1 = " << ch1 << endl
    << "\n\tch2 = " << ch2 << endl
    << "\n\tch3 = " << ch3 << endl;

return 0;

}

```

Ex1222.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂.၂၃) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၂.၂၃)

## Using the peek( ), putback( ), and ignore( ) Functions

// Listing 12.23: Using peek( ), putback( ), and ignore( ) functions  
#include <iostream>

```

int main( )
{
    char ch;
    cout << "Enter a sentence\n";

```

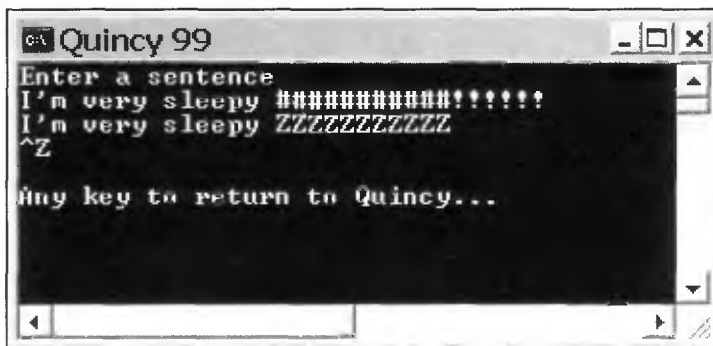
```

while (cin.get(ch))
{
    if (ch == '#')
        cin.putback('Z');
    else
        cout << ch ;

    while (cin.peek( ) == '!')
        cin.ignore(1, '!');
}
return 0;
}

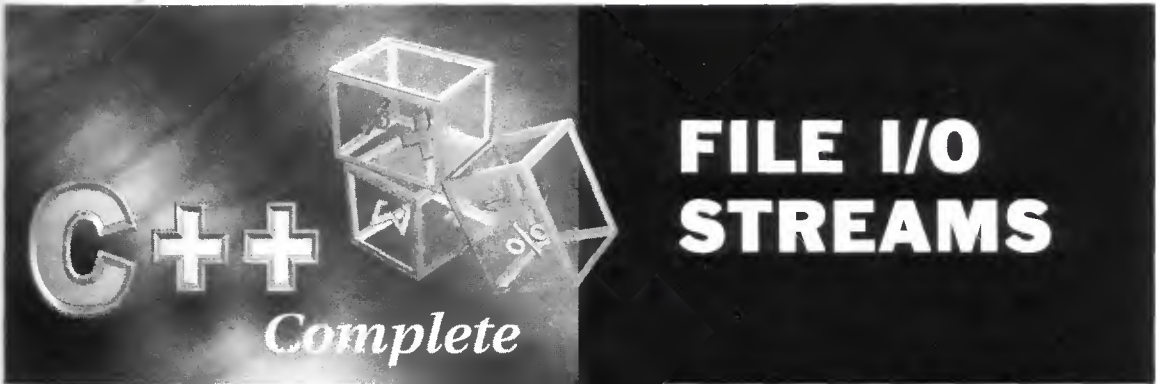
```

Ex1223.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၂. ၂၄) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၁၂. ၂၄)

# Chapter 13



data တွေကို disk မှိုင်းတွေကနေဖတ်ယူမယ် သို့မဟုတ် disk မှိုင်းတွေပေါ်ကို ရေးထည့်ပေးချင်တယ် ဆိုလို့ရှိရင် C++ input/ output system ကိုနားလည်ဖို့လိုပါတယ်။ standard C++ library မှာ disk မှိုင်း တွေကို manage လုပ်ဖို့အတွက် I/O class တွေအများအပြားပါရှိပါတယ်။ C++ ရဲ့ I/O operation တွေမှာ အဓိကအကျဆုံးကတော့ stream ပါပဲ။ stream ဆိုတာ data flow တစ်ခုကို အလွယ်ကူဆုံး ခေါ်ဝေါ်တဲ့အမည် တစ်ခုဖြစ်ပါတယ်။ အမျိုးအစားမတူတဲ့ stream တွေဟာဆိုရင် ကွဲပြားခြားနားတဲ့ data flow အလုပ်တွေကိုလုပ်ပေး ကြမှာပါ။ stream တိုင်းမှာသက်ဆိုင်ရာ class တွေရှိနေပါတယ်။ အဲဒီ class တွေထဲက member function တွေ၊ definition တွေဟာသက်ဆိုင်ရာ stream တွေကိုထိန်းချုပ်ပေးသွားမှာဖြစ်ပါတယ်။ ဥပမာ ifstream class object ဟာ input disk မှိုင်းတွေမှာ write လုပ်ဖို့အတွက်အသုံးပြုပါတယ်။ extraction >> operator က istream class ရဲ့ member တစ်ခုဖြစ်သလို insertion << operator ဟာ ostream class ရဲ့ member တစ်ခုဖြစ်ပါတယ်။ istream နဲ့ ostream class တွေဟာလည်း ios class ကနေ derive လုပ်လာတဲ့ derived class တွေပဲလေ။ မှိုင်း input/ output အလုပ်တွေမှာ ကျွန်တော်တို့စိတ်ဝင်စားဆုံး class တွေက input မှိုင်းတွေ အတွက်ဆိုရင် ifstream နဲ့ output မှိုင်းတွေအတွက် ofstream class object တွေဖြစ်ပါတယ်။ ဒါပေမယ့် <ofstream> နဲ့ <ifstream> class object တို့ဟာ <fstream> header မှိုင်းတစ်ခုတည်း အသုံးပြုတာနဲ့ ရပါတယ်။

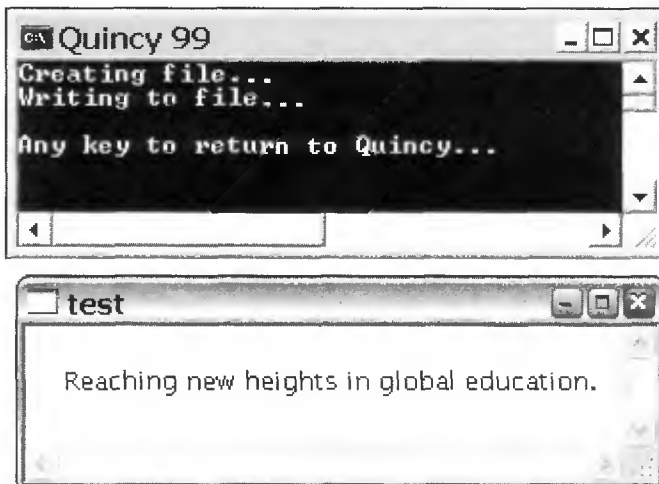
## ၁၃.၁ The fstream Class

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1301.cpp program ဟာ test.txt ဆိုတဲ့ output ဖိုင်တစ်ခုမှာ string တွေ write လုပ်ပေးတဲ့ program ဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

```
// Listing 13.1: Writing strings to a file
#include <fstream>
```

```
int main( )
{
    cout << "Creating file...\n" ;
    ofstream outfile("test.txt");
    cout << "Writing to file...\n" ;
    outfile << "\n\tReaching new heights in global education. \n";
    return 0;
}
```

၂။ Ex1301.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် outfile လို့ခေါ်တဲ့ ofstream class object instance တစ်ခုကို construct လုပ်ပြီး တစ်ချိန်တည်းမှာပဲ ဒီ object ကို test.txt ဖိုင်နဲ့ initialize လုပ်ပေးထားတာကိုကြည့်ပါ။ ဒီ program ကို run လိုက်တဲ့အခါကျရင် output က screen မှာမပေါ်ပါဘူး။ test.txt ဖိုင်မှာပေါ်နေပါလိမ့်မယ်။ ပုံ (၁၃. ၁) ကိုကြည့်ပါ။



ပုံ (၁၃. ၁)

## ၁၃.၂ Appending to an Output File

၁။ ပုံ (၁၃. ၁) မှာဖော်ပြထားတဲ့ text.txt output မိုင်မှာ string တစ်ချို့ကိုတိုးပေးချင်ရင် ios::app ကို ရေးထည့်ပေးရပါတယ်။ Ex1302.cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 13.2: Appending to an output file

```
#include <fstream>
```

```
int main( )
```

```
{
```

```
    cout << "Opening file...\n" ;
```

```
    ofstream outfile("test.txt", ios::app);
```

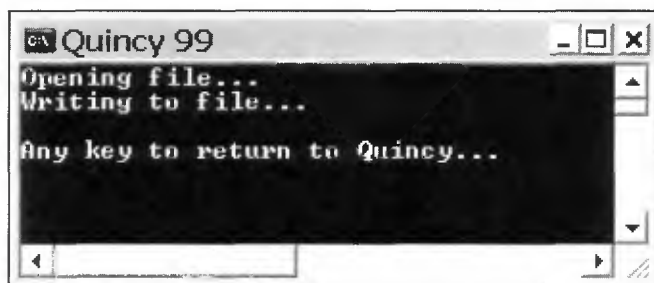
```
    cout << "Writing to file...\n" ;
```

```
    outfile << "\n\t.... here more to come!";
```

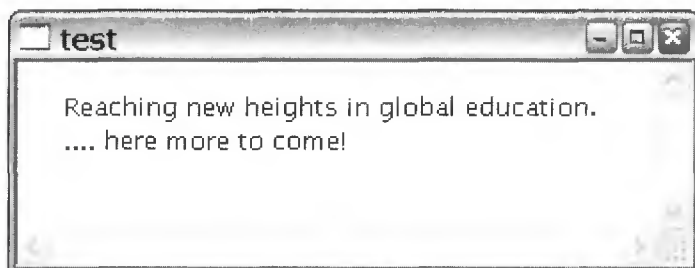
```
    return 0;
```

```
}
```

၂။ Ex1302.cpp program ကို run လိုက်တဲ့အခါကျရင် output က screen မှာမပေါ်ပါဘူး။ test.txt မိုင်မှာပဲ string တစ်ကြောင်း တိုးနေတာကိုတွေ့ရပါလိမ့်မယ်။ ပုံ (၁၃. ၂) ကိုကြည့်ပါ။



ပုံ (၁၃. ၂)



## ၁၃.၃ Avoiding to Open an Existing File

၁။ Ex1303.cpp program ဟာဆိုရင် ရှိပြီးသားဖိုင်တစ်ခုကို မှားဖွင့်မိပြီး override မလုပ်မိအောင် fail( ) member function ကနေ ကာကွယ်ပေးတဲ့နည်းကို ရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

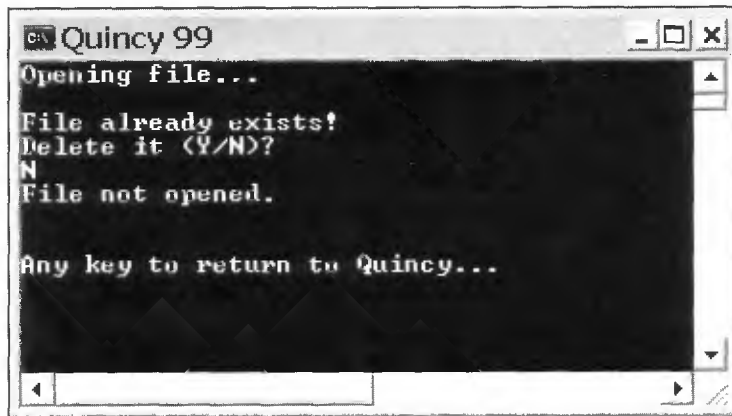
// Listing 13.3: Avoiding opening an existing file

```
#include <fstream>
```

```
int main( )
{
    cout << "Opening file...\n" ;
    ofstream outfile("test.txt", ios::noreplace);
    if (outfile.fail( ))
    {
        cout << endl << "File already exists!\n"
            << "Delete it (Y/N)?" << endl;

        char ch;
        cin >> ch;
        if ((ch == 'Y') || (ch == 'y'))
        {
            ofstream outfile("test.txt");
            outfile << "These are test data.\n"
                << "New file created.\n";
        }
        else
            cout << "File not opened.\n";
    }
    cout << endl;
    return 0;
}
```

၂။ Ex1303.cpp program ကို ပုံ (၁၃. ၃) မှာ run ပြထားပါတယ်။ fail( ) member function ကနေ message prompt လုပ်ပေးတဲ့အခါမှာ N ကိုနှိပ်တဲ့အတွက် test.txt ဖိုင်ဟာ override အဖြစ်မခံရတော့ပါဘူး။



ပုံ (၁၃.၃)

## ၁၃.၄ The ofstream( ) Function

၁။ Ex1304.cpp program တာဆိုရင် file pointer `tellp( )` နဲ့ `put( )` function တို့ကိုအသုံးပြုပြီး ရှိပြီးသားဖိုင်တစ်ခုကို input data အသစ်နဲ့ override လုပ်ပေးပါတယ်။ screen ပေါ်မှာလည်း inputdata တစ်လုံးချင်းပေါ်လာအောင် display လုပ်ပြမှာပါ။ program ရေးထားပုံကို လေ့လာကြည့်ပါ။

```
// Listing 13.4: Using ofstream( ) member function
#include <fstream>
#include <string>
```

```
int main( )
{
    string str("This is a test");

    // Create an output stream object
    ofstream outfile;

    // Associate a file with the stream
    outfile.open("test.txt");
```

```

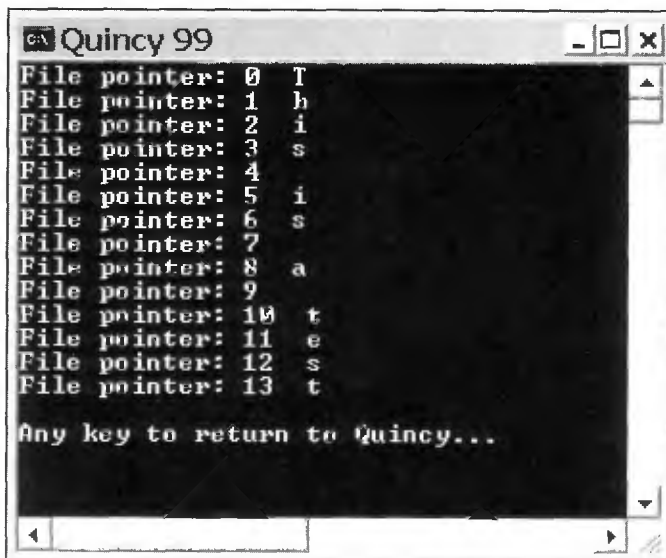
// Write a string one character at a time
for (int x=0; x<14; ++x)
{
    cout << "File pointer: " << outfile.tellp( );
    outfile.put(str[x]);
    cout << " " << str[x] << endl;
}

// Close up the file.
outfile.close( );

return 0;
}

```

၂။ Ex1304.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၃.၄) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။ file pointer ကနေညွှန်တဲ့ input data တစ်ခုချင်းကို screen မှာ display လုပ်ပြနေပါပြီ။ test.txt ဖိုင်ကိုဖွင့်ကြည့်ရင် မူလကရှိတဲ့ information တွေကို This is a test ဆိုတဲ့စာကြောင်းက override လုပ်သွားတာကိုတွေ့ရမှာပါ။



ပုံ (၁၃.၄)



## ၁၃.၅ Reading Strings from a File

၁။ Ex1304.cpp program ကို run လိုက်လို့ရတဲ့ test.txt ဖိုင်က data တွေကို read ပြန်လုပ်ချင်ရင် အခုလိုလုပ်လို့ရပါတယ်။ အောက်မှာရေးပြထားတဲ့ Ex1305.cpp ကိုလေ့လာကြည့်ပါ။

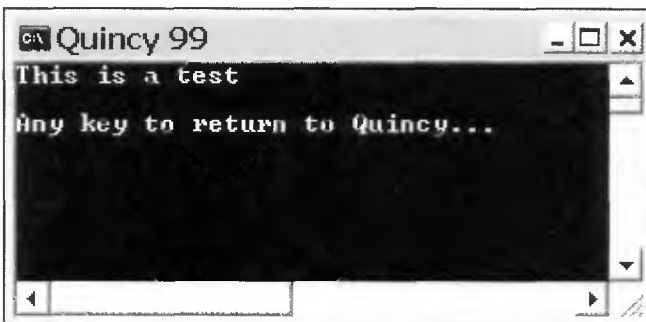
```
// Listing 13.5: Reading strings from a file
#include <fstream>
```

```
int main( )
{
    const int MAX = 80;
    char    buffer[MAX];

    ifstream infile("test.txt");
    while (infile)
    {
        infile.getline (buffer,MAX);
        cout << buffer;

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

၂။ Ex1305cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၃. ၅) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ test.txt ဖိုင်က data ကို program ကဖတ်ယူပြီး screen မှာ display လုပ်ပြတာပါပဲ။



ပုံ (၁၃. ၅)

## ၁၃.၆ Reading until End-of-File

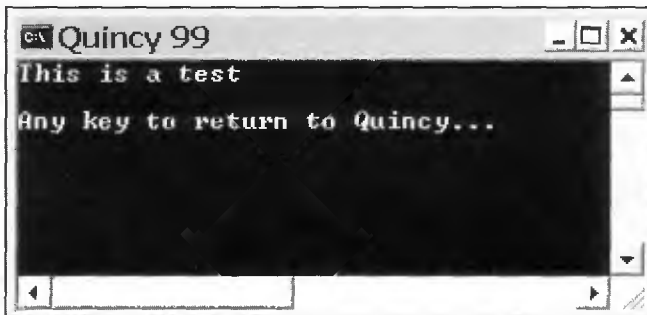
ဖိုင်တစ်ခုကို open လုပ်ပြီး ဖိုင်အဆုံးကိုရောက်တဲ့အထိ ဖတ်သွားချင်ရင် eof( ) member function ကိုအသုံးပြုရမှာပါ။ Ex1306.cpp program မှာ ဒီဥပမာကိုရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။ Ex1306.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၃.၆) မှာပြထားတဲ့အတိုင်းပဲ အဖြေရပါလိမ့်မယ်။

```
// Listing 13.6: Testing End-of-file
#include <fstream>
```

```
int main( )
{
    ifstream outfile("test.txt");

    while (!outfile.eof())
    {
        char ch;

        outfile.get(ch);
        if (!outfile.eof( ))
            cout << ch;
    }
    cout << endl;
    return 0;
}
```



ပုံ (၁၃.၆)

## ၁၃.၇ The seekg( ) Member Function

ဖိုင်တစ်ခုကို open လုပ်ပြီးဖိုင်ထဲက information တွေကို user ကြိုက်တဲ့နေရာကနေ စဖတ်ပြီးတော့ screen မှာ display လုပ်ခိုင်းချင်ရင် seekg( ) member function ကိုအသုံးပြုရပါမယ်။ Ex1307.cpp program ကိုလေ့လာကြည့်ပါ။ ဒီ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၃.၇) မှာပြထားတဲ့အတိုင်းမြင်ရပါမယ်။

// Listing 13.7: Seeking within a file

```
#include <fstream>
```

```
int main( )
```

```
{
```

```
    ifstream outfile("test.txt");
```

```
    outfile.seekg(4);
```

```
    while (!outfile.eof())
```

```
    {
```

```
        char ch;
```

```
        outfile.get(ch);
```

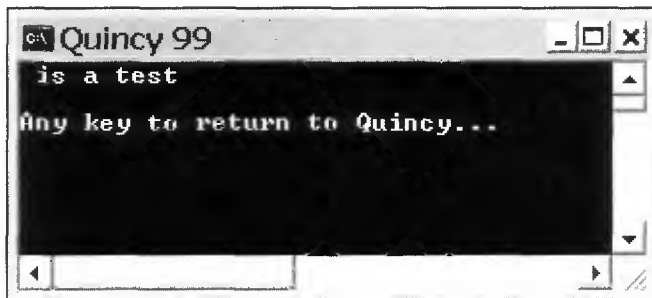
```
        if (!outfile.eof( )) cout << ch;
```

```
    }
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၃.၇)

## ၁၃.၈ The tellg( ) Member Function

၁။ ဖိုင်တစ်ခုထဲက information မှာ ကျွန်တော်တို့ရှာချင်တဲ့ဥစ္စာတစ်ခု ဥပမာ white space နေရာတွေကိုရှာပေးချင်ရင် tellg( ) member function ကိုအသုံးပြုရပါမယ်။ Ex1309.cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 13.8: The tellg( ) member function

```
#include <fstream>
```

```
int main( )
```

```
{
```

```
    ifstream f1("test.txt");
```

```
    while (!f1.eof( ))
```

```
    {
```

```
        char c1;
```

```
        f1.get(c1);
```

```
        if (!f1.eof( ))    cout << c1;
```

```
    }
```

```
    cout << endl << endl;
```

```
    ifstream f2("test.txt");
```

```
    while (!f2.eof( ))
```

```
    {
```

```
        char c2;
```

```
        streampos here = f2.tellg();
```

```
        f2.get(c2);
```

```
        if (c2 == ' ')
```

```
            cout << "Position " << here << " is a space\n";
```

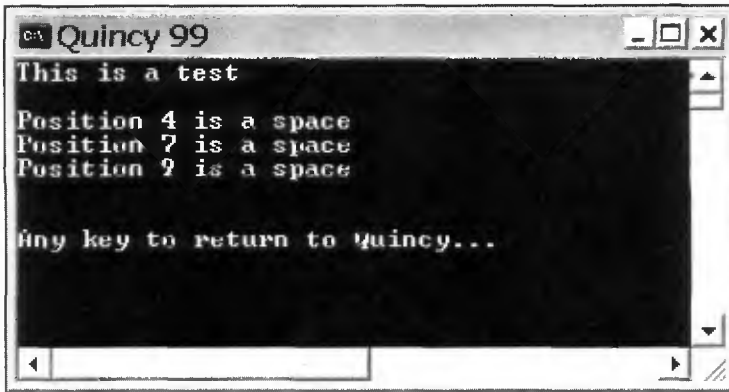
```
    }
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1308.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၃. ၈) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၃. ၈)

## ၁၃.၉ Read and Write a Stream File

၁။ တစ်ခါတစ်ရံမှာ program တစ်ခုအနေနဲ့ ဖိုင်တစ်ခုကိုဖွင့်တဲ့အခါကျရင် read/ write access ရရှိဖို့လိုပါတယ်။ ဥပမာ database ဖိုင်မျိုးတွေမှာ record တွေကိုဖတ်မယ်၊ ပြင်မယ်။ ဖိုင်မှာပြန်ရေးမယ်ဆိုတာမျိုး လုပ်ရမယ်မဟုတ်လား။ ဖိုင်တစ်ခုကို read/ write access ရအောင်လုပ်ပေးချင်ရင် program မှာ ios::openmode argument ကို ifstream constructor မှာရေးထည့်ပေးရပါမယ်။ ပြီးတော့ရင် ostream object instance တစ်ခုကို declare လုပ်ပြီး constructor ကို ifstream::rdbuf( ) member function ရဲ့ return value နဲ့ initialize လုပ်ပေးရမှာပါ။ ဒါဆိုရင် ostream object ကို ဖိုင် write လုပ်ဖို့အတွက် အသုံးပြုလို့ရပါပြီ။ ဒီဥပမာကို Ex1309.cpp program မှာရေးပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 13.9: Reading and writing a stream file

```
#include <fstream>
```

```
#include <cctype>
```

```
int main( )
```

```
{
```

```
    char*  fname = "test.txt";
```

```
    // Read the file into an array.
```

```
    ifstream  infile(fname, ios::in | ios::out | ios::binary);
```

```

ostream outfile(infile.rdbuf());
char ch[100];
int i = 0;

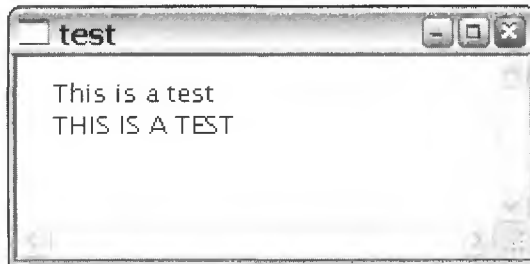
// Write the array from the file
while (!infile.eof() && i < sizeof ch)
    infile.get(ch[i++]);

// Write the array to the file
outfile.seekp(0, ios::end);
outfile << "\r\n";
for (int j = 0; j < i-1; j++)
    outfile.put(static_cast<char>(toupper(ch[j])));

return 0;
}

```

၂။ Ex1309.cpp program ကိုလေ့လာကြည့်ရင် စတင်ချင်းမှာ test.txt ဖိုင်ကို fname နဲ့ point လုပ်ခိုင်းထားပါတယ်။ နောက်ပြီးဖိုင်အတွက် read/write access ကို define လုပ်ပေးပါတယ်။ while (!infile.eof() && i < sizeof ch) infile.get(ch[i++]); statement ဟာ test.txt ဖိုင်ထဲက information ကို ch[ ] array ထဲဖတ်ထည့်ပေးပြီး အဲဒီ data တွေကို uppercase letter တွေဖြစ်အောင်ပြောင်းပါတယ်။ ပြီးတော့ရင် test.txt ဖိုင်ထဲက information မှာ updated data ကိုဆက်ရေးထည့်ပါတယ်။ Ex1309.cpp program ကို run လိုက်ရင် output က screen မှာမပေါ်ပါဘူး။ test.txt ဖိုင်မှာပဲ information တစ်ခု တိုးနေတာကိုတွေ့ရပါလိမ့်မယ်။ ပုံ (၁၃.၉) ကိုကြည့်ပါ။



ပုံ (၁၃.၉)

## ၁၃.၁၀ Opening and Closing a Stream File

၁။ C++ program တစ်ခုမှာ ifstream သို့မဟုတ် ofstream object တွေကို နာမည်မပါပဲနဲ့ declare လုပ်လို့ရပါတယ်။ အဲဒီလို declare လုပ်လိုက်ရင် object ကရှိနေပြီ။ ဒါပေမယ့် object နဲ့ဆက်သွယ်တဲ့ ဖိုင်တော့မရှိ သေးဘူးပေါ့။ ဖိုင်တစ်ခုခုနဲ့ဆက်သွယ်ပေးချင်ရင် open( ) member function ကိုအသုံးပြုရမှာပါ။ ဖိုင်နဲ့အဆက် အသွယ်ဖြတ်ချင်ရင်တော့ close( ) member function ကိုအသုံးပြုရပါမယ်။ Ex3010.cpp program မှာ outfile ဆိုတဲ့ ofstream object တစ်ခုကို declare လုပ်ပြီး test1.txt ၊ test2.txt ဖိုင် (2) ခုနဲ့ တစ်ခုချင်း ဆက်သွယ်ပြီးတော့ အလုပ်လုပ်သွားပုံကို တင်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 13.10: The open( ) and close( ) member functions

```
#include <fstream>
```

```
#include <cctype>
```

```
int main( )
```

```
{
```

```
    // An ofstream object without a file.
```

```
    ofstream outfile;
```

```
    cout << "Creating the test1.txt file...\n";
```

```
    outfile.open("test1.txt");
```

```
    outfile << "This is TEST1";
```

```
    outfile.close( );
```

```
    cout << "Creating the test2.txt file...\n";
```

```
    outfile.open("test2.txt");
```

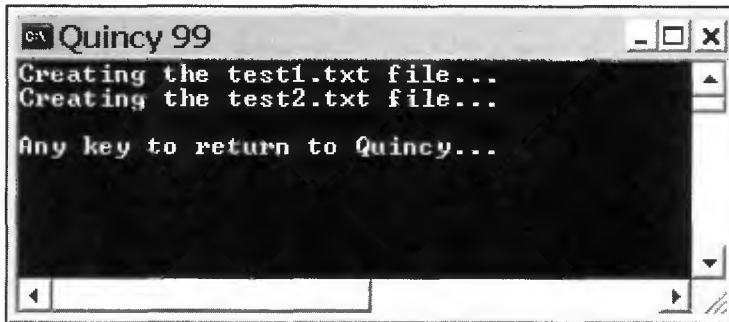
```
    outfile << "This is TEST2";
```

```
    outfile.close( );
```

```
    return 0;
```

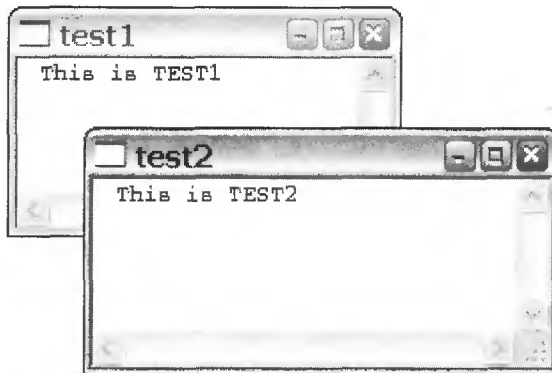
```
}
```

၂။ Ex1310.cpp program ကို run လိုက်ရင် output က screen မှာမပေါ်ပါဘူး။ prompt တွေပဲ display လုပ်ပြခိုင်းထားပါတယ်။ ပုံ (၁၃. ၁၀) ကိုကြည့်ပါ။



ပုံ (၁၃. ၁၀)

၃။ test1.txt နဲ့ test2.txt ဖိုင်တွေမှာ အခုလို write လုပ်ပြီးဖြစ်နေတာကိုတွေ့ရမှာပါ။ ပုံ (၁၃. ၁၁) ကိုကြည့်ပါ။



ပုံ (၁၃. ၁၁)

## ၁၃.၁၁ Objects I/O

၁။ ဒီတစ်ခါနောက်ဆုံး လေ့လာမှာက object တွေကို ဖိုင်တွေမှာဘယ်လို write လုပ်မယ် ၊ အဲဒီဖိုင်တွေကနေ ဘယ်လိုပြန်ဖတ်မလဲဆိုတဲ့နည်းတွေကိုလေ့လာမှာဖြစ်ပါတယ်။ Ex3011.cpp program ဟာ class history object တစ်ခုနဲ့ပတ်သက်တဲ့ data တွေကို history.dat ဖိုင်မှာ write အရင်လုပ်ပြီး screen မှာ display ပြခိုင်းတာဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။



// Listing 13.11: I/O with multiple objects

```
#include <fstream>
```

```
class history
```

```
{
```

```
    char name[30];
```

```
    char degree[30];
```

```
    int age;
```

```
    public:
```

```
    void getData( )
```

```
    {
```

```
        cin >> name >> age >> degree;
```

```
    }
```

```
    void showData( )
```

```
    {
```

```
        cout << "\nName = " << name << endl
```

```
            << "Age    = "    << age    << endl
```

```
            << "Degree = "    << degree << endl;
```

```
    }
```

```
};
```

```
int main( )
```

```
{
```

```
    char ch;
```

```
    history person;
```

```
    fstream file;
```

```
    file.open("history.dat", ios::app| ios::out| ios::in);
```

```
    do {
```

```
        person.getData( );
```

```
        file.write((char*) &person, sizeof person);
```

```
        cout << "\nEnter another person (y/n)? ";
```

```
        cin >> ch;
```

```
    } while (ch == 'y');
```

```
    file.seekg(0);
```

```

file.read((char*) &person, sizeof person);
while (!file.eof( ))
{
    cout << "\nPerson : ";
    person.showData( );
    file.read((char*) &person, sizeof person);
}
return 0;
}

```

၂။ Ex1311.cpp program ကို run လိုက်ပြီး data တွေကို history.dat မှာ write လုပ်ခိုင်းပြီး screen မှာ echo ပြန်လုပ်ပြခိုင်းထားပါတယ်။ ပုံ (၁၃.၁၂) ကိုကြည့်ပါ။

```

Quincy 99
ZARNI
25
B.Arch
Enter another person (y/n)? y
ARKAR
24
MA
Enter another person (y/n)? y
AYELWIN
40
PHD
Enter another person (y/n)? n
Person :
Name   = ZARNI
Age    = 25
Degree = B.Arch
Person :
Name   = ARKAR
Age    = 24
Degree = MA
Person :
Name   = AYELWIN
Age    = 40
Degree = PHD
Any key to return to Quincy...

```

ပုံ (၁၃.၁၂)

၃။ Ex3012.cpp program ဟာဆိုရင် history.dat ဖိုင်ထဲမှာပါတဲ့ person data တွေက ကျွန်တော်တို့ သိချင်တဲ့လူတစ်ယောက်ရဲ့ information ကိုလိုက်ရှာပြီး screen မှာ display လုပ်ပြခိုင်းတာဖြစ်ပါတယ်။

// Listing 13.12: Seeking a particular object in a file

```
#include <fstream>
```

```
class history
```

```
{
```

```
    char name[30];
```

```
    char degree[30];
```

```
    int age;
```

```
public:
```

```
    void showData( )
```

```
    {
```

```
        cout << "Name   = " << name << endl
```

```
        << "Age     = " << age << endl
```

```
        << "Degree  = " << degree << endl;
```

```
    }
```

```
};
```

```
int main( )
```

```
{
```

```
    history person;
```

```
    ifstream infile;
```

```
    infile.open("history.dat");
```

```
    infile.seekg(0, ios::end);
```

```
    int endpos = infile.tellg( );
```

```
    int n = endpos/sizeof (history);
```

```
    cout << "\nThere are " << n << " persons in file";
```

```
    cout << "\nEnter person number : ";
```

```
    cin >> n;
```

```

int pos = (n-1)*sizeof (history);

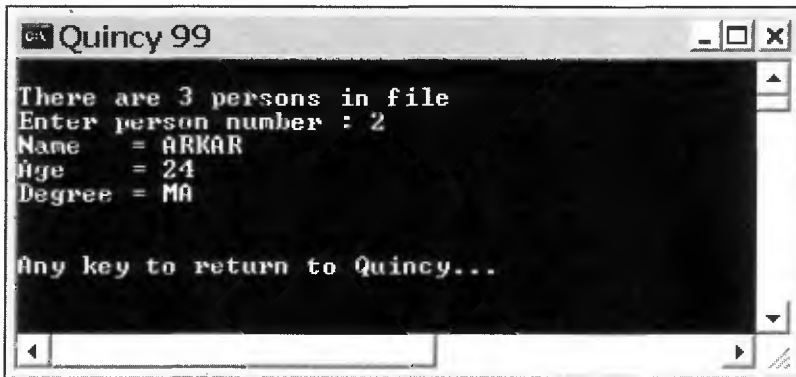
infile.seekg(pos);
infile.read((char*) &person, sizeof person);
person.showData( );
cout << endl;

return 0;

}

```

၄။ ပုံ (၁၃. ၁၃) တာဆိုရင် Ex1312.cpp program ကို run လိုက်ပြီး person number : 2 ကို history.dat မှိုင်းထဲမှာရှာခိုင်းပြီးတော့ information ကို screen မှာ display လုပ်ပြခိုင်းထားတာပါ။



ပုံ (၁၃. ၁၃)

# Chapter 14



sequence ဆိုတာ Standard Template Library (STL) မှာပါဝင်တဲ့ container class template တစ်ခုဖြစ်ပြီး အမျိုးအစားတူညီတဲ့ object အရေအတွက်တစ်ခုကို linear organization ပုံစံနဲ့ သူ့ထဲမှာ store လုပ်ထားပါတယ်။ ဥပမာ array ပုံစံနဲ့သတ်မှတ်ထားတဲ့ name တွေဟာ sequences ပါပဲ။ sequence type တွေကို (၁) vector (၂) deque (၃) list (၄) stack (၅) queue နဲ့ (၆) priority\_queue type ဆိုပြီး (၆) မျိုးထပ်ခွဲထားပါတယ်။ ကောင်းပြီ ၊ ကျွန်တော်တို့ vector class type ကစပြီး sequence တွေကိုတစ်ခုချင်း လေ့လာကြည့်ရအောင်။

## ၁၄.၁ The vector Class Template

၁။ vector class template တွေဟာ array element တွေကို random access လုပ်ပေးနိုင်တဲ့အစွမ်း ရှိပါတယ်။ sequence ရဲ့အဆုံးနေရာမှာ element တွေကိုဖြည့်မယ် ၊ ဖြုတ်မယ်ဆိုရင် vector ဟာအလွန်မြန်ပါတယ်။ sequence ရဲ့အစ သို့မဟုတ် အလယ်နေရာတွေမှာ အလုပ်လုပ်ခိုင်းမယ်ဆိုရင် နည်းနည်းပိုကြာမှာပါ။ Ex1401.cpp program ဟာ vector object တစ်ခုကို create လုပ်ပြီး content တွေကို screen မှာ display လုပ်ပြခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 14.1: Creating a simple vector

```
#include <iostream>
```

```
#include <vector>
```

```
int main( )
```

```
{
```

```
    // create a 10-element vector object,  
    // of which all the elements are initialized to 15  
    vector <int> intVec(10, 15);
```

```
    // initialize vector element count to zero  
    int kount = 0;
```

```
    // iterator object iter points to an element stored in the container  
    vector <int>::iterator iter;
```

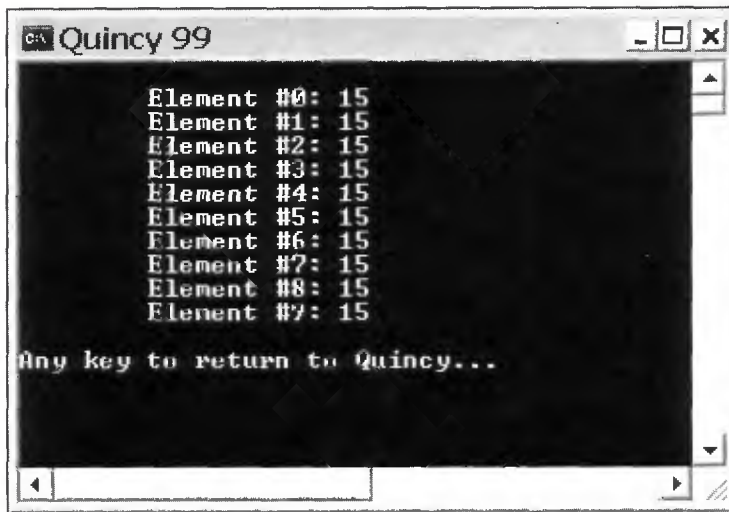
```
    // display the content of the vector object  
    cout << endl;
```

```
    for (iter= intVec.begin( ); iter != intVec.end( ); iter++)  
        cout << "\tElement #" << kount++ << ": " << *iter << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1401.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် စတင်ချင်းမှာ 10-element ပါဝင်တဲ့ intVec ဆိုတဲ့ vector object တစ်ခုကို create လုပ်ပြီး element တိုင်းရဲ့တန်ဖိုးကို 15 ဖြစ်အောင် initialize လုပ်ထားပါတယ်။ program ကနေ iterator object ဖြစ်တဲ့ iter ကိုအသုံးပြုပြီး vector object က element တွေကိုတစ်ခုချင်း access လုပ်သွားပါတယ်။ တစ်ကယ်တော့ iterator ဆိုတာ container ထဲက element တစ်ခုချင်းကို point လုပ်နိုင်တဲ့ pointer တစ်ခုလို့ မြင်ကြည့်လို့ရပါတယ်။ for loop ထဲမှာ begin( ) member function ကို call ခေါ်လိုက်တာနဲ့ vector object က first element ကို point လုပ်နေတဲ့ iterator တစ်ခုကို return လုပ်ပေးပါလိမ့်မယ်။ ထို့နည်းတူ end( ) member function ကို call ခေါ်လိုက်ရင် vector ရဲ့ last element ကို point လုပ်တဲ့ iterator ကို return လုပ်ပေးမှာပါပဲ။ ပုံ (၁၄. ၁) မှာ Ex1401.cpp program ကို run ပြထားပါတယ်။ vector element အားလုံးဟာ 15 ချည်းပဲဖြစ်တဲ့အတွက် output မှာ 15 တွေပေါ်နေတာဖြစ်ပါတယ်။



ပုံ (၁၄. ၁)

## Adding Elements to a Vector

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1402.cpp program ဟာ empty vector object တစ်ခုနဲ့ sequence မှာ new element တွေကို `push_back( )` member function အသုံးပြုပြီး insert လုပ်ပေးတဲ့ program တစ်ခုဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 14.2: Adding elements to a vector

```
#include <iostream>
```

```
#include <vector>
```

```
int main( )
```

```
{
```

```
    // create an empty vector object
```

```
    vector<char> charVec;
```

```
    // initialize vector element count to zero
```

```
    int kount = 0;
```

```

// populate the sequence with the characters 'A' through 'J'
for (int j=0; j<10; ++j)
    charVec.push_back(65 + j);

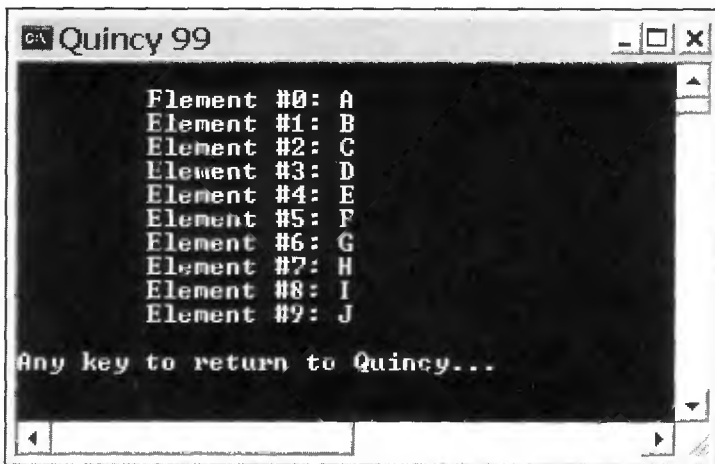
vector<char>::iterator iter;
cout << endl;

// display the content of the vector object
for (iter= charVec.begin( ); iter != charVec.end( ); iter++)
    cout << "\tElement #" << kount++ << ": " << *iter << endl;

return 0;
}

```

၂။ Ex1402.cpp program ကိုလေ့လာကြည့်မယ်ဆိုရင် စတုရန်းမှာ charVec ဆိုတဲ့ <char> type vector object တစ်ခုကို create လုပ်ပြီး element တွေကို 'A' ကနေ 'J' အထိ character တွေနဲ့ assign လုပ်ပေးမှာပါ။ push\_back( ) member function အသုံးပြုပုံကို သတိပြုကြည့်ပါ။ အရင်ဆုံးသွင်းတဲ့ data ကို first element အနေနဲ့သတ်မှတ်ပါတယ်။ display လုပ်ရင်လည်း အပေါ်ဆုံး သို့မဟုတ် ရှေ့ဆုံးမှာရှိနေမှာပါ။ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၄. ၂) မှာပြထားတဲ့အတိုင်း vector object ရဲ့ content တွေပေါ်လာတာကို တွေ့ရပါလိမ့်မယ်။



ပုံ (၁၄. ၂)



# Inserting Elements anywhere in a Vector

၁။ အခုတစ်ခါတင်ပြမှာကတော့ `insert( )` member function ကိုအသုံးပြုပြီး `vector` object တစ်ခုရဲ့ `sequence` မှာ `element` တွေကိုကြိုက်တဲ့နေရာမှာ `insert` လုပ်ပေးတဲ့နည်းကိုတင်ပြပါမယ်။ Ex1403.cpp program ကိုလေ့လာကြည့်ပါ။

// Listing 14.3: Inserting elements in a vector

```
#include <iostream>
```

```
#include <vector>
```

```
int main( )
```

```
{
```

```
    // Create and populate the vector
```

```
    vector<char> charVec;
```

```
    for (int j=0; j<10; ++j)    charVec.push_back(65 + j);
```

```
    // Display the starting vector
```

```
    cout << "\n\tOriginal vector : ";
```

```
    vector<char>::iterator iter;
```

```
    for (iter= charVec.begin( ); iter != charVec.end( ); iter++)
```

```
        cout << *iter;
```

```
    cout << endl << endl;
```

```
    // Insert five Xs into the vector starting from the front
```

```
    vector<char>::iterator start = charVec.begin( );
```

```
    charVec.insert(start, 5, '*');
```

```
    // Display the result
```

```
    cout << "\tResultant vector : ";
```

```
    for (iter = charVec.begin( ); iter != charVec.end( ); iter++)
```

```
        cout << *i;
```

```
    cout << endl;
```

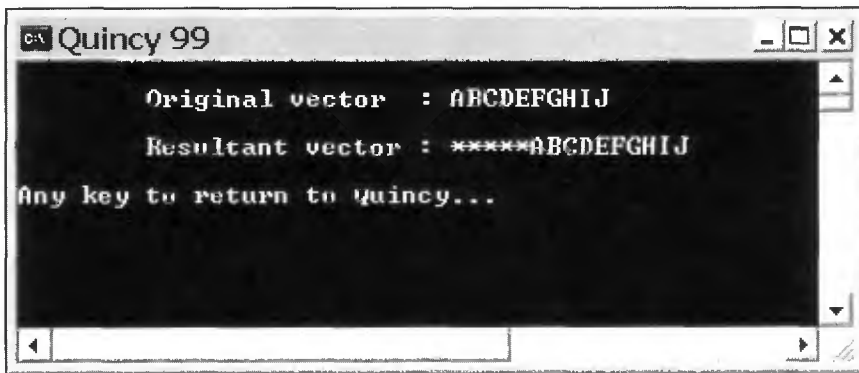
```
    return 0;
```

```
}
```

၂။ Ex1403.cpp program မှာ insert( ) member function အသုံးပြုပုံကို သတိပြုကြည့်ပါ။

```
vector<char>::iterator start = charVec.begin( );  
charVec.insert(start, 5, '*');
```

original vector ရဲ့ရှေ့ဆုံးနေရာကို start = charVec.begin( ); လို့သတ်မှတ်ပေးပြီး asterisk '\*' (5) လုံး ကို start နေရာကစပြီး insert လုပ်ပေးအောင် program ရေးထားပါတယ်။ Ex1403.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၄. ၃) မှာပြထားတဲ့အတိုင်း sequence ရဲ့ရှေ့ဆုံးနေရာတွေမှာ asterisk '\*' တွေ ဖြည့်ပြီးသားဖြစ်နေတာကို တွေ့ရပါလိမ့်မယ်။



ပုံ (၁၄. ၃)

## Removing Elements from a Vector

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1404.cpp program ဟာဆိုရင် ABCDEFGHIJ ဆိုတဲ့ sequence ရဲ့ နောက်ဆုံးကနေ element တစ်ခုချင်းကို ဖြုတ်ချပေးသွားတဲ့ program ဖြစ်ပါတယ်။ pop\_back( ) function အသုံးပြုပုံကို လေ့လာကြည့်ပါ။

```
// Listing 14.4: Removing vector elements  
#include <iostream>  
#include <vector>
```

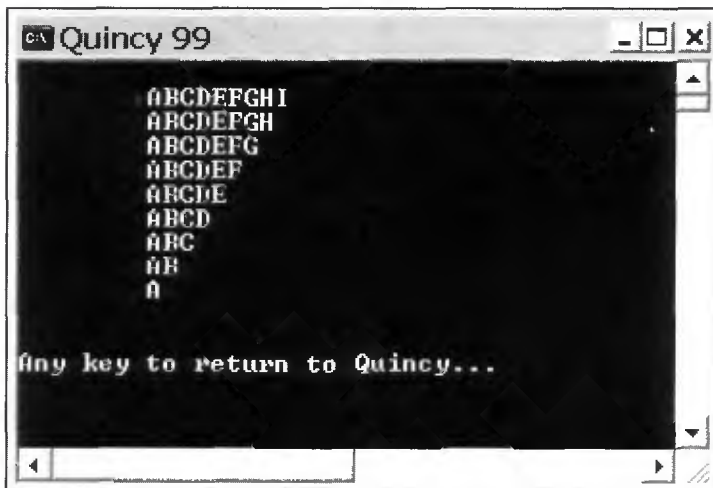
```

int main( )
{
    vector<char> charVec;
    for (int j=0; j<10; ++j)
        charVec.push_back(65 + j);

    int size = charVec.size( );
    cout << endl;
    for (int k=0; k<size; ++k)
    {
        charVec.pop_back( );
        vector<char>::iterator iter;
        cout << '\t';
        for (iter= charVec.begin( ); iter != charVec.end( ); iter++)
            cout << *iter;
        cout << endl;
    }
    return 0;
}

```

၂။ ပုံ (၁၄.၄) မှာ Ex1404.cpp program ကို run ပြထားပါတယ်။



ပုံ (၁၄.၄)

# Removing Elements Anywhere within a Vector

၁။ `pop_back( )` member function လိုပဲအသုံးပြုလို့ရတာက `erase( )` member function ဖြစ်ပါတယ်။ အောက်မှာဖော်ပြထားတဲ့ Ex1405.cpp program မှာ ABCDEFGHIJ ဆိုတဲ့ sequence ရဲ့ ရှေ့ဆုံးကနေ element တစ်ခုချင်းကို ဖျက်ပေးသွားတဲ့ program ပါ။ `erase( )` function အသုံးပြုပုံကို လေ့လာကြည့်ပါ။

// Listing 14.5: Removing elements anywhere within a vector

```
#include <iostream>
```

```
#include <vector>
```

```
int main( )
```

```
{
```

```
    vector<char> charVec;
```

```
    for (int x=0; x<10; ++x)
```

```
        charVec.push_back(65 + x);
```

```
    int size = charVec.size( );
```

```
    cout << endl;
```

```
    for (int x=0; x<size; ++x)
```

```
    {
```

```
        vector<char>::iterator start = charVec.begin( );
```

```
        charVec.erase(start);      // erase forward
```

```
        vector<char>::iterator iter;
```

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

```
        for (iter= charVec.begin( ); iter != charVec.end( ); iter++)
```

```
            cout << *iter;
```

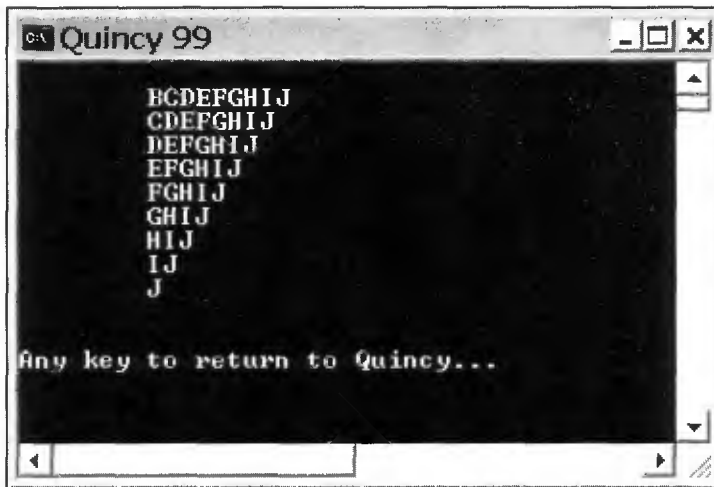
```
        cout << endl;
```

```
    }
```

```
    return 0;
```

```
}
```

၂။ ပုံ (၁၄.၅) မှာ Ex1405.cpp program ကို run ပြထားပါတယ်။



ပုံ (၁၄. ၅)

## Comparing Vectors

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1406.cpp program ဟာဆိုရင် <char> vector object (2) ခုထဲက အယ်ညွှန်းကကြီးလဲဆိုတာကို နှိုင်းယှဉ်ခိုင်းတဲ့ program ဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 14.6: Comparing vectors

```
#include <iostream>
```

```
#include <vector>
```

```
int main( )
```

```
{
```

```
    // Create two vector objects.
```

```
    vector<char> charVec1;
```

```
    for (int x=0; x<10; ++x)
```

```
        charVec1.push_back(65 + x);
```

```
    vector<char> charVec2;
```

```
    for (int x=0; x<10; ++x)
```

```
        charVec2.push_back(66 + x);
```

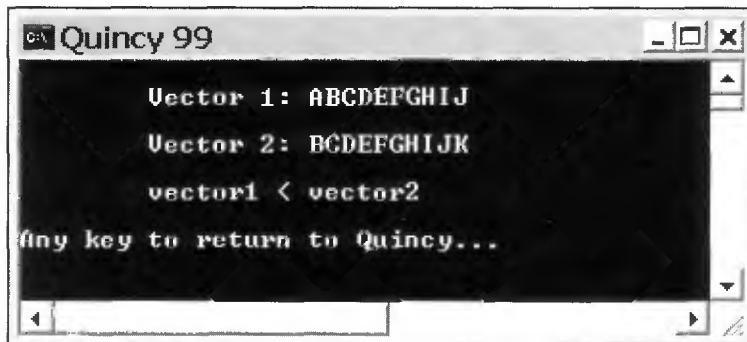
```

// Display the vectors.
cout << "\n\tVector 1: ";
vector<char>::iterator iter;
for (iter= charVec1.begin( ); iter != charVec1.end( ); iter++)
    cout << *iter;
cout << endl;
cout << "\n\tVector 2: ";
for (iter= charVec2.begin( ); iter != charVec2.end( ); iter++)
    cout << *iter;
cout << endl;

// Compare the vectors.
if (charVec1 == charVec2)
    cout << "\n\tvector1 == vector2";
else if (charVec1 < charVec2)
    cout << "\n\tvector1 < vector2";
else
    cout << "\n\tvector1 > vector2";
cout << endl;
return 0;
}

```

၂။ ဝုံ (၁၄. ၆) မှာ Ex1406.cpp program ကို run ပြထားပါတယ်။



```

C:\ Quincy 99
Vector 1: ABCDEFGHIJ
Vector 2: BCDEFGHIJK
vector1 < vector2
Any key to return to Quincy...

```

ဝုံ (၁၄. ၆)

## ၁၄.၂ Sorting a Vector of Integers

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1407.cpp program ဟာ rand( ) function အသုံးပြုပြီး generate လုပ်ထားတဲ့ <int> vector sequence ကဂဏန်းတွေကို ငယ်စဉ်ကြီးလိုက် sort လုပ်ပေးတဲ့ program တစ်ခု ဖြစ်ပါတယ်။ sort( ) function အသုံးပြုပုံကို လေ့လာကြည့်ပါ။

// Listing 14.7: Sorting a vector of integers

```
#include <iostream>
```

```
#include <iomanip>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
int main( )
```

```
{
```

```
    int n;
```

```
    int i;
```

```
    cout << "\n\tHow many integers? ";
```

```
    cin >> n;
```

```
    vector<int> intVec;
```

```
    for (i=0; i < n; i++)
```

```
        intVec.insert(intVec.end( ), rand( ));
```

```
    cout << "\n\t--- Unsorted ---\n";
```

```
    vector<int>::iterator iter;
```

```
    int k=0;
```

```
    for (iter = intVec.begin( ); iter != intVec.end( ); iter++)
```

```
    {
```

```
        if ((k%4) == 0) cout << endl;
```

```
        k++;
```

```
        cout << setw(8) << *iter;
```

```
    }
```

```

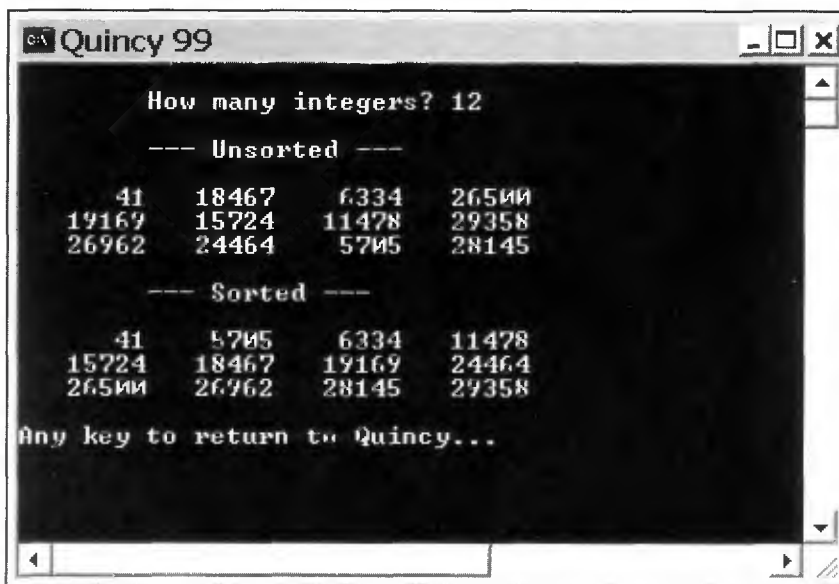
cout << "\n\n\t--- Sorted ---\n";
sort(intVec.begin( ), intVec.end( ));

for (iter = intVec.begin( ); iter != intVec.end( ); iter++)
{
    if ((k%4) == 0) cout << endl;
    k++;
    cout << setw(8) << *iter;
}

cout << endl;
return 0;
}

```

၂။ Ex1407.cpp program ကို run လိုက်ပြီး How many integers? လို့ prompt ပေါ်လာတဲ့အခါမှာ 12 ကိုရိုက်ထည့်ပေးရင် ပထမ ၈ နား (12) လုံးကို random generate လုပ်ပြီး ငယ်စဉ်ကြီးလိုက် အသစ်ပြန်စီပေးထားတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၄.၇) ကိုကြည့်ပါ။



ပုံ (၁၄.၇)



## ၁၄.၃ The deque Class Template

၁။ deque class template ဟာ vector တစ်ခုနဲ့ အတူတူပါပဲ။ deque sequence ရဲ့အစနဲ့အဆုံးနေရာမှာ element တွေကိုဖြည့်မယ် ၊ ဖြုတ်မယ်ဆိုရင် vector ထက်ပိုမြန်ပါတယ်။ array element တွေကို random access လုပ်ပေးနိုင်တဲ့အစွမ်းရှိပါတယ်။ vector လိုပဲ array size ကို dynamically resize လုပ်ပေးနိုင်ပါတယ်။ Ex1408.cpp program ဟာဆိုရင် deque object တစ်ခုကို create လုပ်ပြီး content တွေကို screen မှာ display လုပ်ပြနိုင်တဲ့ program တစ်ခုပါ။ ပုံ (၁၄. ၈) မှာ program ကို run ပြထားပါတယ်။

// Listing 14.8: Creating a simple deque

```
#include <iostream>
```

```
#include <deque>
```

```
int main( )
```

```
{
```

```
    deque<int> intDeq(5, 1234);
```

```
    int kount = 0;
```

```
    deque<int>::iterator iter;
```

```
    cout << endl;
```

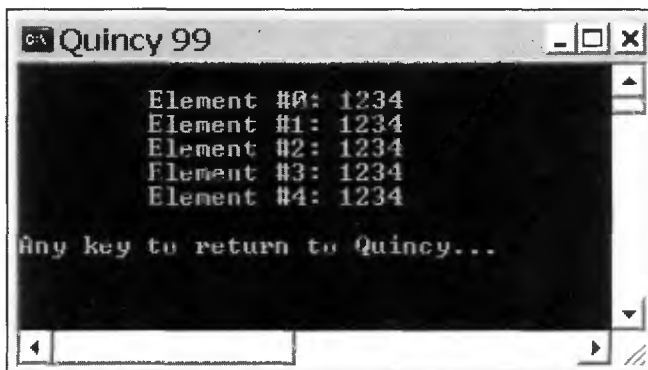
```
    for (iter = intDeq.begin( ); iter != intDeq.end( ); iter++)
```

```
        cout << "\tElement #" << kount++ << ": "
```

```
            << *iter << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၄. ၈)

# Adding Elements to a Deque

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1409.cpp program ဟာ empty deque object တစ်ခုရဲ့ sequence မှာ new element တွေကို push\_front( ) member function အသုံးပြုပြီး အစဆုံးသွင်းတဲ့ data ကို last element အနေနဲ့ဖြစ်အောင် insert လုပ်ပေးတဲ့ program တစ်ခုဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။ ပုံ (၁၄. ၉) မှာ Ex1409.cpp program ကို run ပြထားပါတယ်။

// Listing 14.9: Adding elements to a deque

```
#include <iostream>
```

```
#include <deque>
```

```
int main( )
```

```
{
```

```
    deque<char> charDeq;
```

```
    int kount= 0;
```

```
    for (int i=0; i<5; ++i)
```

```
        charDeq.push_front(65 + i);
```

```
    deque<char>::iterator iter;
```

```
    cout << endl;
```

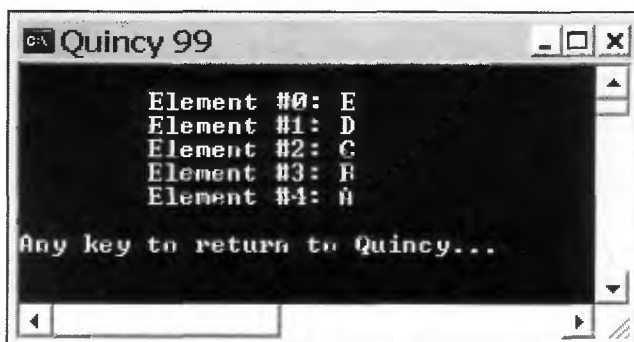
```
    for (iter= charDeq.begin( ); iter != charDeq.end( ); iter++)
```

```
        cout << "\tElement #" << kount++ << ": "
```

```
        << *iter << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၄. ၉)

# Inserting Elements anywhere in a Deque

၁။ အခုတစ်ခါတင်ပြမှာကတော့ `insert( )` member function ကိုအသုံးပြုပြီး deque object တစ်ခုရဲ့ sequence မှာ element တွေကိုကြိုက်တဲ့နေရာမှာ insert လုပ်ပေးတဲ့နည်းကို တင်ပြမှာပါမယ်။ Ex1410.cpp program မှာရေးထားတဲ့ဥစ္စာကို လေ့လာကြည့်ပါ။

```
// Listing 14.10: Inserting elements anywhere within a deque
#include <iostream>
#include <deque>

int main( )
{
    deque<char> charDeq;
    for (int x=0; x<10; ++x)
        charDeq.push_front(65 + x);

    cout << "\n\tOriginal deque : ";
    deque<char>::iterator iter;
    for (iter= charDeq.begin( ); iter != charDeq.end( ); iter++)
        cout << *iter;
    cout << endl;

    deque<char>::iterator start = charDeq.begin();
    charDeq.insert(start, 5, '$');

    cout << "\n\tResultant deque: ";
    for (iter= charDeq.begin( ); iter != charDeq.end( ); iter++)
        cout << *iter;

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

Ex1410.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၄. ၁၀) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။

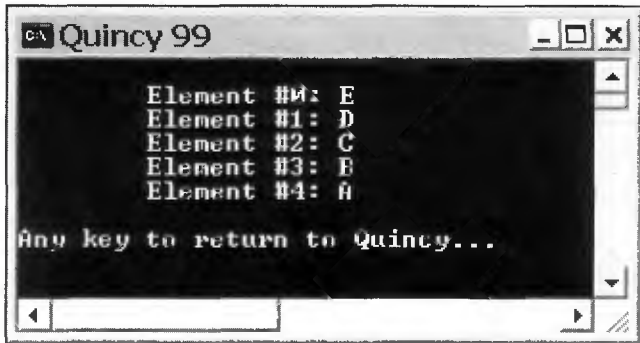
# Adding Elements to a Deque

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1409.cpp program ဟာ empty deque object တစ်ခုရဲ့ sequence မှာ new element တွေကို push\_front( ) member function အသုံးပြုပြီး အစဆုံးသွင်းတဲ့ data ကို last element အနေနဲ့ဖြစ်အောင် insert လုပ်ပေးတဲ့ program တစ်ခုဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။ ပုံ (၁၄. ၉) မှာ Ex1409.cpp program ကို run ပြထားပါတယ်။

```
// Listing 14.9: Adding elements to a deque
#include <iostream>
#include <deque>

int main( )
{
    deque<char> charDeq;
    int kount= 0;
    for (int i=0; i<5; ++i)
        charDeq.push_front(65 + i);

    deque<char>::iterator iter;
    cout << endl;
    for (iter= charDeq.begin( ); iter != charDeq.end( ); iter++)
        cout << "\tElement #" << kount++ << ": "
            << *iter << endl;
    return 0;
}
```



ပုံ (၁၄. ၉)

# Inserting Elements anywhere in a Deque

၁။ အခုတစ်ခါတင်ပြမှာကတော့ `insert( )` member function ကိုအသုံးပြုပြီး deque object တစ်ခုရဲ့ sequence မှာ element တွေကိုကြိုက်တဲ့နေရာမှာ insert လုပ်ပေးတဲ့နည်းကို တင်ပြမှာပါမယ်။ Ex1410.cpp program မှာရေးထားတဲ့ဥပဒေကို လေ့လာကြည့်ပါ။

// Listing 14.10: Inserting elements anywhere within a deque

```
#include <iostream>
```

```
#include <deque>
```

```
int main( )
```

```
{
```

```
    deque<char> charDeq;
```

```
    for (int x=0; x<10; ++x)
```

```
        charDeq.push_front(65 + x);
```

```
    cout << "\n\tOriginal deque : ";
```

```
    deque<char>::iterator iter;
```

```
    for (iter= charDeq.begin( ); iter != charDeq.end( ); iter++)
```

```
        cout << *iter;
```

```
    cout << endl;
```

```
    deque<char>::iterator start = charDeq.begin();
```

```
    charDeq.insert(start, 5, '$');
```

```
    cout << "\n\tResultant deque: ";
```

```
    for (iter= charDeq.begin( ); iter != charDeq.end( ); iter++)
```

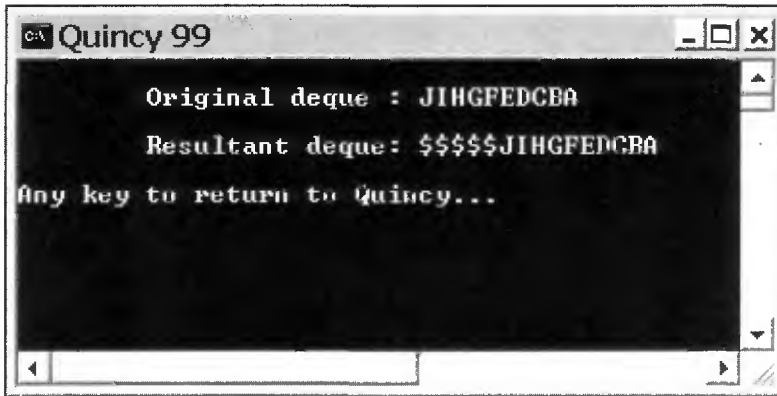
```
        cout << *iter;
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1410.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၄. ၁၀) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၄.၁၀)

## Removing Elements from a Deque

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1411.cpp program ဟာဆိုရင် ONMLKJIHG ဆိုတဲ့ sequence ရဲ့ element တွေကို `push_front( )` function ကိုအသုံးပြုပြီး create လုပ်ပြီးတော့ နောက်ဆုံးကနေ element တစ်ခုချင်းကို ဖြုတ်ချပေးသွားတဲ့ program ဖြစ်ပါတယ်။ `pop_back( )` function အသုံးပြုပုံကို လေ့လာကြည့်ပါ။

// Listing 14.11: Removing elements from a deque

```
#include <iostream>
```

```
#include <deque>
```

```
int main( )
```

```
{
```

```
    deque<char> charDeq;
```

```
    for (int x=0; x<10; ++x)
```

```
        charDeq.push_front(70 + x);
```

```
    int size = charDeq.size();
```

```
    cout << endl;
```

```
    for (int x=0; x<size; ++x)
```

```
    {
```

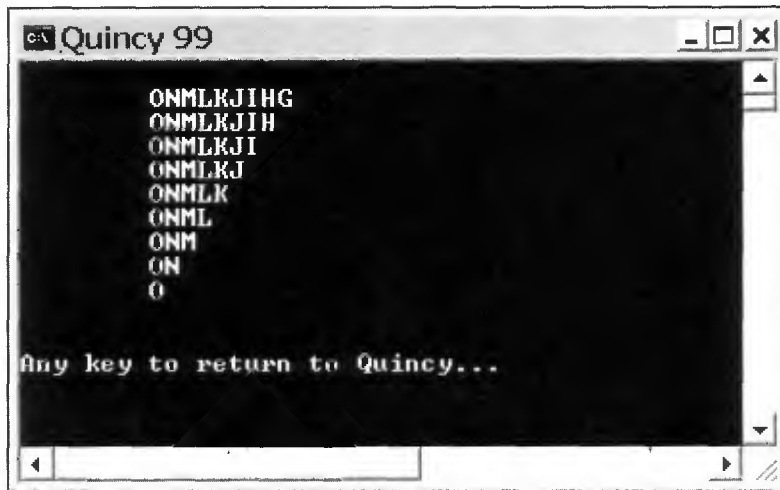
```
        charDeq.pop_back( );
```

```

        cout << '\t';
        deque<char>::iterator iter;
        for (iter= charDeq.begin(); iter != charDeq.end(); iter++)
            cout << *iter;
        cout << endl;
    }
    return 0;
}

```

Ex1411.cpp program ကို ပုံ (၁၄. ၁၁) မှာ run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။



ပုံ (၁၄. ၁၁)

## Removing Elements Anywhere within a Deque

`push_front( )` နဲ့ `erase( )` member function တွေကို အသုံးပြုပြီး deque object ကို create လုပ်မယ် ၊ element တွေကိုပြန်ဖျက်မယ်ဆိုရင် Ex1412.cpp program အတိုင်းရေးလို့ရပါတယ်။ ပုံ (၁၄. ၁၂) မှာ program ကို run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 14.12: Removing elements anywhere within a deque

```
#include <iostream>
```

```
#include <deque>
```

```
int main( )
```

```
{
```

```
    deque<char> charDeq;
```

```
    for (int x=0; x<5; ++x)
```

```
        charDeq.push_front(65 + x);
```

```
    int size = charDeq.size( );
```

```
    cout << endl;
```

```
    for (int x=0; x<size; ++x)
```

```
    {
```

```
        deque<char>::iterator start = charDeq.begin();
```

```
        charDeq.erase(start);
```

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

```
        deque<char>::iterator iter;
```

```
        for (iter= charDeq.begin( ); iter != charDeq.end( ); iter++)
```

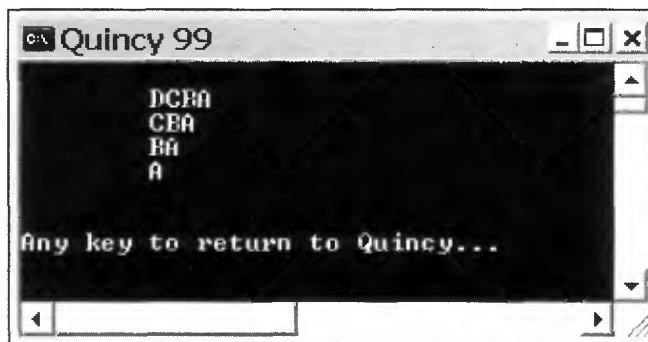
```
            cout << *iter;
```

```
        cout << endl;
```

```
    }
```

```
    return 0;
```

```
}
```



0 ( 09. 0J)



# Comparing Deques

အောက်မှာဖော်ပြထားတဲ့ Ex1413.cpp program မှာ <char> vector object (2) ခုကို ဘယ်သူက  
မေးလဲဆိုတာ နှိုင်းယှဉ်ပြထားပါတယ်။ ပုံ (၁၄. ၁၃) မှာ program ကို run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 14.13: Comparing deques

```
#include <iostream>
```

```
#include <deque>
```

```
int main( )
```

```
{
```

```
    deque<char> charDeq1;
```

```
    for (int x=0; x<10; ++x)    charDeq1.push_front(70 + x);
```

```
    deque<char> charDeq2;
```

```
    for (int x=0; x<10; ++x)    charDeq2.push_front(65 + x);
```

```
    cout << "\n\tDeque 1: ";
```

```
    deque<char>::iterator iter;
```

```
    for (iter= charDeq1.begin( ); iter != charDeq1.end( ); iter++)
```

```
        cout << *iter;
```

```
    cout << endl;
```

```
    cout << "\n\tDeque 2: ";
```

```
    for (iter = charDeq2.begin( ); iter != charDeq2.end( ); iter++)
```

```
        cout << *iter;
```

```
    cout << endl;
```

```
    if (charDeq1 == charDeq2)
```

```
        cout << "\n\tdeque1 == deque2";
```

```
    else if (charDeq1 < charDeq2)
```

```
        cout << "\n\tdeque1 < deque2";
```

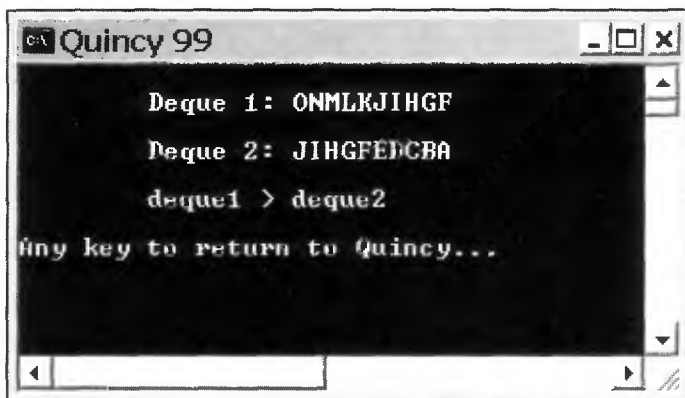
```
    else if (charDeq1 > charDeq2)
```

```
        cout << "\n\tdeque1 > deque2";
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၄. ၁၃)

## ၁၄.၄ The list Class Template

အောက်မှာဖော်ပြထားတဲ့ Ex1414.cpp program ဟာ `rand( )` function အသုံးပြုပြီး generate လုပ်ထားတဲ့ `<int>` list sequence ကကစားနန်းတွေကို ငယ်စဉ်ကြီးလိုက် sort လုပ်ပေးတဲ့ program တစ်ခုဖြစ်ပါတယ်။ `sort( )` function အသုံးပြုပုံကို လေ့လာကြည့်ပါ။ ပုံ (၁၄. ၁၄) မှာ program ကို run ပြထားပါတယ်။

// Listing 14.14: Creating a single list

```
#include <iostream>
```

```
#include <list>
```

```
int main( )
```

```
{
```

```
    list<int> intList(5, 123);
```

```
    int kount = 0;
```

```
    list<int>::iterator iter;
```

```
    cout << endl;
```

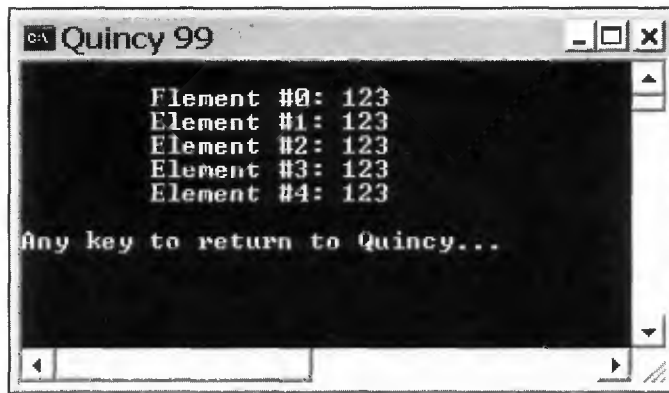
```
    for (iter= intList.begin(); iter != intList.end(); iter++)
```

```
        cout << "\tElement #" << kount++ << ": "
```

```
            << *iter << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၄. ၁၄)

## Adding Elements to a List

အောက်မှာဖော်ပြထားတဲ့ Ex1415.cpp program ဟာ empty list object တစ်ခုရဲ့ sequence မှာ new element တွေကို `push_front( )` member function အသုံးပြုပြီး insert လုပ်ပေးတဲ့ program တစ်ခုဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။ ပုံ (၁၄. ၁၅) မှာ program ကို run ပြထားပါတယ်။

// Listing 14.15: Adding elements to a list

```
#include <iostream>
```

```
#include <list>
```

```
int main( )
```

```
{
```

```
    list<char> charList;
```

```
    int kount = 0;
```

```
    for (int i=0; i<7; ++i)      charList.push_front(65 + i);
```

```
    list<char>::iterator iter;
```

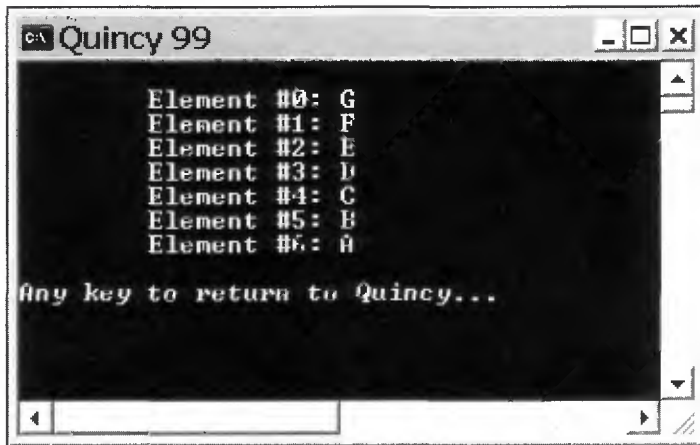
```
    cout << endl;
```

```
    for (iter = charList.begin( ); iter != charList.end( ); iter++)
```

```
        cout << "\tElement #" << kount++ << ": " << *iter << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၄. ၁၅)

## Inserting Elements anywhere in a List

၁။ အခုတစ်ခါတင်ပြမှာကတော့ `insert( )` member function ကိုအသုံးပြုပြီး `list` object တစ်ခုရဲ့ `sequence` မှာ `blank element` တွေကိုကြိုက်တဲ့နေရာမှာ `insert` လုပ်ပေးတဲ့နည်းကိုတင်ပြပါမယ်။ `Ex1416.cpp` program ကိုလေ့လာကြည့်ပါ။

// Listing 14.16: Inserting elements anywhere within a list

```
#include <iostream>
```

```
#include <list>
```

```
int main( )
```

```
{
```

```
    list<char> charList;
```

```
    for (int x=0; x<10; ++x)
```

```
        charList.push_front(65 + x);
```

```
    cout << "\n\tOriginal list : ";
```

```
    list<char>::iterator iter;
```

```
    for (iter= charList.begin( ); iter != charList.end( ); iter++)
```

```
        cout << *iter;
```

```

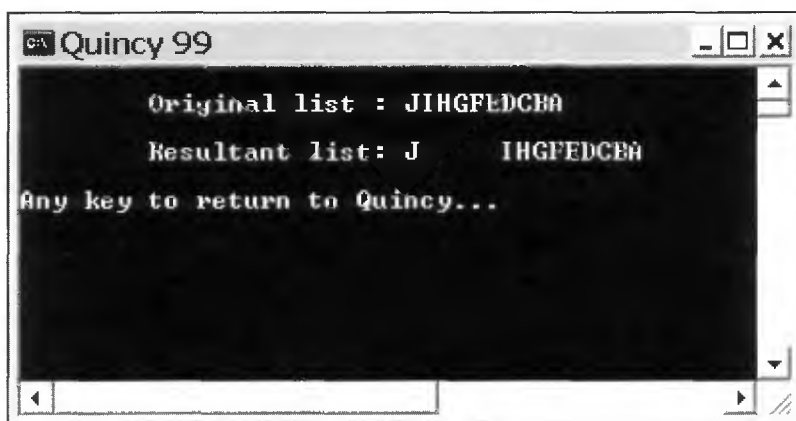
cout << endl;

list<char>::iterator start = charList.begin();
charList.insert(++start, 5, ' ');

cout << "\n\tResultant list: ";
for (iter= charList.begin( ); iter != charList.end( ); iter++)
    cout << *iter;
cout << endl;
return 0;
}

```

၂။ ပုံ (၁၄. ၁၆) မှာ Ex1416.cpp program ကို run ပြထားပါတယ်။



ပုံ (၁၄. ၁၆)

## Removing Elements from a List

၁။ Ex1417.cpp program ဟာဆိုရင် ABCDEFG ဆိုတဲ့ sequence ထဲက elements 'E' နဲ့ 'B' တို့ကို ဖြုတ်ချပေးသွားတဲ့ program ဖြစ်ပါတယ်။ push\_back( ) function နဲ့ remove( ) function တွေအသုံးပြုပုံကို လေ့လာကြည့်ပါ။

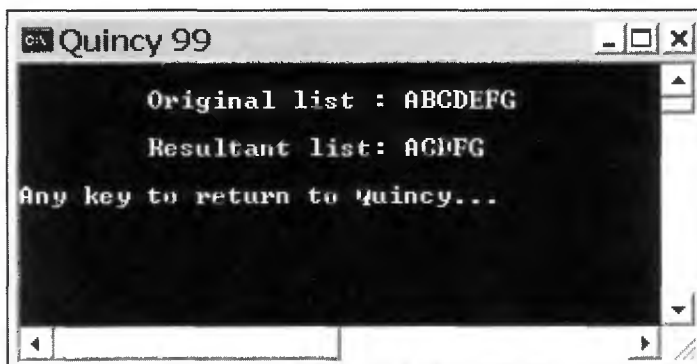
```
// Listing 14.17: Removing elements from a list
#include <iostream>
#include <list>

int main( )
{
    list<char> charList;
    for (int x=0; x<7; ++x)
        charList.push_back(65 + x);

    cout << "\n\tOriginal list : ";
    list<char>::iterator iter;
    for (iter= charList.begin( ); iter != charList.end( ); iter++)
        cout << *iter;
    cout << endl;

    charList.remove('E');
    charList.remove('B');
    cout << "\n\tResultant list: ";
    for (iter= charList.begin( ); iter != charList.end( ); iter++)
        cout << *iter;
    cout << endl;
    return 0;
}
```

၂။ ပုံ (၁၄. ၁၇) မှာ Ex1417.cpp program ကို run ပြထားပါတယ်။



ပုံ (၁၄. ၁၇)

## Removing Elements Anywhere within a List

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1418.cpp program မှာ JIHGFEDCBA ဆိုတဲ့ sequence ရဲ့ကနေ element 'I' ကိုဖျက်ပေးတဲ့ program ဖြစ်ပါတယ်။ erase( ) function အသုံးပြုပုံကို လေ့လာကြည့်ပါ။

// Listing 14.18: Removing elements anywhere within a list

```
#include <iostream>
```

```
#include <list>
```

```
int main( )
```

```
{
```

```
    list<char> charList;
```

```
    for (int x=0; x<10; ++x)
```

```
        charList.push_front(65 + x);
```

```
    list<char>::iterator iter;
```

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

```
    for (iter= charList.begin( ); iter != charList.end( ); iter++)
```

```
        cout << *iter;
```

```
    cout << endl;
```

```
    list<char>::iterator start = charList.begin();
```

```
    charList.erase(++start);
```

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

```
    for (iter= charList.begin( ); iter != charList.end( ); iter++)
```

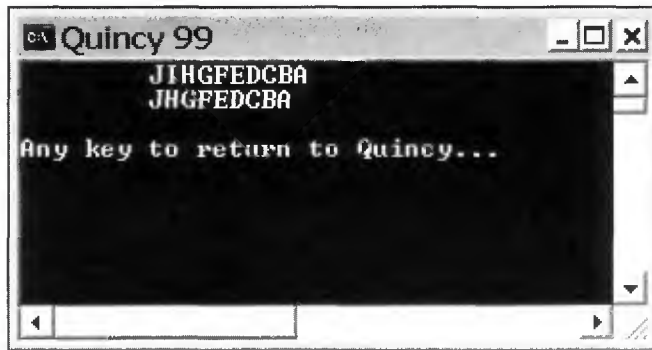
```
        cout << *iter;
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```

Ex1418.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၄. ၁၈) မှာပြထားတဲ့အတိုင်း မြင်ရမှာပါ။



ပုံ (၁၄. ၁၈)

## Comparing Lists

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1419.cpp program မှာ <char> list object (2) ခု ဘယ်သူကကြီးလဲ ဆိုတာကို နှိုင်းယှဉ်ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 14.19: Comparing lists

```
#include <iostream>
```

```
#include <list>
```

```
int main( )
```

```
{
```

```
    list<char> charList1;
```

```
    for (int x=0; x<10; ++x)
```

```
        charList1.push_front(65 + x);
```

```
    list<char> charList2;
```

```
    for (int x=0; x<10; ++x)
```

```
        charList2.push_front(66 + x);
```

```
    cout << "\n\tlist 1: ";
```

```
    list<char>::iterator iter;
```



```

for (iter= charList1.begin( ); iter != charList1.end( ); iter++)
    cout << *iter;
cout << endl;

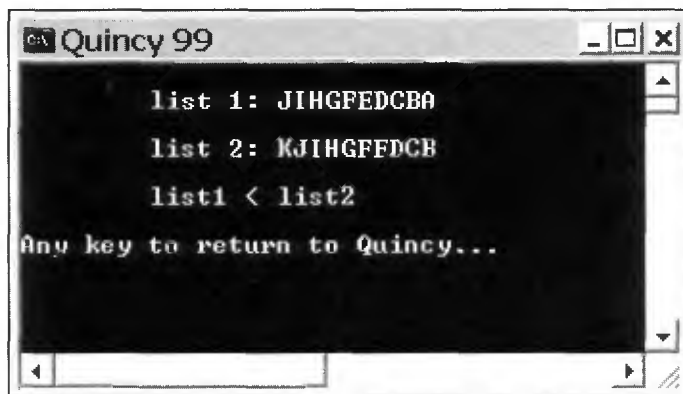
cout << "\n\tlist 2: ";
for (iter = charList2.begin( ); iter != charList2.end( ); iter++)
    cout << *iter;
cout << endl;

if (charList1 == charList2)
    cout << "\n\tlist1 == list2";
else if (charList1 < charList2)
    cout << "\n\tlist1 < list2";
else if (charList1 > charList2)
    cout << "\n\tlist1 > list2";

cout << endl;
return 0;
}

```

၂။ Ex1419.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၄. ၁၉) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၁၄. ၁၉)

# ၁၄.၅ The stack Container Adaptor

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1420.cpp program ဟာ push( ) နဲ့ pop( ) function တွေကိုအသုံးပြုပြီး stack sequence မှာဝင်ရောက်မှုတွေကို သွင်းပေးတာနဲ့ ပြန်ထုတ်ပေးတဲ့ program ဖြစ်ပါတယ်။

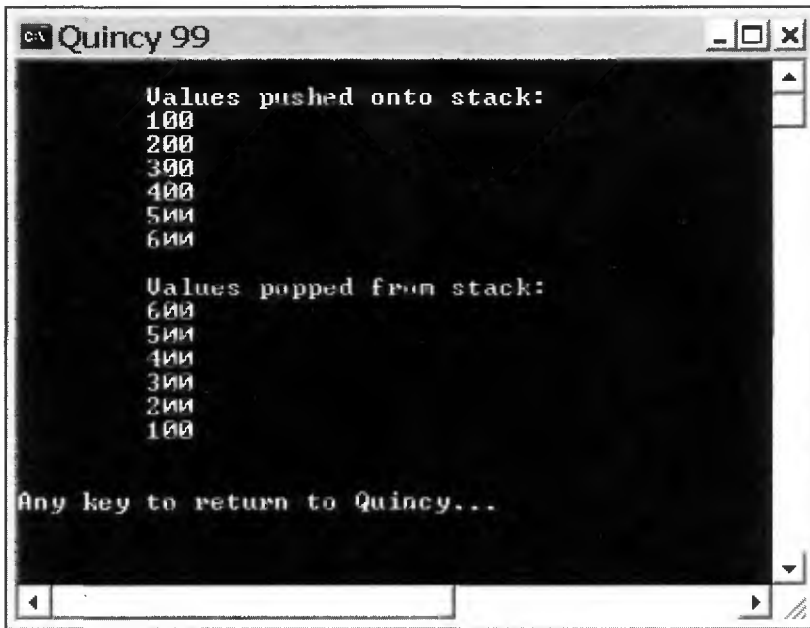
```
// Listing 14.20: Managing a stack
#include <iostream>
#include <list>
#include <stack>

int main( )
{
    stack<int, list<int> > intStack;

    cout << "\n\tValues pushed onto stack:\n";
    for (int x=1; x<7; ++x)
    {
        intStack.push(x*100);
        cout << '\t' << x*100 << endl;
    }

    cout << "\n\tValues popped from stack:\n";
    int size = intStack.size( );
    for (int x=0; x<size; ++x)
    {
        cout << '\t' << intStack.top( ) << endl;
        intStack.pop();
    }
    cout << endl;
    return 0;
}
```

၂။ Ex1420.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၄.၂၀) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။



ပုံ (၁၄. ၂၀)

## Sorting a Stack of Integers

အောက်မှာဖော်ပြထားတဲ့ Ex1421.cpp program ဟာ random generate လုပ်ထားတဲ့ stack sequence ဂဏန်းတွေကို ယ်စဉ်ကြီးလိုက် sort လုပ်ပေးတဲ့ program တစ်ခုဖြစ်ပါတယ်။ push( ) နဲ့ pop( ) function တွေအသုံးပြုပုံကို လေ့လာကြည့်ပါ။

```

// Listing 14.21: Stacking an array of integers
#include <iostream>
#include <iomanip>
#include <stack>

```

```

int main( )
{
    int n;

```

```

cout << "\n\tHow many integers? ";
cin >> n;
stack<int> intStk;                                // a stack of integers

cout << "\n\t--- Pushing ---\n";                // push values onto the stack
for (int i= 0; i < n; i++)
{
    if ((i%4) == 0)
        cout << endl;
    int rn = rand( );
    cout << setw(8) << rn;
    intStk.push(rn);
}

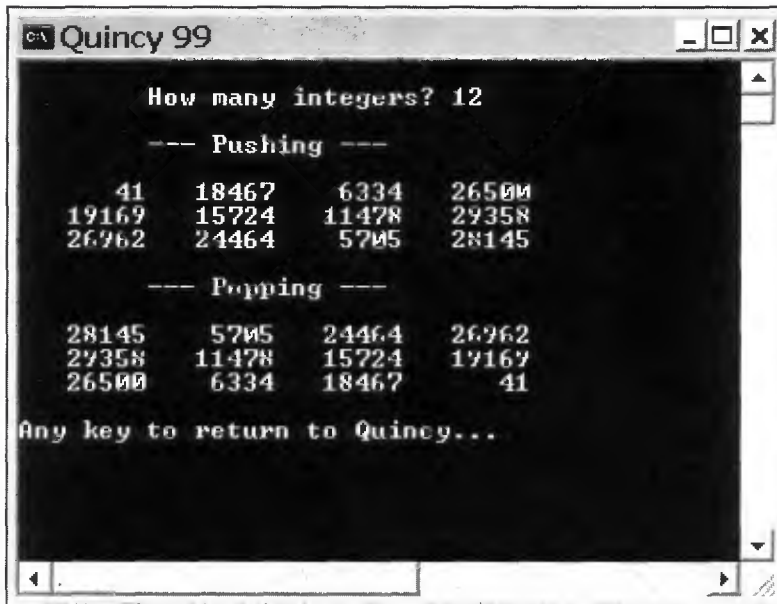
cout << "\n\n\t--- Popping ---\n";
for (int j=0; !intStk.empty( );j++)
{
    if ((j%4) == 0)
        cout << endl;
    cout << setw(8) << intStk.top( );
    intStk.pop( );
}
cout << endl;
return 0;
}

```

၂။ ပုံ (၁၄. ၂၁) မှာ Ex1421.cpp program ကို run ပြထားပါတယ်။

## ၁၄.၆ The queue Container Adaptor

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1422.cpp program ဟာ push( ) function အသုံးပြုပြီး generate လုပ်ထားတဲ့ queue sequence က ဂဏန်းတွေကို pop( ) function နဲ့ remove ပြန်လုပ်ပြထားပါတယ်။



٥ (٥٩. ٣٥)

// Listing 14.22: Managing a queue

```
#include <iostream>
```

```
#include <list>
```

```
#include <queue>
```

```
int main( )
```

```
{
```

```
    queue<int, list<int> > intQue;
```

```
    cout << "\n\tValues pushed onto queue:\n";
```

```
    for (int x=1; x<7; ++x)
```

```
    {
```

```
        intQue.push(x*100);
```

```
        cout << '\t' << x*100 << endl;
```

```
    }
```

```
    cout << "\n\tValues removed from queue:\n";
```

```
    int size = intQue.size( );
```

```

cout << "\n\tHow many integers? ";
cin >> n;
stack<int> intStk;                                // a stack of integers

cout << "\n\t--- Pushing ---\n";                // push values onto the stack
for (int i= 0; i < n; i++)
{
    if ((i%4) == 0)
        cout << endl;
    int m = rand( );
    cout << setw(8) << m;
    intStk.push(m);
}

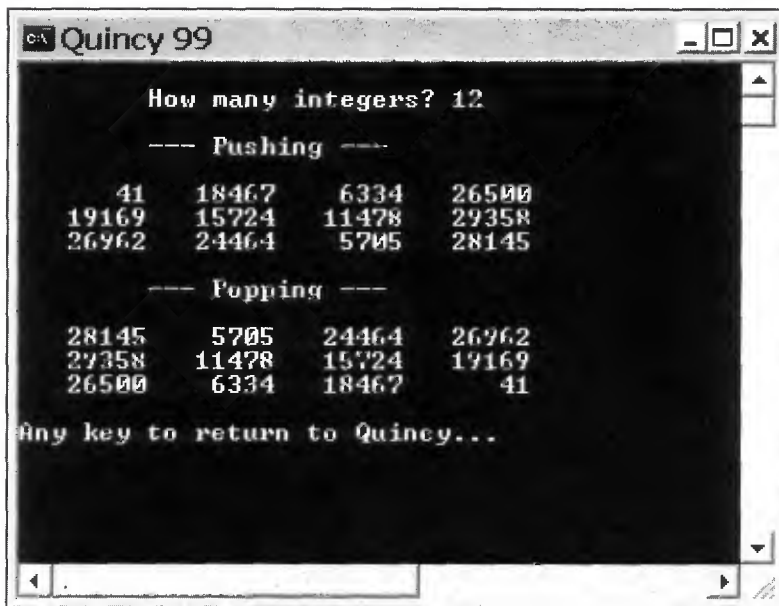
cout << "\n\n\t--- Popping ---\n";
for (int j=0; !intStk.empty( );j++)
{
    if ((j%4) == 0)
        cout << endl;
    cout << setw(8) << intStk.top( );
    intStk.pop( );
}
cout << endl;
return 0;
}

```

၁။ ဝုံ (၁၄. ၂၁) မှာ Ex1421.cpp program ကို run ပြထားပါတယ်။

## ၁၄.၆ The queue Container Adaptor

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1422.cpp program ဟာ push( ) function အသုံးပြုပြီး generate လုပ်ထားတဲ့ queue sequence က ဂဏန်းတွေကို pop( ) function နဲ့ remove ပြန်လုပ်ပြထားပါတယ်။



٥٩٠ ١٥

// Listing 14.22: Managing a queue

```
#include <iostream>
```

```
#include <list>
```

```
#include <queue>
```

```
int main( )
```

```
{
```

```
    queue<int, list<int> > intQue;
```

```
    cout << "\n\tValues pushed onto queue:\n";
```

```
    for (int x=1; x<7; ++x)
```

```
    {
```

```
        intQue.push(x*100);
```

```
        cout << '\t' << x*100 << endl;
```

```
    }
```

```
    cout << "\n\tValues removed from queue:\n";
```

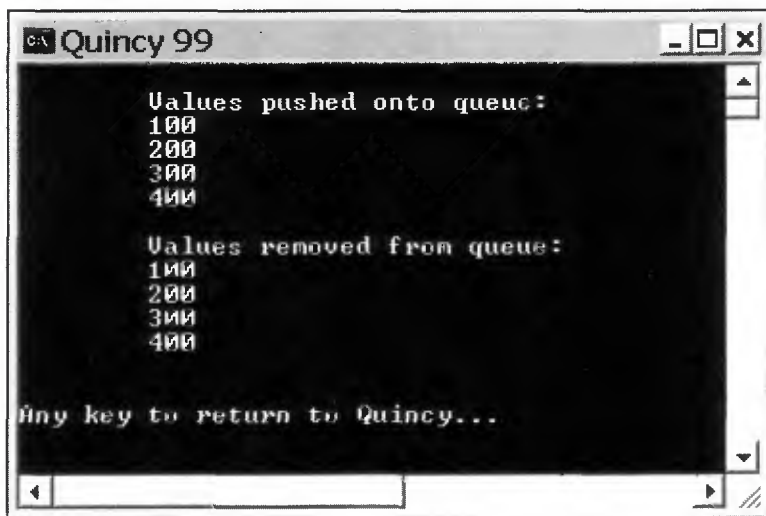
```
    int size = intQue.size( );
```

```

for (int x=0; x<size; ++x)
{
    cout << '\t' << intQue.front( ) << endl;
    intQue.pop( );
}
cout << endl;
return 0;
}

```

၂။ ပုံ (၁၄. ၂၂) မှာ Ex1422.cpp program ကို run ပြထားပါတယ်။



ပုံ (၁၄. ၂၂)

## ၁၄.၇ The priority\_queue Container Adaptor

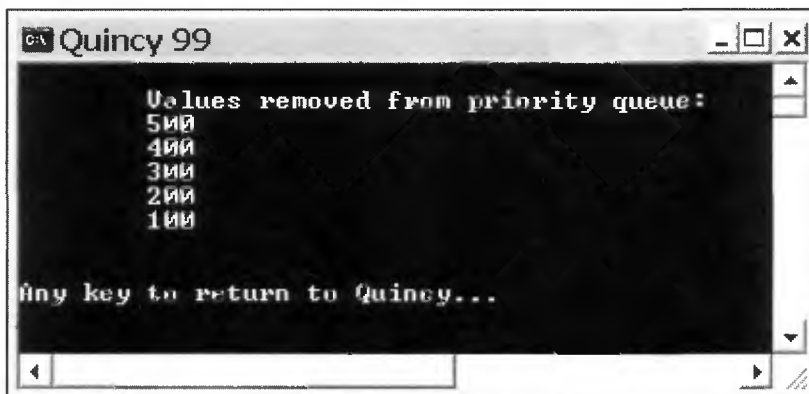
၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1423.cpp program ဟာဆိုရင် queue sequencer မှာ random add လုပ်ထားတဲ့ဂဏန်းတွေကို ကြီးစဉ်ငယ်လိုက် remove ပြန်လုပ်ပေးတဲ့နည်းကို program ရေးပြထားပါတယ်။



```
// Listing 14.23: Managing a priority_queue
#include <iostream>
#include <list>
#include <queue>

int main( )
{
    priority_queue<int, vector<int> > intPQue;
    intPQue.push(400);
    intPQue.push(100);
    intPQue.push(500);
    intPQue.push(300);
    intPQue.push(200);
    cout << "\n\tValues removed from priority queue:\n";
    int size = intPQue.size( );
    for (int x=0; x<size; ++x)
    {
        cout << '\t' << intPQue.top( ) << endl;
        intPQue.pop();
    }
    cout << endl;
    return 0;
}
```

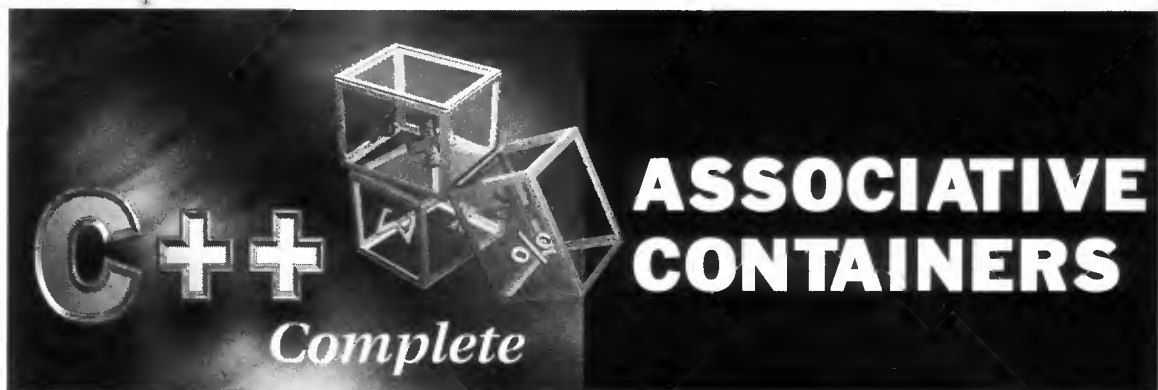
ပုံ (၁၄. ၂၃) မှာ Ex1423.cpp program ကို run ပြထားပါတယ်။



```
Quincy 99
Values removed from priority queue:
500
400
300
200
100
Any key to return to Quincy...
```

ပုံ (၁၄. ၂၃)

# Chapter 15



STL associative container တွေဟာ sequence တွေနဲ့မတူပါဘူး။ container ထဲက element တစ်ခုချင်းအတွက်သတ်မှတ်ထားတဲ့ key တွေကိုအသုံးပြုမှ element တွေကို locate လုပ်နိုင်မှာပါ။ associative container type တွေကို (၅) မျိုးခွဲထားပါတယ်။ (၁) set (၂) multiset (၃) map (၄) multimap နဲ့ (၅) bitset type တို့ဖြစ်ပါတယ်။ ကောင်းပြီ ၊ ကျွန်တော်တို့ set class template ကံစပြီး တစ်ခုချင်းလေ့လာ ကြည့်ရအောင်။

## ၁၅.၁ The set Class Template

၁။ set class object တစ်ခုဟာ တန်ဖိုးအစုတစ်ခုကို sorted order ဖြစ်အောင် program မှာထည့်သွင်း အသုံးပြုလို့ရပါတယ်။ set တစ်ခုဟာ ordered list တစ်ခုနဲ့ ဆင်တူပါတယ်။ set element တွေဟာ stored data အနေနဲ့ပါဝင်သလို data တွေအတွက် key တွေအဖြစ် အလုပ်လုပ်ဆောင်ပါတယ်။ Ex1501.cpp ဟာဆိုရင် set object တစ်ခုကို create လုပ်ပြီး content တွေကို screen မှာ display လုပ်ပြခိုင်းတဲ့ program တစ်ခု ဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 15.1: Creating a simple set

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

```
    // Create the set object.
```

```
    set<int> intSet;
```

```
    // Populate the set with values.
```

```
    intSet.insert(10);
```

```
    intSet.insert(5);
```

```
    intSet.insert(1);
```

```
    intSet.insert(3);
```

```
    intSet.insert(8);
```

```
    // Display the contents of the set.
```

```
    cout << "\n\tContents of set:\n";
```

```
    set<int>::iterator iter;
```

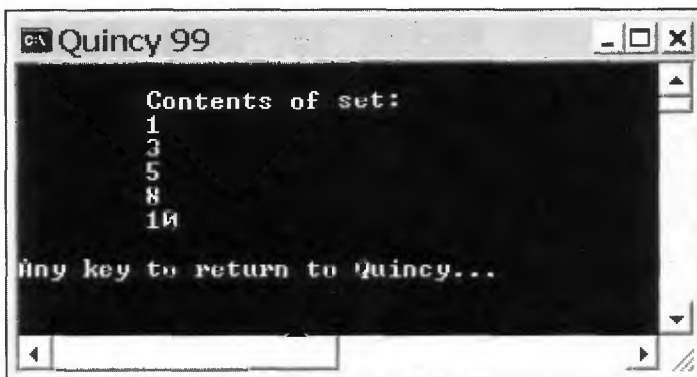
```
    for (iter=intSet.begin( ); iter!=intSet.end( ); iter++)
```

```
        cout << '\t' << *iter << endl;
```

```
    return 0;
```

```
}
```

Ex1501.cpp program ကို run လိုက်မယ်ဆိုရင် set object ထဲကို insert လုပ်ပေးတဲ့ element တွေဟာ အလိုအလျောက် စီပြီးသားဖြစ်သွားတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၅. ၁) ကိုကြည့်ပါ။



ပုံ (၁၅. ၁)

# Adding Elements to a set

တစ်ကယ်လို့ char element တွေကို set ထဲမှာ insert လုပ်ပေးရင်လည်း စောစောကနဲ့အတူတူပါပဲ ၊ စီပြီးသားဖြစ်နေပါပြီ။ ပုံ (၁၅. ၂) ကိုကြည့်ပါ။

// Listing 15.2: Adding char elements to a set

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

```
    set<char> charSet;
```

```
    charSet.insert('C');
```

```
    charSet.insert('E');
```

```
    charSet.insert('B');
```

```
    charSet.insert('D');
```

```
    charSet.insert('A');
```

```
    cout << "\n\tContents of set:\n";
```

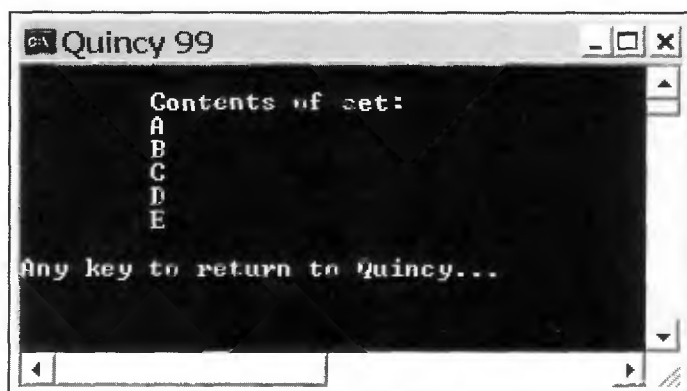
```
    set<char>::iterator iter;
```

```
    for (iter= charSet.begin( ); iter != charSet.end( ); iter++)
```

```
        cout << '\t' << *iter << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၅. ၂)

# Removing Elements anywhere within a set

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1503.cpp program မှာ empty set object တစ်ခုကို create အရင် လုပ်ပြီး new element တွေကို insert( ) function အသုံးပြုပြီး စီပြီးသားဖြစ်အောင်ထည့်ပေးပါတယ်။ နောက်ပြီး charSet.erase(++iter); statement ကနေ second element ကို erase လုပ်ပြီး new element တွေ ကို display လုပ်ခိုင်းထားပါတယ်။

// Listing 15.3: Removing elements anywhere within a set

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

```
    set<char> charSet;
```

```
    charSet.insert('E');
```

```
    charSet.insert('D');
```

```
    charSet.insert('C');
```

```
    charSet.insert('B');
```

```
    charSet.insert('A');
```

```
    // Display the contents of the set.
```

```
    cout << "\n\tContents of set:\n";
```

```
    set<char>::iterator iter;
```

```
    for (iter= charSet.begin( ); iter != charSet.end( ); iter++)
```

```
        cout << '\t' << *iter << endl;
```

```
    iter = charSet.begin( );
```

```
    charSet.erase(++iter);
```

```
    cout << "\n\tContents of new set:\n";
```

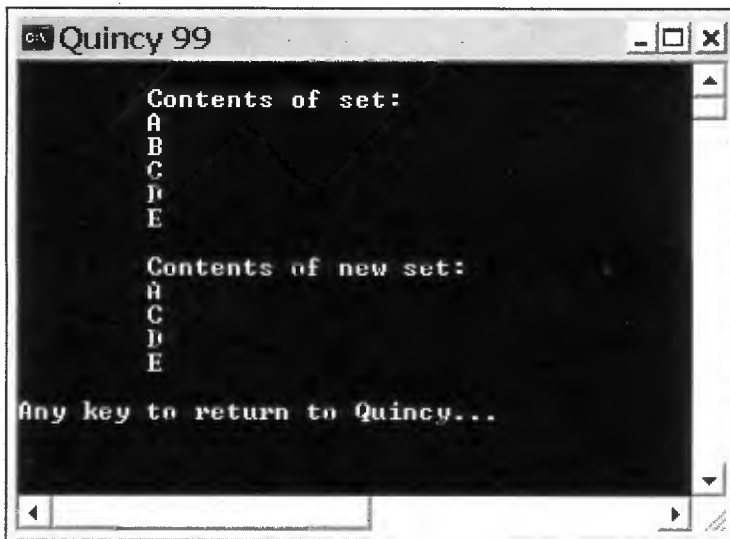
```
    for (iter= charSet.begin( ); iter != charSet.end( ); iter++)
```

```
        cout << '\t' << *iter << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1503.cpp program မှာ run လိုက်မယ်ဆိုရင် ပထမ sorted set element တွေကို display လုပ်ပြပြီး second element ဖြစ်တဲ့ 'B' ကို erase လုပ်ပြီးတော့ new element တွေကို display လုပ်ပြတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၅. ၃) ကိုကြည့်ပါ။



ပုံ (၁၅. ၃)

## Searching a set

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1504.cpp program ဟာဆိုရင် set element တွေထဲက ကြိုက်တဲ့ element တစ်ခုကိုရှာပေးတဲ့ program ဖြစ်ပါတယ်။ element ကိုရှာတွေ့ရင် Element found: လို့ကြေငြာပေးပါလိမ့်မယ်။ မတွေ့ရင် Element not found: လို့ကြေငြာပေးမှာပါ။

```
// Listing 15.4: Searching a set
#include <iostream>
#include <set>
```

```
int main( )
{
```

```

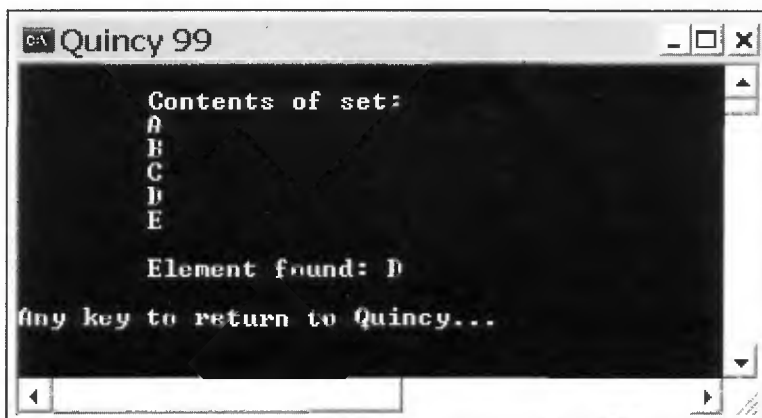
    set<char> charSet;
charSet.insert('E');
charSet.insert('D');
charSet.insert('C');
charSet.insert('B');
charSet.insert('A');

cout << "\n\tContents of set:\n";
set<char>::iterator iter;
for (iter= charSet.begin( ); iter != charSet.end( ); iter++)
    cout << '\t' << *iter << endl;

// Find the D.
iter = charSet.find('D');
if (iter == charSet.end())
    cout << "\n\tElement not found.";
else
    cout << "\n\tElement found: " << *iter;
cout << endl;
return 0;
}

```

Ex1504.cpp program မှာ run လိုက်မယ်ဆိုရင် ABCDE ဆိုတဲ့ set ထဲက 'D' ကိုရှာခိုင်းထားတဲ့အတွက် Element found: D လို့ display လုပ်ပြတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၅. ၄) ကိုကြည့်ပါ။



ပုံ (၁၅. ၄)

# Comparing sets

၁။ Ex1505.cpp program ဟာဆိုရင် set object (2) ခုကိုနှိုင်းယှဉ်ခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ်။  
လေ့လာကြည့်ပါ။

// Listing 15.5: Comparing sets

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

```
    // Create the first set object.
```

```
    set<char> charSet1;
```

```
    charSet1.insert('E');
```

```
    charSet1.insert('D');
```

```
    charSet1.insert('C');
```

```
    charSet1.insert('B');
```

```
    charSet1.insert('A');
```

```
    cout << "\n\tContents of first set:\n";
```

```
    set<char>::iterator iter;
```

```
    for (iter= charSet1.begin( ); iter != charSet1.end( ); iter++)
```

```
        cout << '\t' << *iter << endl;
```

```
    // Create the second set object.
```

```
    set<char> charSet2;
```

```
    charSet2.insert('J');
```

```
    charSet2.insert('I');
```

```
    charSet2.insert('H');
```

```
    charSet2.insert('G');
```

```
    charSet2.insert('F');
```

```
    cout << "\n\tContents of second set:\n";
```



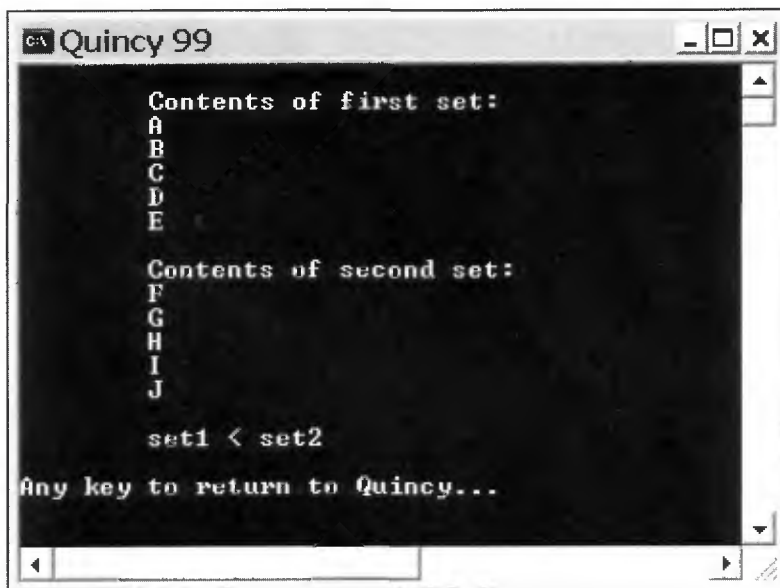
```

for (iter= charSet2.begin( ); iter != charSet2.end( ); iter++)
    cout << '\t' << *iter << endl;
cout << endl;

// Compare the sets.
if (charSet1 == charSet2)
    cout << "\tset1 == set2";
else if (charSet1 < charSet2)
    cout << "\tset1 < set2";
else
    cout << "\tset1 > set2";
cout << endl;
return 0;
}

```

Ex1505.cpp program ကို run လိုက်မယ်ဆိုရင် ABCDE နဲ့ FGHIJ ဆိုတဲ့ set (2) ခုပေါ်လာပြီး set1 က set2 ထက်ငယ်တယ်လို့ ကွန်ပျူတာမှာ display လုပ်ပြပါလိမ့်မယ်။ ပုံ (၁၅.၅) ကိုကြည့်ပါ။



```

C:\ Quincy 99
Contents of first set:
A
B
C
D
E

Contents of second set:
F
G
H
I
J

set1 < set2

Any key to return to Quincy...

```

ပုံ (၁၅.၅)

# ၁၅.၂ The multiset Class Template

၁။ multiset object တစ်ခုဟာ တန်ဖိုးအစုတစ်ခုကို sorted order ဖြစ်အောင် program မှာထည့်သွင်း အသုံးပြုလို့ရပါတယ်။ multiset တစ်ခုဟာ set နဲ့အတူတူပါပဲ။ တစ်ခုပဲပိုတာက multiset element တွေဟာ duplicate ဖြစ်လို့ရပါတယ်။ Ex1506.cpp ဟာဆိုရင် multiset object တစ်ခုကို create လုပ်ပြီး content တွေကို display လုပ်ပြခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

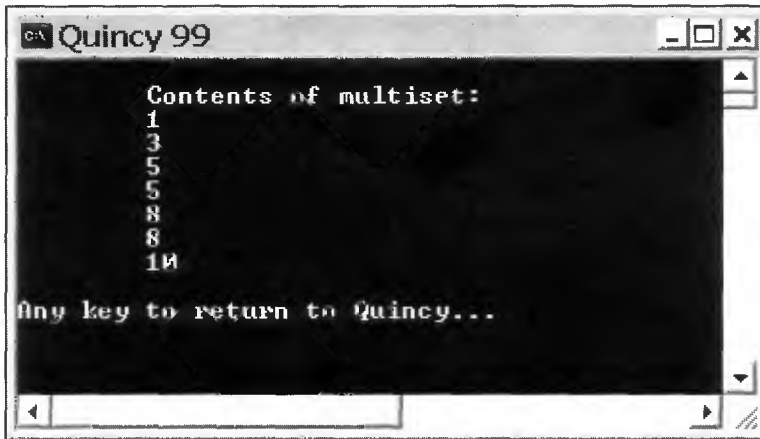
```
// Listing 15.6: Creating a simple multiset class template
#include <iostream>
#include <set>
```

```
int main( )
{
    // Create the multiset object.
    multiset<int> intMultiset;

    intMultiset.insert(10);
    intMultiset.insert(5);
    intMultiset.insert(1);
    intMultiset.insert(3);
    intMultiset.insert(8);
    intMultiset.insert(5);
    intMultiset.insert(8);

    // Display the contents of the multiset.
    cout << "\n\tContents of multiset:\n";
    multiset<int>::iterator iter;
    for (iter= intMultiset.begin( );iter != intMultiset.end( ); iter++)
        cout << '\t' << *iter << endl;
    return 0;
}
```

၂။ Ex1506.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၅. ၆) မှာပြထားတဲ့အတိုင်း 1 3 5 5 8 8 10 ဆိုတဲ့ multiset တစ်ခုကို create လုပ်ပြီး display လုပ်ပြတာကို တွေ့ရပါလိမ့်မယ်။



ပုံ (၁၅.၆)

## Inserting multiset Elements

တစ်ကယ်လို့ char element တွေကို multiset ထဲမှာ insert လုပ်ပေးရင်လည်း စောစောကနဲ့ အတူတူပါပဲ။ ပြီးသားဖြစ်နေပါပြီ။ ပုံ (၁၅.၇) ကိုကြည့်ပါ။

// Listing 15.7: Adding elements to a multiset

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

```
    multiset<char> charMultiset;
```

```
    charMultiset.insert('E');
```

```
    charMultiset.insert('D');
```

```
    charMultiset.insert('C');
```

```
    charMultiset.insert('B');
```

```
    charMultiset.insert('A');
```

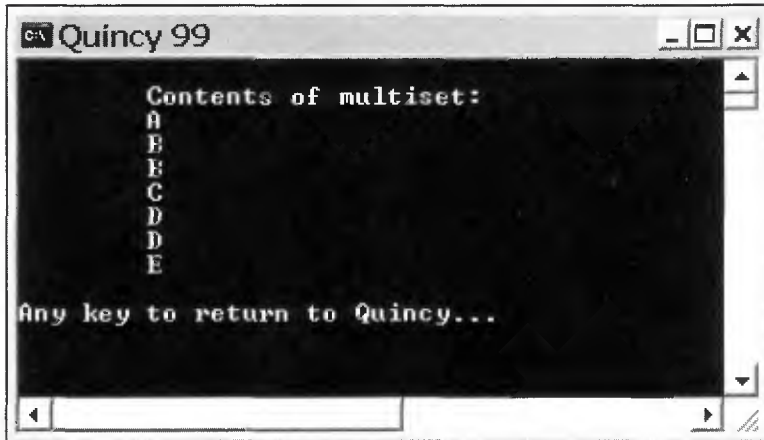
```
    charMultiset.insert('B');
```

```
    charMultiset.insert('D');
```

```

cout << "\n\tContents of multiset:\n";
multiset<char>::iterator iter;
for (iter= charMultiset.begin( ); iter != charMultiset.end( ); iter++)
    cout << '\t' << *iter << endl;
return 0;
}

```



ပုံ (၁၅.၇)

## Removing multiset Elements

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1508.cpp program ဟာဆိုရင် ABBCDDE ဆိုတဲ့ multiset တစ်ခုကို create လုပ်ပြီး second element 'B' ကို erase လုပ်ခြင်းအားဖြင့် new set ဟာ ABCDDE ပဲကျန်အောင် လုပ်ပေးတဲ့ program ဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

```

// Listing 15.8: Removing multiset elements
#include <iostream>
#include <set>

```

```

int main( )

```

```

{
    // Create the set object.
    multiset<char> charMultiset;

    // Populate the multiset with values.
    charMultiset.insert('E');
    charMultiset.insert('D');
    charMultiset.insert('C');
    charMultiset.insert('B');
    charMultiset.insert('A');
    charMultiset.insert('B');
    charMultiset.insert('D');

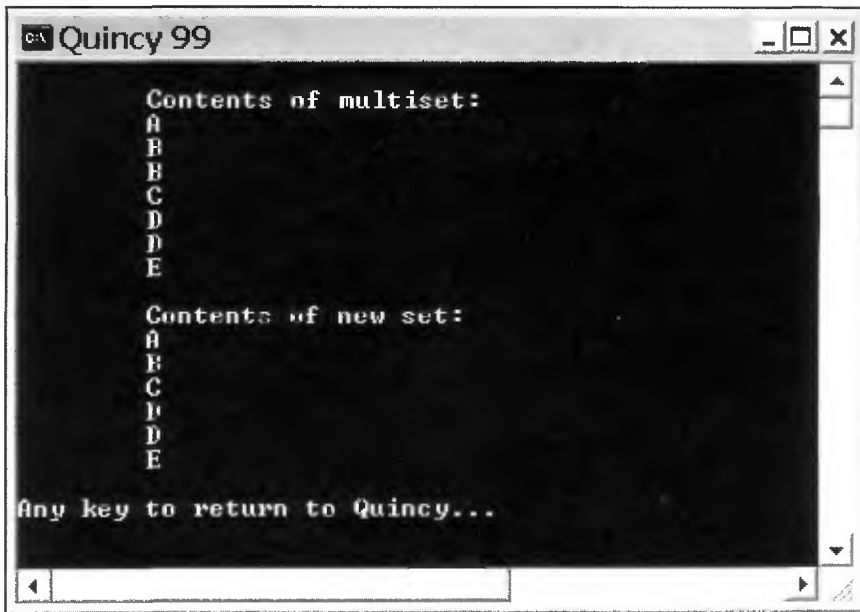
    // Display the contents of the multiset.
    cout << "\n\tContents of multiset:\n";
    multiset<char>::iterator iter;
    for (iter= charMultiset.begin( ); iter != charMultiset.end( ); iter++)
        cout << '\t' << *iter << endl;

    // Erase the multiset's second element.
    iter = charMultiset.begin( );
    charMultiset.erase(++iter);

    // Display the new contents of the multiset.
    cout << "\n\tContents of new set:\n";
    for (iter= charMultiset.begin( ); iter != charMultiset.end( ); iter++)
        cout << '\t' << *iter << endl;
    return 0;
}

```

၂။ Ex1508.cpp program မှာ run လိုက်မယ်ဆိုရင် ပထမ sorted multiset element တွေကို display လုပ်ပြီး second element ဖြစ်တဲ့ 'B' ကို erase လုပ်ပြီးတော့ new element တွေ display လုပ် ပြတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၅. ၈) ကိုကြည့်ပါ။



ပုံ (၁၅. ၈)

## Searching a multiset

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1509.cpp program ဟာဆိုရင် multiset element တွေထဲက ကြိုက်တဲ့ element တစ်ခုကိုရှာပေးတဲ့ program ဖြစ်ပါတယ်။ element ကိုရှာတွေ့ရင် Element found: လို့ကြေငြာပေးမှာပါ။ နောက်ပြီး Next element: ကဘာလို့ ဆက်ကြေငြာပါလိမ့်မယ်။ ရှာခိုင်းတဲ့ element ကိုမတွေ့ဘူးဆိုရင် Element not found: လို့ကြေငြာပေးမှာပါ။

// Listing 15.9: Searching a multiset

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

```
    multiset<char> charMultiset;
```

```
    charMultiset.insert('E');
```

```
    charMultiset.insert('D');
```

```
    charMultiset.insert('C');
```

```
charMultiset.insert('B');
charMultiset.insert('A');
charMultiset.insert('B');
charMultiset.insert('D');
```

```
cout << "\n\tContents of multiset:\n";
multiset<char>::iterator iter;
for (iter= charMultiset.begin( ); iter != charMultiset.end( ); iter++)
    cout << '\t' << *iter << endl;
cout << endl;
```

```
// Find the first D.
```

```
iter = charMultiset.find('D');
```

```
if (iter == charMultiset.end( ))
```

```
    cout << "\tElement not found.";
```

```
else
```

```
{
```

```
    cout << "\tElement found: " << *iter++ << endl;
```

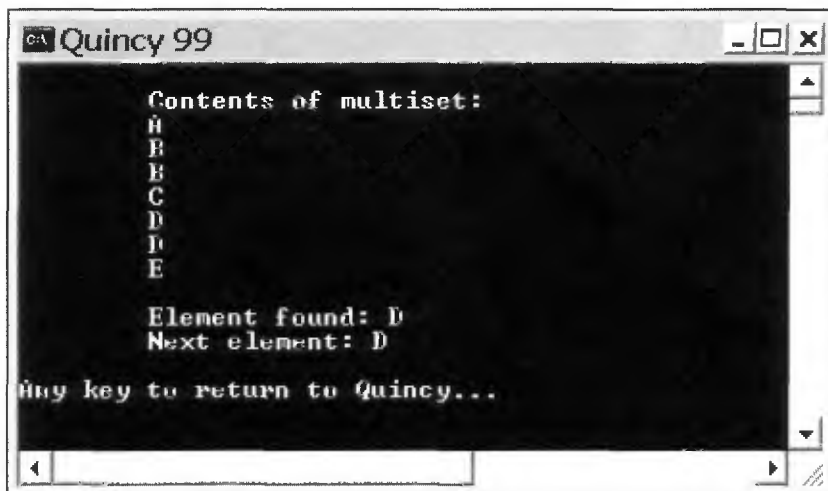
```
    cout << "\tNext element: " << *iter;
```

```
}
```

```
cout << endl;
```

```
return 0;
```

```
}
```



```
Quincy 99
Contents of multiset:
A
B
B
C
D
D
E

Element found: D
Next element: D

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

0 (39.0)

# Comparing multisets

၁။ Ex1510.cpp program ဟာဆိုရင် multiset object (၂) ခုကိုနှိုင်းယှဉ်ခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 15.10: Comparing multisets

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

```
    multiset<char> charMultiset1;
```

```
    charMultiset1.insert('E');
```

```
    charMultiset1.insert('D');
```

```
    charMultiset1.insert('C');
```

```
    charMultiset1.insert('B');
```

```
    charMultiset1.insert('A');
```

```
    charMultiset1.insert('B');
```

```
    charMultiset1.insert('D');
```

```
    cout << "\n\tContents of first multiset:\n";
```

```
    multiset<char>::iterator iter;
```

```
    for (iter= charMultiset1.begin( );iter != charMultiset1.end( ); iter++)
```

```
        cout << 't' << *iter << endl;
```

```
    cout << endl;
```

```
    multiset<char> charMultiset2;
```

```
    charMultiset2.insert('J');
```

```
    charMultiset2.insert('I');
```

```
    charMultiset2.insert('H');
```

```
    charMultiset2.insert('G');
```

```
    charMultiset2.insert('F');
```

```
    charMultiset2.insert('G');
```

```
    charMultiset2.insert('I');
```

```
    cout << "\n\tContents of second multiset:\n";
```



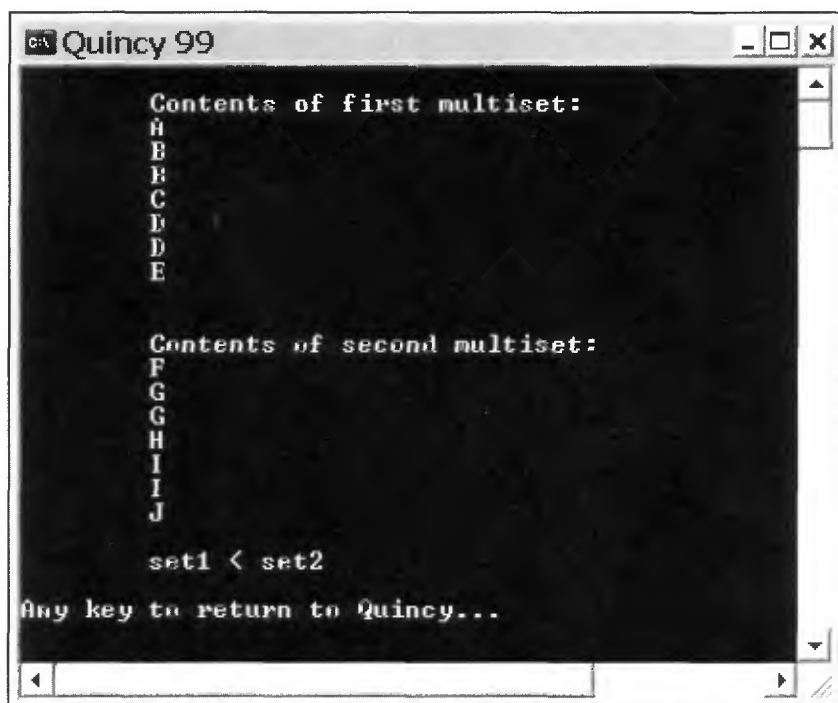
```

for (iter= charMultiset2.begin( ); iter != charMultiset2.end( ); iter++)
    cout << "\t" << *iter << endl;
cout << endl;

// Compare the sets.
if (charMultiset1 == charMultiset2)
    cout << "\tset1 == set2";
else if (charMultiset1 < charMultiset2)
    cout << "\tset1 < set2";
else
    cout << "\tset1 > set2";
cout << endl;
return 0;
}

```

၂။ Ex1510.cpp program ကို run လိုက်မယ်ဆိုရင် ABBCDDE နဲ့ FGHHIIJ ဆိုတဲ့ multiset (၂) ခု ပေါ်လာပြီး set1 က set2 ထက်ငယ်တယ်လို့ ကွန်ပျူတာမှာ display လုပ်ပြပါလိမ့်မယ်။ ပုံ (၁၅. ၁၀) ကိုကြည့်ပါ။



ပုံ (၁၅. ၁၀)

## ၁၅.၃ The map Class Template

၁။ map class object တစ်ခုဟာ တန်ဖိုးအစုတစ်ခုကို sorted order ဖြစ်အောင် program မှာထည့်သွင်း အသုံးပြုလို့ရပါတယ်။ map element တွေဟာ stored data အနေနဲ့ပါဝင်သလို data တွေအတွက် search key တွေအဖြစ် အလုပ်လုပ်ဆောင်ပါတယ်။ Ex1511.cpp ဟာဆိုရင် simple map object တစ်ခုကို create လုပ်ပြီး content တွေကို display လုပ်ပြခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ်။ ပုံ (၁၅. ၁၁) ကိုလေ့လာကြည့်ပါ။

// Listing 15.11: Creating a simple map

```
#include <iostream>
```

```
#include <map>
```

```
int main( )
```

```
{
```

```
    // Create the map object.
```

```
    map<int, char> charMap;
```

```
    // Populate the map with values.
```

```
    charMap.insert(std::map<int, char>::value_type(1,'A'));
```

```
    charMap.insert(std::map<int, char>::value_type(3,'C'));
```

```
    charMap.insert(std::map<int, char>::value_type(2,'B'));
```

```
    charMap.insert(std::map<int, char>::value_type(5,'E'));
```

```
    charMap.insert(std::map<int, char>::value_type(4,'D'));
```

```
    // Display the contents of the map.
```

```
    cout << "\n\tContents of map:\n";
```

```
    map<int, char>::iterator iter;
```

```
    for (iter= charMap.begin( ); iter != charMap.end( ); iter++)
```

```
    {
```

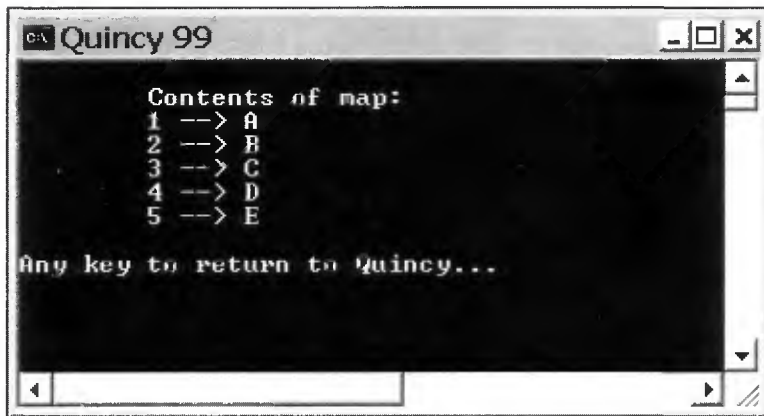
```
        cout << "\t" << (*iter).first << " --> ";
```

```
        cout << (*iter).second << endl;
```

```
    }
```

```
    return 0;
```

```
}
```



ပုံ (၁၅. ၁၁)

## Adding Elements to a map

တစ်ကယ်လို့ char element တွေကို map ထဲမှာ add လုပ်ပေးရင်လည်း စောစောကနဲ့အတူတူပါပဲ ၊ စီပြီးသားဖြစ်နေပါပြီ။ ပုံ (၁၅. ၁၂) မှာ Ex1512.cpp program ကို run ပြထားပါတယ်။

// Listing 15.12: Adding elements to a map

```
#include <iostream>
#include <map>
```

```
typedef map<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    // Create the map object.
    MYMAP charMap;
```

```
    // Populate the map with values.
```

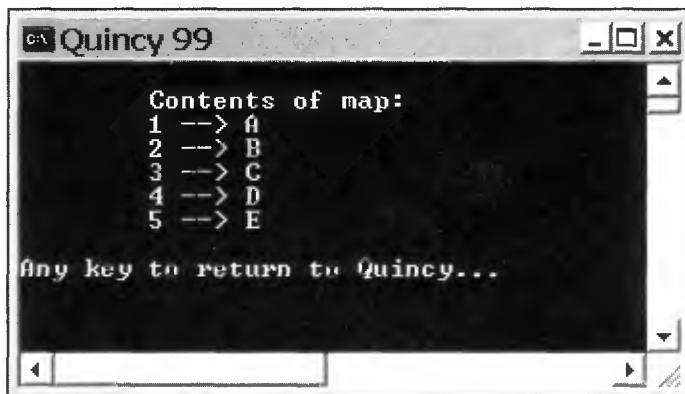
```
    charMap.insert(MYMAP::value_type(1,'A'));
    charMap.insert(MYMAP::value_type(3,'C'));
    charMap.insert(MYMAP::value_type(2,'B'));
```

```

charMap.insert(MYMAP::value_type(5,'E'));
charMap.insert(MYMAP::value_type(4,'D'));

// Display the contents of the map.
cout << "\n\tContents of map:\n";
MYMAP::iterator iter;
for (iter= charMap.begin( );iter != charMap.end( ); iter++)
{
    cout << '\t' << (*iter).first << " --> ";
    cout << (*iter).second << endl;
}
return 0;
}

```



ပုံ (၁၅. ၁၂)

## Adding Elements to a map Using the [ ] Operator

၁။ map class template မှာပါဝင်တဲ့ [ ] operator ကို အသုံးပြုမယ်ဆိုရင် insert( ) function ကို အသုံးမပြုပဲ element တွေကို map object တစ်ခုမှာ insert လုပ်ပေးလို့ရပါတယ်။ နမူနာဖော်ပြထားတဲ့ Ex1513.cpp ကိုလေ့လာကြည့်ရင်သဘောပေါက်ပါလိမ့်မယ်။

// Listing 15.13: Adding elements to a map using the [ ] operator

```
#include <iostream>
```

```
#include <map>
```

```
typedef map<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    MYMAP charMap;
```

```
    charMap[1] = 'A';
```

```
    charMap[4] = 'D';
```

```
    charMap[2] = 'B';
```

```
    charMap[5] = 'E';
```

```
    charMap[3] = 'C';
```

```
    cout << "\n\tContents of map:\n";
```

```
    MYMAP::iterator iter;
```

```
    for (iter= charMap.begin( ); iter != charMap.end( ); iter++)
```

```
    {
```

```
        cout << "\t" << (*iter).first << " --> ";
```

```
        cout << (*iter).second << endl;
```

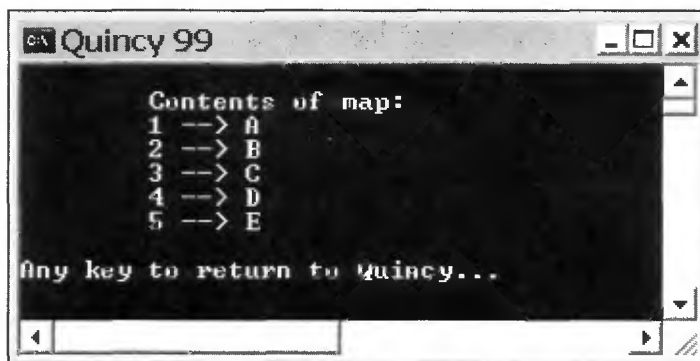
```
    }
```

```
    return 0;
```

```
}
```

၂။

Ex1513.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၅. ၁၃) မှာပြထားတဲ့အတိုင်းတွေ့ရမှာပါ။



ပုံ (၁၅. ၁၃)

# Removing map Elements

အောက်မှာဖော်ပြထားတဲ့ Ex1514.cpp program မှာ empty map object တစ်ခုကို create အရင်လုပ်ပြီး new element တွေကို [ ] operator အသုံးပြုပြီး စီပြီးသားဖြစ်အောင်ထည့်ပေးပါတယ်။ နောက်ပြီး charMap.erase(++iter); statement ကနေ second element ကို erase လုပ်ပြီး new element တွေကို display လုပ်ခိုင်းထားပါတယ်။ program ကို run လိုက်မယ်ဆိုရင် sorted set element တွေကို display လုပ်ပြီး second element ဖြစ်တဲ့ 'B' ကို erase လုပ်ပြီးတော့ new element တွေ display လုပ် ပြတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၅. ၁၄) ကိုကြည့်ပါ။

// Listing 15.14: Removing map elements

```
#include <iostream>
```

```
#include <map>
```

```
typedef map<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    MYMAP charMap;
```

```
    charMap[1] = 'A';
```

```
    charMap[4] = 'D';
```

```
    charMap[2] = 'B';
```

```
    charMap[5] = 'E';
```

```
    charMap[3] = 'C';
```

```
    cout << "\n\tContents of map:\n";
```

```
    MYMAP::iterator iter;
```

```
    for (iter= charMap.begin( ); iter != charMap.end( ); iter++)
```

```
    {
```

```
        cout << "\t" << (*iter).first << " --> ";
```

```
        cout << (*iter).second << endl;
```

```
    }
```

```
    iter = charMap.begin( );
```

```
    charMap.erase(++iter);
```

```

// Display the new contents of the map.
cout << "\n\tContents of new map:\n";
for (iter= charMap.begin( );iter != charMap.end( ); iter++)
{
    cout << '\t' << (*iter).first << " --> ";
    cout << (*iter).second << endl;
}
return 0;
}

```

```

Quincy 99
Contents of map:
1 --> A
2 --> B
3 --> C
4 --> D
5 --> E

Contents of new map:
1 --> A
3 --> C
4 --> D
5 --> E

Any key to return to Quincy...

```

ပုံ (၁၅. ၁၄)

## Searching a map

၁။ Ex1515.cpp program ဟာ map object ထဲကကြိုက်တဲ့ element တစ်ခုကိုရှာပေးတဲ့ program ဖြစ်ပါတယ်။ element ကိုရှာတွေ့ရင် Element found: လို့ကြေငြာပေးပါလိမ့်မယ်။ မတွေ့ရင် Element not found: လို့ကြေငြာပေးမှာပါ။

```

// Listing 15.15: Searching a map
#include <iostream>

```

```
#include <map>
```

```
typedef map<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    MYMAP charMap;
```

```
    // Populate the map with values.
```

```
    charMap[1] = 'A';
```

```
    charMap[4] = 'D';
```

```
    charMap[2] = 'B';
```

```
    charMap[5] = 'E';
```

```
    charMap[3] = 'C';
```

```
    cout << "\n\tContents of map:\n";
```

```
    MYMAP::iterator iter;
```

```
    for (iter= charMap.begin( );iter != charMap.end( ); iter++)
```

```
    {
```

```
        cout << "\t" << (*iter).first << " --> ";
```

```
        cout << (*iter).second << endl;
```

```
    }
```

```
    // Find the D.
```

```
    MYMAP::iterator pos = charMap.find(4);
```

```
    if (pos == charMap.end())
```

```
        cout << "\n\tElement not found.";
```

```
    else
```

```
        cout << "\n\tElement found: " << (*pos).second;
```

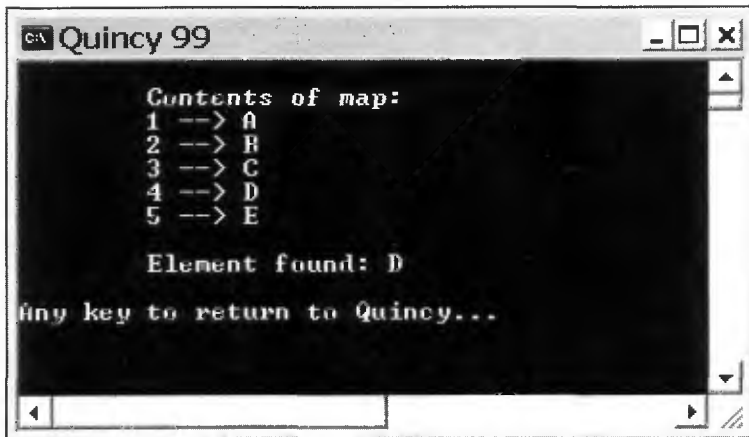
```
    cout << endl;
```

```
    return 0;
```

```
}
```

၂။ Ex1515.cpp program မှာ run လိုက်မယ်ဆိုရင် ABCDE ဆိုတဲ့ map object ထဲက 'D' ကိုရှာခိုင်း  
ထားတဲ့အတွက် Element found: D လို့ display လုပ်ပြတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၅. ၁၅) ကိုကြည့်ပါ။





ပုံ (၁၅. ၁၅)

## Comparing maps

Ex1516.cpp program ဟာဆိုရင် set object (2) ခုကိုနှိုင်းယှဉ်ခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ်။ program ကို run လိုက်မယ်ဆိုရင် ABCDE နဲ့ FG ဆိုတဲ့ map object (2) ခုပေါ်လာပြီး map1 က map2 ထက်ငယ်တယ်လို့ ကွန်ပျူတာမှာ display လုပ်ပြပါလိမ့်မယ်။ ပုံ (၁၅. ၁၆) ကိုကြည့်ပါ။

// Listing 15.16: Comparing maps

```
#include <iostream>
```

```
#include <map>
```

```
typedef map<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    // Create the first map object.
```

```
    MYMAP charMap1;
```

```
    charMap1[1] = 'A';
```

```
    charMap1[4] = 'D';
```

```
    charMap1[2] = 'B';
```

```
    charMap1[5] = 'E';
```

```
    charMap1[3] = 'C';
```

```

cout << "\n\tContents of first map:\n";
MYMAP::iterator iter;
for (iter= charMap1.begin( );iter != charMap1.end( ); iter++)
{
    cout << "\t" << (*iter).first << " --> ";
    cout << (*iter).second << endl;
}
cout << endl;

// Create the second map object.
MYMAP charMap2;
charMap2[1] = 'F';
charMap2[4] = 'I';
charMap2[2] = 'G';
charMap2[5] = 'J';
charMap2[3] = 'H';

cout << "\n\tContents of second map:\n";
for (iter= charMap2.begin( );iter != charMap2.end( ); iter++)
{
    cout << "\t" << (*iter).first << " --> ";
    cout << (*iter).second << endl;
}
cout << endl;

// Compare the maps.
if (charMap1 == charMap2)
    cout << "\tmap1 == map2";
else if (charMap1 < charMap2)
    cout << "\tmap1 < map2";
else
    cout << "\tmap1 > map2";

cout << endl;
return 0;
}

```

```
Quincy 99
Contents of first map:
1 --> H
2 --> B
3 --> C
4 --> D
5 --> E

Contents of second map:
1 --> F
2 --> G
3 --> H
4 --> I
5 --> J

map1 < map2
Any key to return to Quincy...
```

ပုံ (၁၅. ၁၆)

## ၁၅.၄ The multimap Class Template

multimap object တစ်ခုဟာ တန်ဖိုးအစုတစ်ခုကို sorted order ဖြစ်အောင် program မှာထည့်သွင်း အသုံးပြုလို့ရပါတယ်။ multimap တစ်ခုဟာ map နဲ့အတူတူပါပဲ။ တစ်ခုပဲပိုတာက multimap element တွေဟာ duplicate ဖြစ်လို့ရပါတယ်။ Ex1517.cpp program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၅. ၁၇) မှာပြထားတဲ့အတိုင်း A B B C D D E ဆိုတဲ့ multimap တစ်ခုကို create လုပ်ပြီး display လုပ်ပြတာကို တွေ့ရပါလိမ့်မယ်။

```
// Listing 15.17: A simple multimap
#include <iostream>
#include <map>

typedef multimap<int, char> MYMAP;
```

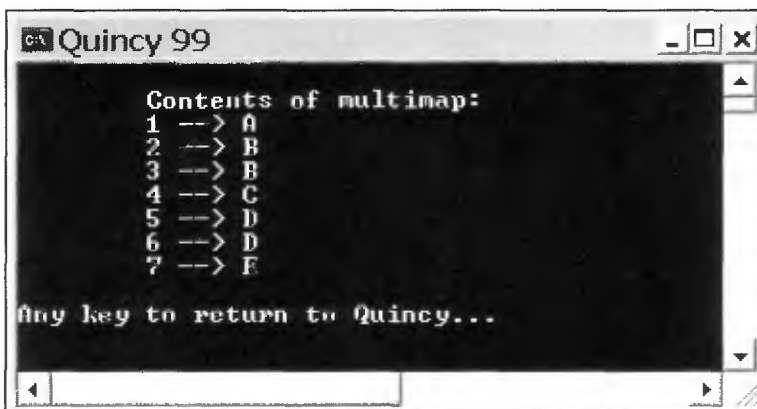
```

int main( )
{
    // Create the multimap object.
    MYMAP charMultimap;

    // Populate the multimap with values.
    charMultimap.insert(MYMAP::value_type(1,'A'));
    charMultimap.insert(MYMAP::value_type(4,'C'));
    charMultimap.insert(MYMAP::value_type(2,'B'));
    charMultimap.insert(MYMAP::value_type(7,'E'));
    charMultimap.insert(MYMAP::value_type(5,'D'));
    charMultimap.insert(MYMAP::value_type(3,'B'));
    charMultimap.insert(MYMAP::value_type(6,'D'));

    // Display the contents of the multimap.
    cout << "\n\tContents of multimap:\n";
    MYMAP::iterator iter;
    for (iter= charMultimap.begin( ); iter != charMultimap.end( ); iter++)
    {
        cout << "\t" << (*iter).first << " --> ";
        cout << (*iter).second << endl;
    }
    return 0;
}

```



```

Quincy 99
Contents of multimap:
1 --> A
2 --> B
3 --> B
4 --> C
5 --> D
6 --> D
7 --> E

Any key to return to Quincy...

```

၀ (၁၅. ၁၇)

## Removing multimap Elements

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1518.cpp program ဟာဆိုရင် ABBCDDE ဆိုတဲ့ multimap တစ်ခုကို create လုပ်ပြီး second element 'B' ကို erase လုပ်ခြင်းအားဖြင့် new set ဟာ ABCDDE ပဲကျန်အောင် လုပ်ပေးတဲ့ program ဖြစ်ပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 15.18: Removing multimap elements

```
#include <iostream>
```

```
#include <map>
```

```
typedef multimap<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    MYMAP charMultimap;
```

```
    charMultimap.insert(MYMAP::value_type(1,'A'));
```

```
    charMultimap.insert(MYMAP::value_type(4,'C'));
```

```
    charMultimap.insert(MYMAP::value_type(2,'B'));
```

```
    charMultimap.insert(MYMAP::value_type(7,'E'));
```

```
    charMultimap.insert(MYMAP::value_type(5,'D'));
```

```
    charMultimap.insert(MYMAP::value_type(3,'B'));
```

```
    charMultimap.insert(MYMAP::value_type(6,'D'));
```

```
// Display the contents of the multimap.
```

```
cout << "\n\tContents of multimap:\n";
```

```
MYMAP::iterator iter;
```

```
for (iter= charMultimap.begin( ); iter != charMultimap.end( ); iter++)
```

```
{
```

```
    cout << "\t" << (*iter).first << " --> ";
```

```
    cout << (*iter).second << endl;
```

```
}
```

```
// Erase the multimap's second element.
```

```
iter = charMultimap.begin( );
```

```

charMultimap.erase(++iter);
// Display the new contents of the multimap.
cout << "\n\tContents of new multimap: " << endl;
for (iter= charMultimap.begin( );iter != charMultimap.end( ); iter++)
{
    cout << '\t' << (*iter).first << " --> ";
    cout << (*iter).second << endl;
}
return 0;
}

```

```

Quincy 99
Contents of multimap:
1 --> A
2 --> B
3 --> B
4 --> C
5 --> D
6 --> D
7 --> E

Contents of new multimap:
1 --> A
3 --> B
4 --> C
5 --> D
6 --> D
7 --> E

Any key to return to Quincy...

```

ပုံ (၁၅. ၁၈)

## Searching a multimap

၁။ Ex1519.cpp program ဟာဆိုရင် multimap object တစ်ခုကကြိုက်တဲ့ element တစ်ခုကိုရှာပေးတဲ့ program ဖြစ်ပါတယ်။ ရှာချင်တဲ့ element ကိုရှာတွေ့ရင် Element found: လို့ကြေငြာပေးမှာပါ။ နောက်ဦး

Next element: ကဘာလို့ ဆက်ကြေငြာပါလိမ့်မယ်။ ရှာခိုင်းတဲ့ element ကိုမတွေ့ဘူးဆိုရင် Element not found: လို့ကြေငြာပေးမှာပါ။

// Listing 15.19: Searching a multimap

```
#include <iostream>
```

```
#include <map>
```

```
typedef multimap<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    MYMAP charMultimap;
```

```
    charMultimap.insert(MYMAP::value_type(1,'A'));
```

```
    charMultimap.insert(MYMAP::value_type(4,'C'));
```

```
    charMultimap.insert(MYMAP::value_type(2,'B'));
```

```
    charMultimap.insert(MYMAP::value_type(7,'E'));
```

```
    charMultimap.insert(MYMAP::value_type(5,'D'));
```

```
    charMultimap.insert(MYMAP::value_type(3,'B'));
```

```
    charMultimap.insert(MYMAP::value_type(6,'D'));
```

```
    cout << "\n\tContents of multimap:\n";
```

```
    MYMAP::iterator iter;
```

```
    for (iter= charMultimap.begin( ); iter != charMultimap.end( ); iter++)
```

```
    {
```

```
        cout << '\t' << (*iter).first << " --> ";
```

```
        cout << (*iter).second << endl;
```

```
    }
```

```
    cout << endl;
```

```
    // Find the first D.
```

```
    iter = charMultimap.find(5);
```

```
    if (iter == charMultimap.end())
```

```
        cout << "\tElement not found.";
```

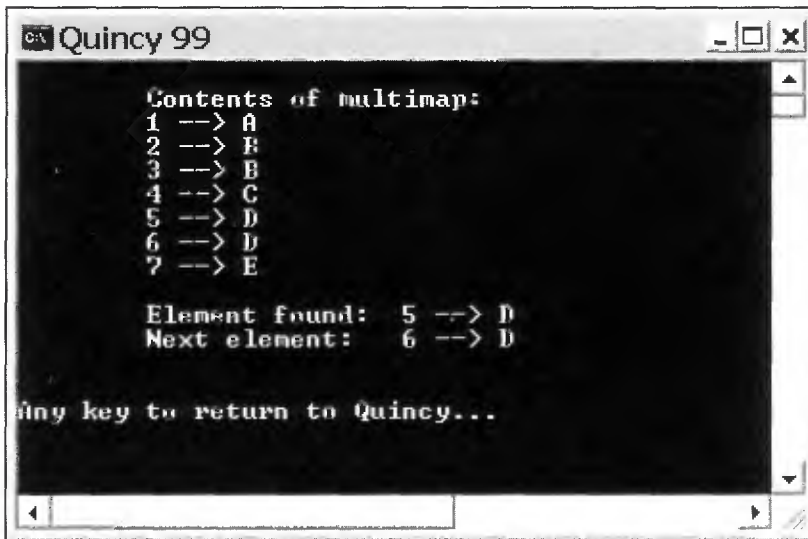
```
    else
```

```
    {
```

```

        cout << "\\tElement found: ";
        cout << '\\t' << (*iter).first << " --> ";
        cout << (*iter++).second << endl;
        cout << "\\tNext element: ";
        cout << '\\t' << (*iter).first << " --> ";
        cout << (*iter).second << endl;
    }
    cout << endl;
    return 0;
}

```



ပုံ (၁၅. ၁၉)

## Comparing multimaps

Ex1520.cpp program ဟာဆိုရင် set object (2) ခုကိုနှိုင်းယှဉ်ခိုင်းတဲ့ program တစ်ခုဖြစ်ပါတယ်။ program ကို run လိုက်မယ်ဆိုရင် ABBCDDE နဲ့ CDEFFGE ဆိုတဲ့ multimap object (2) ခုပေါ်လာပြီး multimap1 က multimap2 ထက်ငယ်တယ်လို့ ကွန်ပျူတာမှာ display လုပ်ပြပါလိမ့်မယ်။ ပုံ (၁၅. ၂၀) ကိုကြည့်ပါ။



// Listing 15.20: Comparing multimaps

```
#include <iostream>
```

```
#include <map>
```

```
typedef multimap<int, char> MYMAP;
```

```
int main( )
```

```
{
```

```
    // Create the first multimap object.
```

```
    MYMAP charMultimap;
```

```
    charMultimap.insert(MYMAP::value_type(1,'A'));
```

```
    charMultimap.insert(MYMAP::value_type(4,'C'));
```

```
    charMultimap.insert(MYMAP::value_type(2,'B'));
```

```
    charMultimap.insert(MYMAP::value_type(7,'E'));
```

```
    charMultimap.insert(MYMAP::value_type(5,'D'));
```

```
    charMultimap.insert(MYMAP::value_type(3,'B'));
```

```
    charMultimap.insert(MYMAP::value_type(6,'D'));
```

```
    cout << "\n\tContents of first multimap:\n";
```

```
    MYMAP::iterator iter;
```

```
    for (iter= charMultimap.begin( ); iter != charMultimap.end( ); iter++)
```

```
    {
```

```
        cout << '\t' << (*iter).first << " --> ";
```

```
        cout << (*iter).second << endl;
```

```
    }
```

```
    cout << endl;
```

```
    // Create the second multimap object.
```

```
    MYMAP charMultimap2;
```

```
    charMultimap2.insert(MYMAP::value_type(1,'C'));
```

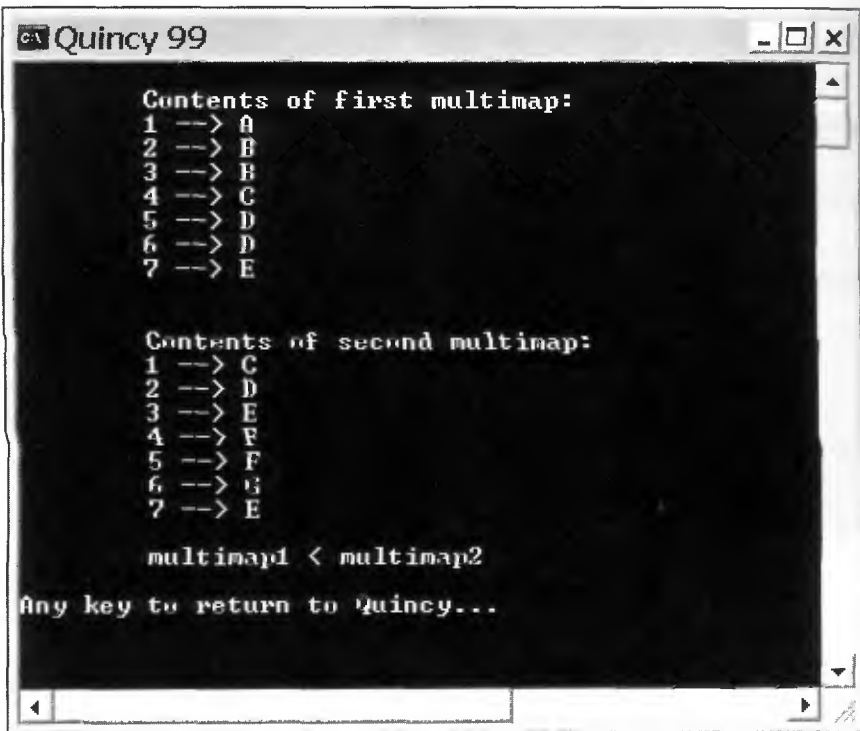
```
    charMultimap2.insert(MYMAP::value_type(4,'F'));
```

```
    charMultimap2.insert(MYMAP::value_type(2,'D'));
```

```
    charMultimap2.insert(MYMAP::value_type(7,'E'));
```

```
    charMultimap2.insert(MYMAP::value_type(5,'F'));
```

```
    charMultimap2.insert(MYMAP::value_type(3,'E'));
```



```
Quincy 99
Contents of first multimap:
1 --> A
2 --> B
3 --> B
4 --> C
5 --> D
6 --> D
7 --> E

Contents of second multimap:
1 --> C
2 --> D
3 --> E
4 --> F
5 --> F
6 --> G
7 --> E

multimap1 < multimap2
Any key to return to Quincy...
```

◊ (◊◊. ◊◊)

```
charMultimap2.insert(MYMAP::value_type(6,'G'));
```

```
cout << "\n\tContents of second multimap:\n";
```

```
for (iter= charMultimap2.begin( ); iter != charMultimap2.end( ); iter++)
{
    cout << '\t' << (*iter).first << " --> ";
    cout << (*iter).second << endl;
}
cout << endl;
```

```
// Compare the multimaps.
```

```
if (charMultimap == charMultimap2)
    cout << "\tmultimap1 == multimap2";
else if (charMultimap < charMultimap2)
```

```

        cout << "\tmultimap1 < multimap2";
    else
        cout << "\tmultimap1 > multimap2";

    cout << endl;
    return 0;
}

```

## ၁၅.၅ User-Defined Predicates

Ex1521.cpp program ဟာ compare ဆိုတဲ့ predicate ကို define လုပ်ပြီး map object ထဲက element တွေကို compare\_and\_sort လုပ်ပေးသွားတဲ့ program ဖြစ်ပါတယ်။ predicate ဟာ class အသေးစားပုံစံမျိုးပါ။ predicate name က compare ဖြစ်ပြီး overloaded ( ) operator ကိုအသုံးပြုသွားပါတယ်။ program ကို run လိုက်မယ်ဆိုရင် ပုံ (၁၅. ၂၁) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။

```

// Listing 15.21: User-defined predicates
#include <iostream>
#include <map>

class compare
{
public:
    bool operator( )(const int c1, const int c2) const
    {
        return c1 < c2;
    }
};

int main( )
{
    // Create the map object.
    map<int, char, compare> charMap;
}

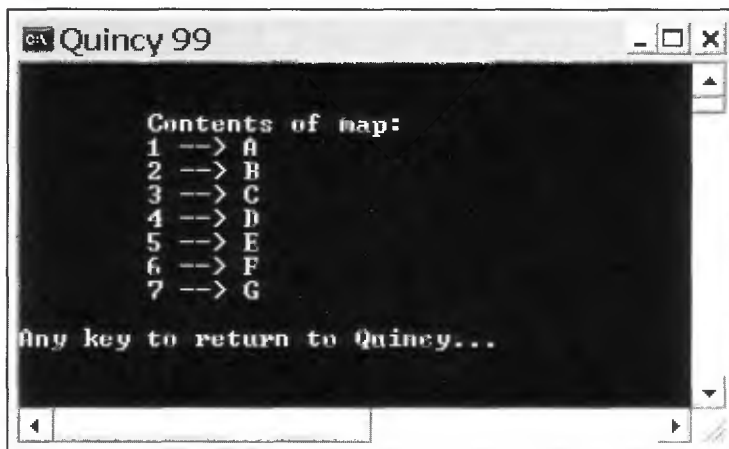
```

```

// Populate the map with values.
charMap.insert(map<int, char>::value_type(6,'F'));
charMap.insert(map<int, char>::value_type(2,'B'));
charMap.insert(map<int, char>::value_type(7,'G'));
charMap.insert(map<int, char>::value_type(4,'D'));
charMap.insert(map<int, char>::value_type(3,'C'));
charMap.insert(map<int, char>::value_type(5,'E'));
charMap.insert(map<int, char>::value_type(1,'A'));

// Display the contents of the map.
cout << endl << "\n\tContents of map:\n";
map<int, char>::iterator iter;
for (iter= charMap.begin( );iter != charMap.end( ); iter++)
{
    cout << '\t' << (*iter).first << " --> ";
    cout << (*iter).second << endl;
}
return 0;
}

```



၁၄၂ (၁၅.၁၀)

# Chapter 16



Standard Template Library မှာထည့်သွင်းပေးထားတဲ့ container တွေကိုအသုံးပြုပြီး type အမျိုးမျိုးကတန်ဖိုးတွေနဲ့ ကိုင်တွယ်လုပ်ကိုင်တာကို ကျွန်တော်တို့ ရှေ့ပိုင်းမှာလေ့လာခဲ့ပြီးပါပြီ။ အခုတင်ပြမှာကတော့ အဲဒါထက်အသုံးပြုလို့လွယ်ကူတဲ့ generic algorithm တွေအသုံးပြုနည်းဖြစ်ပါတယ်။ STL မှာ generic algorithm တွေကို အုပ်စု (4) ခုခွဲပြီးထည့်သွင်းထားပါတယ်။ အဲဒါတွေက (၁) Non-modifying sequence algorithms (၂) Mutating sequence algorithms (၃) Sorting algorithms နဲ့ (၄) Numeric algorithm တို့ဖြစ်ကြပါတယ်။ ပထမဆုံး non-modifying sequence algorithms ကိုလေ့လာကြည့်မယ်ဆိုရင် sequence ကို modify မလုပ်ဘဲ function apply လုပ်တာပဲလုပ်မှာပါ။ ဥပမာ `count( )` function အသုံးပြုပြီး sequence တစ်ခုမှာပါဝင်တဲ့ element အရေအတွက်ကို ရေတွက်ခိုင်းတာမျိုးပေါ့။ ဒါဟာ sequence ကိုပြုပြင်ပြောင်းလဲတာ မဟုတ်ပါဘူး။ STL မှာ non-modifying sequence algorithm (9) မျိုးကိုထည့်ထားပေးပါတယ်။ အဲဒါတွေက `adjacent_find( )`၊ `find( )`၊ `find_end( )`၊ `find_first( )`၊ `count( )`၊ `mismatch( )`၊ `equal( )`၊ `for_each( )` နဲ့ `search( )` algorithm တွေဖြစ်ကြပါတယ်။ အဲဒီထဲကတစ်ချို့ကို လက်တွေ့လေ့လာကြည့်ရအောင်။

## ၁၆.၁ Using the adjacent\_find( ) Function

၁။ Ex1601.cpp program ဟာဆိုရင် set object တစ်ခုမှာပါဝင်တဲ့ တန်ဖိုးတွေထဲက တူတဲ့ value တွေကို adjacent\_find( ) generic algorithm အသုံးပြုပြီး တစ်စုံပြီးတစ်စုံရှာပေးသွားတဲ့ program ဖြစ်ပါတယ်။ ဒီ algorithm ဟာ set object element တွေကို modify မလုပ်ပါဘူး။ ရှာရုံပဲရှာတာပါ။

// Listing 16.1: Using adjacent\_find( ) function

```
#include <iostream>
```

```
#include <set>
```

```
#include <algorithm>
```

```
int main( )
```

```
{
```

```
    // Create the set object.
```

```
    multiset<int, less<int> > intSet;
```

```
    intSet.insert(10);
```

```
    intSet.insert(3);
```

```
    intSet.insert(1);
```

```
    intSet.insert(3);
```

```
    intSet.insert(8);
```

```
    intSet.insert(8);
```

```
    intSet.insert(5);
```

```
    // Display the contents of the set.
```

```
    cout << "\n\tContents of set:\n";
```

```
    multiset<int, less<int> >::iterator it = intSet.begin( );
```

```
    for (int x=0; x < intSet.size( ); ++x)
```

```
        cout << '\t' << *it++ << endl;
```

```
    cout << endl;
```

```
    // Find the first pair of equal values.
```

```
    cout << "\n\tFirst matching pair:\n";
```

```
    it = adjacent_find (intSet.begin( ), intSet.end( ));
```

```

cout << '\t' << *it++ << endl;
cout << '\t' << *it << endl << endl;

// Find the second pair of equal values.
cout << "\n\tSecond matching pair:\n";
it = adjacent_find (it, intSet.end());
cout << '\t' << *it++ << endl;
cout << '\t' << *it << endl;
return 0;
}

```

Ex1601.cpp program ကို run လိုက်မယ်ဆိုရင် set element (7) ခုကို create အရင်လုပ်ပြီး အဲဒီထဲကတန်ဖိုးတူတဲ့ 3 နဲ့ 8 ဝဏန်း (2) စုံကို adjacent\_find( ) algorithm အသုံးပြုပြီး ရှာပေးသွားတာကို ပုံ (၁၆. ၁) မှာပြထားတဲ့အတိုင်း မြင်ရပါလိမ့်မယ်။

```

Quincy 99
Contents of set:
1
3
3
5
8
8
10

First matching pair:
3
3

Second matching pair:
8
8

Any key to return to Quincy...

```

ပုံ (၁၆. ၁)

## ၁၆.၂ Using the count( ) Function

၁။ အောက်မှာဖော်ပြထားတဲ့ Ex1602.cpp program ဟာ set object တစ်ခုမှာပါဝင်တဲ့ သတ်မှတ်ထားတဲ့ element တစ်ခုရဲ့ အရေအတွက်ကိုရှာပေးတဲ့ program ပါ။ count( ) algorithm အသုံးပြုပုံကို လေ့လာကြည့်ပါ။

// Listing 16.2: Using the count( ) function

```
#include <iostream>
```

```
#include <set>
```

```
#include <algorithm>
```

```
int main( )
```

```
{
```

```
    // Create the set object.
```

```
    multiset<int, less<int> > intSet;
```

```
    intSet.insert(10);
```

```
    intSet.insert(8);
```

```
    intSet.insert(1);
```

```
    intSet.insert(3);
```

```
    intSet.insert(8);
```

```
    intSet.insert(8);
```

```
    intSet.insert(5);
```

```
    cout << "\n\tContents of set:\n";
```

```
    multiset<int, less<int> >::iterator it = intSet.begin( );
```

```
    for (int x=0; x < intSet.size( ); ++x)
```

```
        cout << '\t' << *it++ << endl;
```

```
    cout << endl;
```

```
    // Count the number of 8s in the set.
```

```
    int cnt = count(intSet.begin( ), intSet.end( ), 8);
```

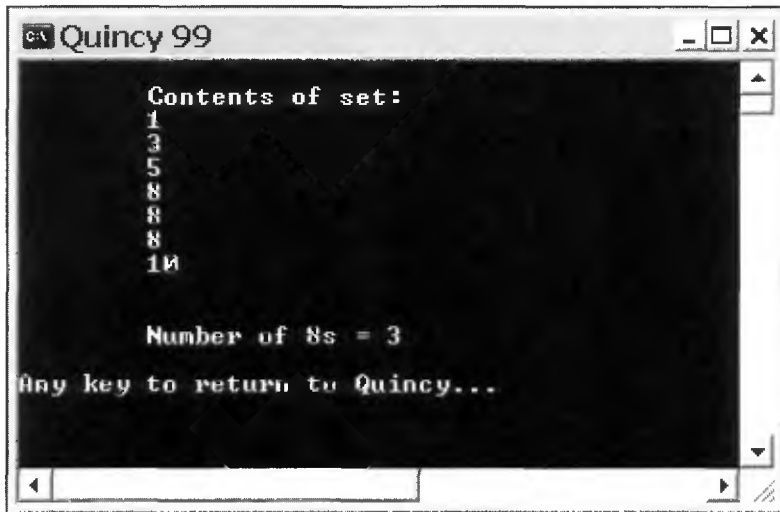
```
    cout << "\n\tNumber of 8s = " << cnt << endl;
```

```
    return 0;
```

```
}
```



၂။ Ex1602.cpp program ကို run လိုက်မယ်ဆိုရင် create လုပ်ထားတဲ့ sequence မှာ (8) ဂဏန်း ဘယ်နှစ်လုံးပါလဲဆိုတာကို ရေတွက်ပေးမှာဖြစ်ပါတယ်။ ပုံ (၁၆.၂) ကိုကြည့်ပါ။



ပုံ (၁၆.၂)

## ၁၆.၃ Using the for\_each( ) Function

Ex1603.cpp program ဟာ for\_each (start, end, func\_call) format ပုံစံကိုအသုံးပြုပြီး sequence တစ်ခုက element တွေကို sorted order နဲ့ display လုပ်ပြခိုင်းတဲ့ program ပါပဲ။ count( ) algorithm အသုံးပြုပုံကိုလေ့လာကြည့်ပါ။

// Listing 16.3: Using the for\_each( ) function

```
#include <iostream>
```

```
#include <set>
```

```
#include <algorithm>
```

```
void showVal(int val)
```

```
{
```

```

        cout << '\t' << val << endl;
    }

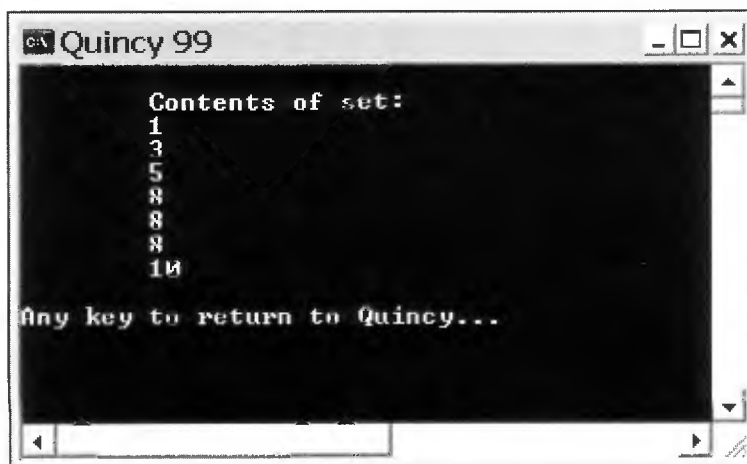
    int main( )
    {
        // Create the set object.
        multiset<int, less<int> > intSet;

        // Populate the set with values.
        intSet.insert(10);
        intSet.insert(8);
        intSet.insert(1);
        intSet.insert(3);
        intSet.insert(8);
        intSet.insert(8);
        intSet.insert(5);

        // Display the contents of the set.
        cout << "\n\tContents of set:\n";
        for_each(intSet.begin( ), intSet.end( ), showVal);

        return 0;
    }

```



0 (06. 9)

## ၁၆.၄ Using the fill( ) Function

sequence operation တွေကိုလုပ်ခြင်းအားဖြင့် container တွေ modify ဖြစ်သွားတဲ့ algorithm တွေကို Mutating sequence algorithm လို့ခေါ်ပါတယ်။ ဥပမာ sequence တစ်ခုရဲ့အပိုင်းတစ်ခုကို same sequence ရဲ့တစ်ခြားအပိုင်းနေရာမှာ copy ထည့်ပေးမယ်ဆိုရင် sequence operation ကြောင့် container content ဟာပြောင်းလဲသွားပြီမဟုတ်ပါလား။ Mutating sequence algorithm အုပ်စုမှာ ပါဝင်တာတွေကတော့ copy( )၊ copy\_backward( )၊ fill( )၊ generate( )၊ partition( )၊ random\_shuffle( )၊ swap( )၊ swap\_ranges( )၊ remove( )၊ replace( )၊ rotate( )၊ reverse( )၊ transform( ) နဲ့ unique( ) တို့ဖြစ်ကြပါတယ်။ Ex1604.cpp program မှာ fill( ) algorithm ကိုသုံးပြထားတာကို လေ့လာကြည့်ပါ။

// Listing 16.4: Using the fill( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
void showVal(int val)
```

```
{
```

```
    cout << '\t' << val << endl;
```

```
}
```

```
int main( )
```

```
{
```

```
    vector<int> intVector;           // Create the vector object.
```

```
    for (int x=0; x<6; ++x)
```

```
        intVector.push_back(x);
```

```
    cout << "\n\tContents of vector:\n";
```

```
    for_each(intVector.begin( ), intVector.end( ), showVal);
```

```
    // Fill vector with zeroes.
```

```
    fill(intVector.begin( ), intVector.begin( ) + 4, 0);
```

```
    // Display the contents of the new vector.
```

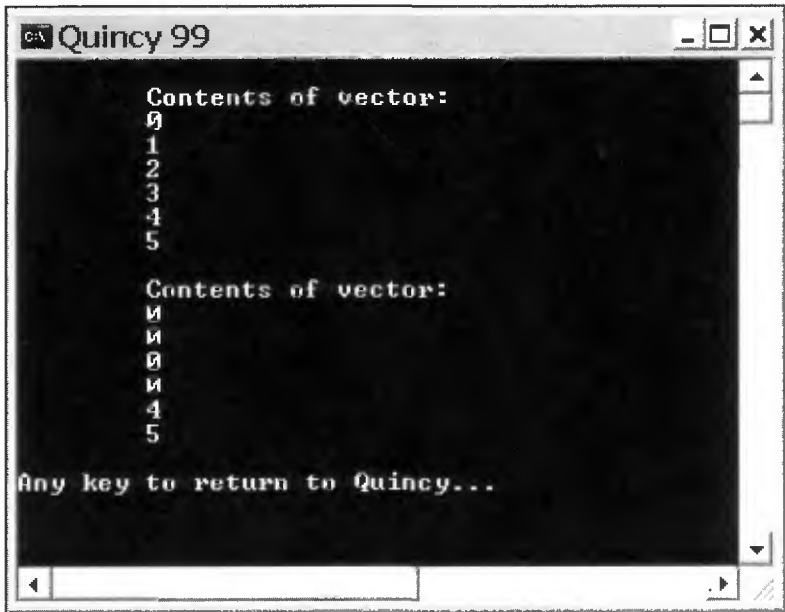
```
    cout << "\n\tContents of vector:\n";
```

```

    for_each (intVector.begin( ), intVector.end( ), showVal);
    return 0;
}

```

၂။ Ex1604.cpp program ကို run လိုက်မယ်ဆိုရင် create လုပ်ထားတဲ့ sequence က ပထမ (4) လုံးနေရာမှာ သုညဂဏန်းတွေ fill ဖြစ်သွားတာကို တွေ့ရပါလိမ့်မယ်။ ပုံ (၁၆.၄) ကိုကြည့်ပါ။



ပုံ (၁၆.၄)

## ၁၆.၅ Using the random\_shuffle( ) Function

၁။ Ex1605.cpp program ဟာ sequence တစ်ခုမှာပါဝင်တဲ့ value တွေကို random\_shuffle( ) generic algorithm အသုံးပြုပြီး တည်နေရာကိုလျှောက်ပြောင်းပေးသွားတဲ့ program ဖြစ်ပါတယ်။ algorithm ဟာ sequence element တွေကို modify လုပ်ပစ်လိုက်တာကိုတွေ့မှာပါ။

// Listing 16.5: Using the random\_shuffle( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
void showVal(int val)
```

```
{
```

```
    cout << '\t' << val << endl;
```

```
}
```

```
int main( )
```

```
{
```

```
    // Create the vector object.
```

```
    vector<int> intVector;
```

```
    // Populate the vector with values.
```

```
    for (int x=0; x<7; ++x)
```

```
        intVector.push_back(x);
```

```
    // Display the contents of the vector.
```

```
    cout << "\n\tContents of vector:\n";
```

```
    for_each (intVector.begin( ), intVector.end( ), showVal);
```

```
    // Shuffle the vector.
```

```
    random_shuffle (intVector.begin( ), intVector.end( ));
```

```
    // Display the contents of the new vector.
```

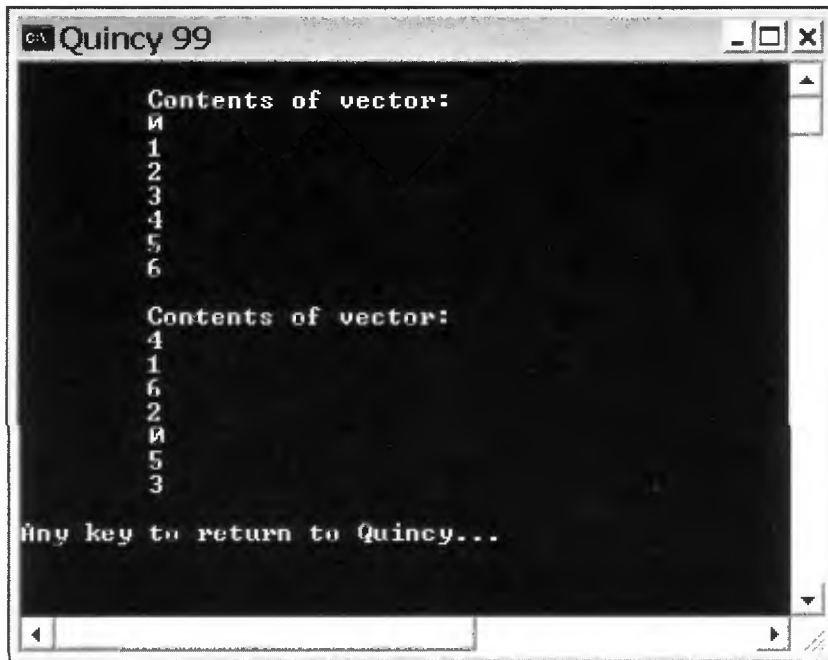
```
    cout << "\n\tContents of vector:\n";
```

```
    for_each (intVector.begin( ), intVector.end( ), showVal);
```

```
    return 0;
```

```
}
```

Ex1605.cpp program ကို run လိုက်မယ်ဆိုလို့ရှိရင် ပုံ (၁၆.၅) မှာပြထားတဲ့အတိုင်းမြင်ရမှာပါ။ sequence ဟာ modify မဖြစ်ခင်နဲ့ ဖြစ်သွားပြီးတဲ့အနေအထား (2) ခုကိုနှိုင်းယှဉ်ပြထားပါတယ်။



ပုံ (၁၆. ၅)

## ၁၆.၆ Using the partition( ) Function

၁။ Ex1606.cpp program ဟာ sequence တစ်ခုမှာပါဝင်တဲ့ element တွေထဲက သတ်မှတ်ထားတဲ့ element အုပ်စုတစ်ခုကို သီးသန့်ခွဲထုတ်ပြီး sequence ရဲ့အပိုင်းတစ်ပိုင်းမှာ စုထားပေးတဲ့ program ဖြစ်ပါတယ်။ partition( ) algorithm အသုံးပြုနည်းကို အောက်ပါ program မှာလေ့လာကြည့်ပါ။

// Listing 16.6: Using the partition( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
void showVal(int val)
```

```
{    cout << '\t' << val << endl;    }
```

```

bool equals5 (int val)
{
    return (val == 5);
}

int main( )
{
    // Create the vector object
    vector<int> intVector;

    // Populate the vector with values
    intVector.push_back(8);
    intVector.push_back(5);
    intVector.push_back(1);
    intVector.push_back(7);
    intVector.push_back(5);
    intVector.push_back(2);
    intVector.push_back(5);

    cout << "\n\tContents of vector:\n";
    for_each (intVector.begin( ), intVector.end( ), showVal);

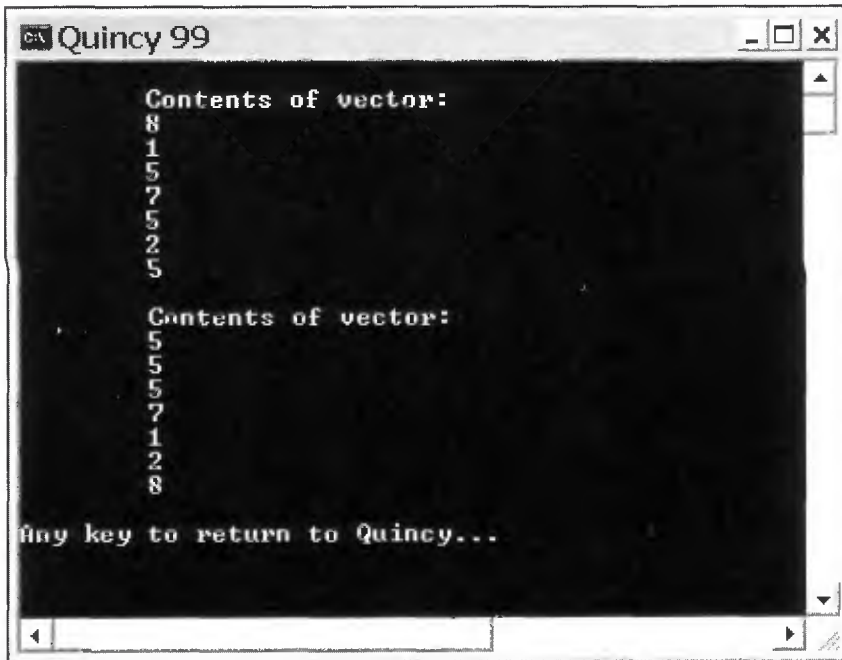
    // Partition the vector.
    partition (intVector.begin( ), intVector.end( ), equals5);

    // Display the contents of the new vector.
    cout << "\n\tContents of vector:\n";
    for_each (intVector.begin( ), intVector.end( ), showVal);

    return 0;
}

```

Ex1606cpp program ကို run လိုက်မယ်ဆိုရင် create လုပ်ထားတဲ့ sequence element တွေထဲက (5) ဂဏန်းကိုပဲခွဲထုတ်ပြီး sequence ရဲ့အစပိုင်းမှာ စုထားပေးပါတယ်။ partition နေရာကတော့ပထမဆုံး (5) ဂဏန်းနဲ့ကပ်နေတဲ့ (7) ဂဏန်းဖြစ်ပါတယ်။ (7) ကလွဲရင် 1 2 8 ဂဏန်းတွေဟာ sorted ဖြစ်နေပါပြီ။ ပုံ (၁၆. ၆) ကိုကြည့်ပါ။



ပုံ (၁၆. ၆)

## ၁၆.၇ Using the rotate( ) Function

၁။ Ex1607.cpp program ဟာဆိုရင် sequence တစ်ခုမှာပါဝင်တဲ့ element အုပ်စုတစ်ခုကို သတ်မှတ်ထားတဲ့နေရာတစ်ခုမှာ ဗဟိုပြုပြီး အုပ်စုလိုက်ပြောင်းပြန်ဖြစ်အောင်လှည့်ပေးတဲ့ program ဖြစ်ပါတယ်။ rotate( ) algorithm အသုံးပြုနည်းကို အောက်ပါ program မှာလေ့လာကြည့်ပါ။

```
// Listing 16.7: Using the rotate( ) function
#include <iostream>
#include <vector>
#include <algorithm>
```

```
void showVal(char val)
{    cout << '\t' << val << endl;    }
```



```

int main( )
{
    // Create the vector object.
    vector<char> charVector;

    // Populate the vector with values.
    charVector.push_back('T');
    charVector.push_back('H');
    charVector.push_back('E');
    charVector.push_back('R');
    charVector.push_back('E');
    charVector.push_back(' ');
    charVector.push_back('H');
    charVector.push_back('I');
    charVector.push_back(' ');

    // Display the contents of the vector.
    cout << "\n\tContents of vector:\n";
    for_each (charVector.begin( ), charVector.end( ), showVal);

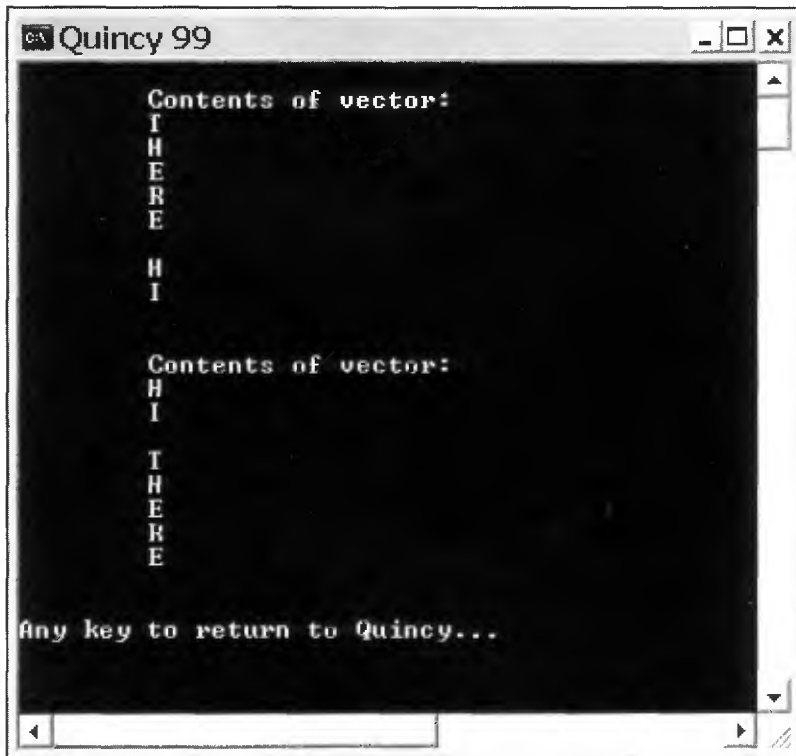
    // Rotate the vector.
    rotate (charVector.begin( ),charVector.begin( )+6, charVector.end( ));

    // Display the contents of the new vector.
    cout << "\n\tContents of vector:\n";
    for_each (charVector.begin( ), charVector.end( ), showVal);

    return 0;
}

```

၂။ Ex1607cpp program ကို run လိုက်မယ်ဆိုရင် create လုပ်ထားတဲ့ THERE HI sequence ထဲက (6) ခုမြောက် element နေရာကိုဗဟိုပြုပြီး HI THERE sequence ဖြစ်အောင် rotate လုပ်ပေးတာကို တွေ့ရမှာပါ။ ပုံ (၁၆.၇) ကိုကြည့်ပါ။



ပုံ (၁၆.၇)

## ၁၆.၈ Using the sort( ) Function

အခုတစ်ခါတင်ပြမှာက Sorting algorithm တွေမှာပါဝင်တဲ့ sort( ) ၊ partial\_sort( ) merge( ) အစရှိတဲ့ algorithm တွေကိုတင်ပြပါမယ်။ Ex1608.cpp program ဟာဆိုရင် sequence တစ်ခုမှာပါဝင်တဲ့ char element တွေကို sort( ) algorithm program အသုံးပြုပြီး A to Z စီပေးမှာပါ။ ပုံ (၁၆.၈) မှ Ex1608.cpp program ကို run ပြထားပါတယ်။

```
// Listing 16.8: Using the sort( ) function
#include <iostream>
#include <vector>
#include <algorithm>
```

```

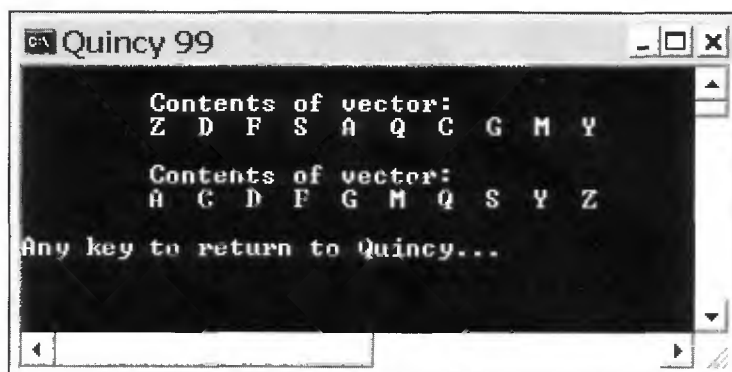
void showVal(char val)
{
    cout << val << " ";
}

int main( )
{
    // Create and populate the vector object
    vector<char> charVector;
    charVector.push_back('Z');
    charVector.push_back('D');
    charVector.push_back('F');
    charVector.push_back('S');
    charVector.push_back('A');
    charVector.push_back('Q');
    charVector.push_back('C');
    charVector.push_back('G');
    charVector.push_back('M');
    charVector.push_back('Y');
    cout << "\n\tContents of vector:\n\t";
    for_each (charVector.begin( ), charVector.end( ), showVal);

    sort (charVector.begin( ), charVector.end( ));

    // Display the contents of the new vector.
    cout << "\n\n\tContents of vector:\n\t";
    for_each (charVector.begin( ), charVector.end( ), showVal);
    cout << endl;
    return 0;
}

```



୦ (୦୫. ୦)

## ၁၆.၉ Using the `partial_sort( )` Function

Ex1609.cpp program ဟာဆိုရင် (10) char element ပါဝင်တဲ့ sequence တစ်ခုမှာ အပေါ်ပိုင်းက char (5) လုံးကိုပဲခွဲပြီး sort လုပ်ပေးတဲ့ program ဖြစ်ပါတယ်။ `partial_sort( )` algorithm အသုံးပြုပုံကို လေ့လာကြည့်ပါ။ ပုံ (၁၆. ၉) မှာ program ကို run ပြထားပါတယ်။

// Listing 16.9: Using the `partial_sort( )` function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <string>
```

```
void showVal(string val)
```

```
{      cout << '\t' << val << endl;      }
```

```
int main( )
```

```
{
```

```
    // Create the vector object.
```

```
    vector<string> strVector;
```

```
    // Populate the vector with values.
```

```
    strVector.push_back("Zebra");
```

```
    strVector.push_back("Deer");
```

```
    strVector.push_back("Fish");
```

```
    strVector.push_back("Snake");
```

```
    strVector.push_back("Bat");
```

```
    strVector.push_back("Cat");
```

```
    strVector.push_back("Bird");
```

```
    strVector.push_back("Turtle");
```

```
    strVector.push_back("Horse");
```

```
    strVector.push_back("Cow");
```

```

// Display the contents of the vector.
cout << "\n\tContents of vector:\n";
for_each (strVector.begin( ), strVector.end( ), showVal);
cout << endl;

// Sort the vector.
partial_sort (strVector.begin( ),strVector.begin() + 5, strVector.end( ));

// Display the contents of the new vector.
cout << "\n\tContents of vector:\n";
for_each (strVector.begin( ), strVector.end( ), showVal);
return 0;
}

```



୦ (୦୯.୧)

# Using the merge( ) Function

Ex1610.cpp program ဟာဆိုရင် vector object (2) ခုကို ပေါင်းပေးတဲ့ program ဖြစ်ပါတယ်။ merge လုပ်ပြီးရင် element တွေကို စီပေးဦးမှာပါ။ ပုံ (၁၆. ၁၀) မှာ program ကို run ပြထားပါတယ်။

// Listing 16.10: Using the merge( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <string>
```

```
void showVal (string val)
```

```
{
```

```
    cout << '\t' << val << endl;
```

```
}
```

```
int main( )
```

```
{
```

```
    // Create the vector objects.
```

```
    vector<string> strVector1;
```

```
    vector<string> strVector2;
```

```
    vector<string> strVector3(7);
```

```
    // Populate two vectors with values.
```

```
    strVector1.push_back("Zebra");
```

```
    strVector1.push_back("Deer");
```

```
    strVector1.push_back("Fish");
```

```
    strVector2.push_back("Cat");
```

```
    strVector2.push_back("Bird");
```

```
    strVector2.push_back("Turtle");
```

```
    strVector2.push_back("Horse");
```

```
    // Display the contents of the vectors.
```

```
    cout << "\n\tContents of vector1:\n";
```

```

for_each (strVector1.begin( ), strVector1.end( ), showVal);
cout << "\n\tContents of vector2:\n";
for_each (strVector2.begin( ), strVector2.end( ), showVal);

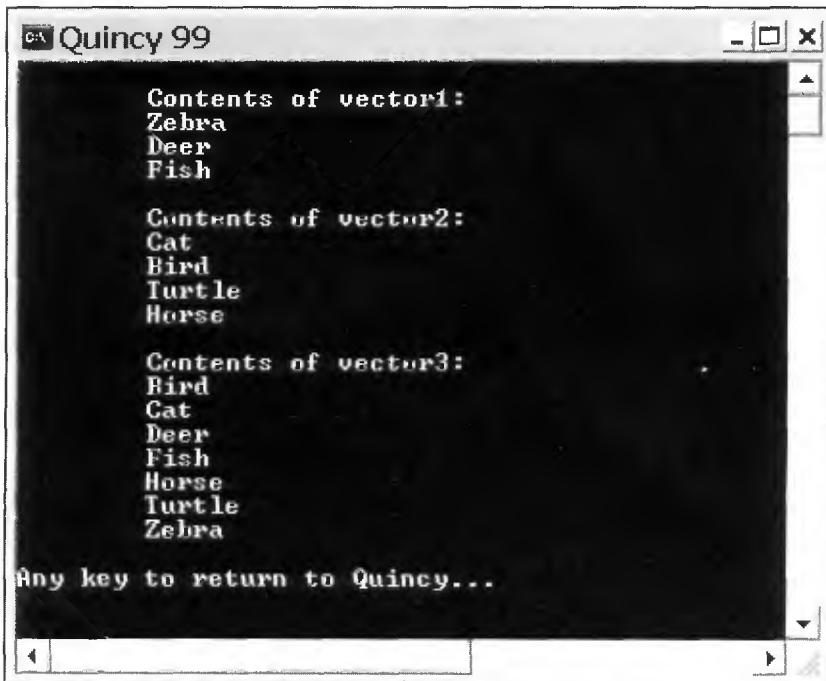
// Sort the vectors.
sort (strVector1.begin( ), strVector1.end( ));
sort (strVector2.begin( ), strVector2.end( ));

// Merge the sorted vectors.
merge (strVector1.begin( ), strVector1.end( ),
strVector2.begin( ), strVector2.end( ),
strVector3.begin( ));

// Display the contents of the new vector.
cout << "\n\tContents of vector3:\n";
for_each(strVector3.begin( ), strVector3.end( ), showVal);
return 0;

```

}



0 (06.00)

## More on Using the merge( ) Function

// Listing 16.11: More on using the merge( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <string>
```

```
void showVal(string val)
```

```
{
```

```
    cout << '\t' << val << endl;
```

```
}
```

```
int main( )
```

```
{
```

```
    // Create the vector objects.
```

```
    vector<string> strVector1;
```

```
    vector<string> strVector2;
```

```
    // Populate two vectors with values.
```

```
    strVector1.push_back("Zebra");
```

```
    strVector1.push_back("Deer");
```

```
    strVector1.push_back("Fish");
```

```
    strVector1.push_back("Snake");
```

```
    strVector1.push_back("Bat");
```

```
    strVector2.push_back("Deer");
```

```
    strVector2.push_back("Antelope");
```

```
    strVector2.push_back("Turtle");
```

```
    strVector2.push_back("Snake");
```

```
    strVector2.push_back("Fish");
```

```
    // Sort the vectors.
```

```
    sort (strVector1.begin( ), strVector1.end( ));
```

```
    sort (strVector2.begin( ), strVector2.end( ));
```



```

// Display the contents of the vectors.
cout << "\n\tContents of vector1:\n";
for_each (strVector1.begin( ), strVector1.end( ), showVal);
cout << endl;
cout << "\n\tContents of vector2:\n";
for_each (strVector2.begin( ), strVector2.end( ), showVal);
cout << endl;

// Search for the sorted range Deer, Fish, Snake.
bool result = includes(strVector1.begin( ), strVector1.end( ),
                      strVector2.begin( )+1, strVector2.begin()+3);
if (result)
    cout << "\tFound sorted range.\n";
else
    cout << "\tDid not find sorted range.\n";
return 0;
}

```

```

Quincy 99
Contents of vector1:
Bat
Deer
Fish
Snake
Zebra

Contents of vector2:
Antelope
Deer
Fish
Snake
Turtle

Found sorted range.

Any key to return to Quincy...

```

0 (06. 00)

## ၁၆.၁၁ Using the accumulate( ) Function

အခုနောက်ဆုံးတင်ပြမှာက accumulate( ) ၊ inner\_product( ) ၊ partial\_sum( ) ၊ adjacent\_difference( ) အစရှိတဲ့ Numeric algorithm တွေဖြစ်ပါတယ်။ Ex1612.cpp program ဟာဆိုရင် accumulate( ) algorithm ကိုအသုံးပြုပြီး vector object တစ်ခုကတန်ဖိုးတွေမှာ တန်ဖိုးတစ်ခုပေါင်းထည့်ပေးတဲ့ program ဖြစ်ပါတယ်။ ဝုံ (၁၆. ၁၂) မှာ program ကို run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 16.12: Using the accumulate( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <numeric>
```

```
void showVal (int val)
```

```
{    cout << '\t' << val << endl;    }
```

```
int main( )
```

```
{
```

```
    // Create and populate the vector object
```

```
    vector<int> intVec;
```

```
    for (int x=1; x<6; x++)
```

```
        intVec.push_back(x);
```

```
    // Display the contents of the vector
```

```
    cout << "\n\tContents of vector:\n";
```

```
    for_each (intVec.begin( ),intVec.end( ), showVal);
```

```
    cout << endl;
```

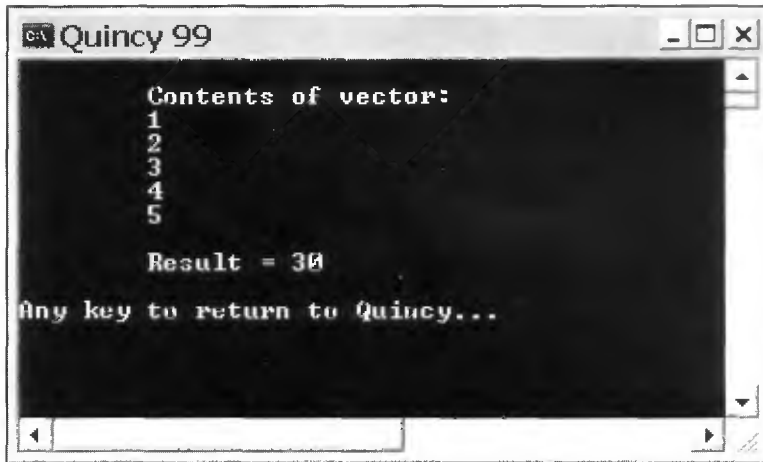
```
    // Calculate and display sum
```

```
    int result = accumulate(intVec.begin( ),intVec.end( ), 15);
```

```
    cout << "\tResult = " << result << endl;
```

```
    return 0;
```

```
}
```



ပုံ (၁၆. ၁၂)

## ၁၆.၁၂ Using the inner\_product( ) Function

အရေအတွက်တူတဲ့ vector object (2) ခု ဥပမာ {0,1,2,3,4} နဲ့ {2,3,4,5,6} တို့ကို inner product လုပ်ပေးချင်ရင်  $\{0*2+1*3+2*4+3*5+4*6\}$  လို့တွက်ပေးရပါတယ်။ အဲဒီဥပဒေကို Ex1613.cpp program မှာရေးပြီး ပုံ (၁၆. ၁၃) မှာ run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 16.13: Using the inner\_product( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <numeric>
```

```
void showVal (int val)
```

```
{
```

```
    cout << '\t' << val << endl;
```

```
}
```

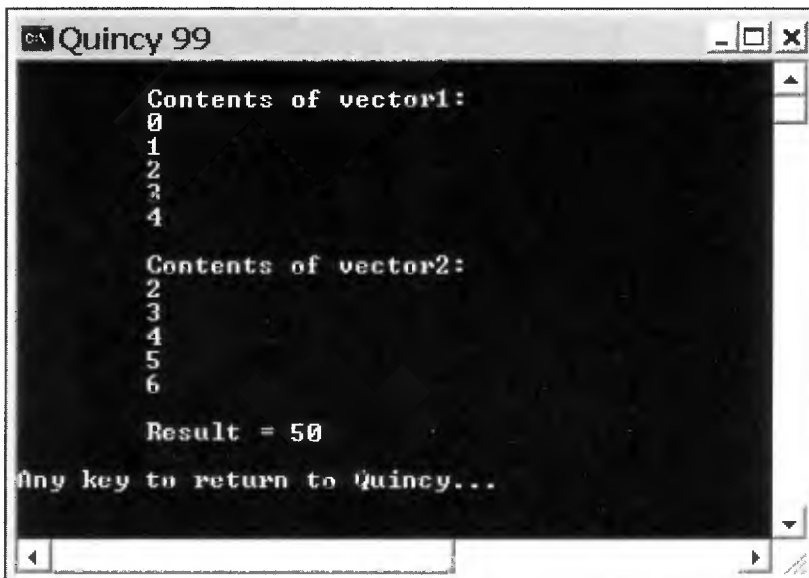
```

int main( )
{
    // Create and populate the two vector objects.
    vector<int> intVector1;
    vector<int> intVector2;
    for (int x=0; x<5; ++x)    intVector1.push_back(x);
    for (int x=2; x<7; ++x)    intVector2.push_back(x);

    // Display the contents of the vectors.
    cout << "\n\tContents of vector1:\n";
    for_each (intVector1.begin( ), intVector1.end( ), showVal);
    cout << "\n\tContents of vector2:\n";
    for_each (intVector2.begin( ),intVector2.end( ), showVal);
    cout << endl;

    // Calculate the inner product.
    int result = inner_product(intVector1.begin( ),
                               intVector1.end( ), intVector2.begin( ), 0);
    cout << "\tResult = " << result << endl;
    return 0;
}

```



```

Quincy 99
Contents of vector1:
0
1
2
3
4

Contents of vector2:
2
3
4
5
6

Result = 50

Any key to return to Quincy...

```

0 (06. 02)

## Using the `partial_sum( )` Function

Ex1614.cpp program ဟာဆိုရင် `vector` object တစ်ခု ဥပမာ  $\{2,3,4,5,6\}$  ကနေ `second` `vector` တစ်ခုကို `partial_sum( )` algorithm အသုံးပြုပြီး အခုလို  $\{2, 3+2, 4+(3+2), 5+(4+3+2, 6+(5+4+3+2)\}$  create လုပ်ယူသွားတာကို program ရေးပြထားတာပါ။ ပုံ (၁၆. ၁၄) မှာ program ကို run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 16.14: Using the `partial_sum( )` function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <numeric>
```

```
void showVal (int val)
```

```
{
```

```
    cout << '\t' << val << endl;
```

```
}
```

```
int main( )
```

```
{
```

```
    // Create the vector objects.
```

```
    vector<int> intVector1;
```

```
    vector<int> intVector2(4);
```

```
    // Populate the vector.
```

```
    for (int x=2; x<7; ++x) intVector1.push_back(x);
```

```
    // Display the contents of the vector.
```

```
    cout << "\n\tContents of vector1:\n";
```

```
    for_each (intVector1.begin( ), intVector1.end(), showVal);
```

```
    // Calculate the partial sum.
```

```
    partial_sum(intVector1.begin( ), intVector1.end( ), intVector2.begin( ));
```

```

// Display the contents of the resultant vector.
cout << "\n\tContents of vector2:\n";
for_each (intVector2.begin( ), intVector2.end( ), showVal);

return 0;
}

```

```

C:\ Quincy 99
Contents of vector1:
2
3
4
5
6
Contents of vector2:
2
5
9
14
20
Any key to return to Quincy...

```

ပုံ (၁၆. ၁၄)

## ၁၆.၁၄ Using the adjacent\_difference( ) Function

အခုနောက်ဆုံးတင်ပြမယ့် Ex1615.cpp program ဟာ vector object တစ်ခု ဥပမာ {3, 4, 12, 6, 10} ကနေ second vector တစ်ခုကို partial\_difference( ) algorithm အသုံးပြုပြီး အခုလို {3, 4-3, 12-4, 6-12, 10-6} create လုပ်ယူသွားတာကို program ရေးပြထားတာပါ။ Ex1615.cpp program ကို ပုံ (၁၆. ၁၅) မှာ run ပြထားပါတယ် ၊ လေ့လာကြည့်ပါ။

// Listing 16.15: Using the adjacent\_difference( ) function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <numeric>
```

```
void showVal(int val)
```

```
{
```

```
    cout << '\t' << val << endl;
```

```
}
```

```
int main( )
```

```
{
```

```
    // Create the vector objects.
```

```
    vector<int> intVector1;
```

```
    vector<int> intVector2(5);
```

```
    // Populate the vector.
```

```
    intVector1.push_back(3);
```

```
    intVector1.push_back(4);
```

```
    intVector1.push_back(12);
```

```
    intVector1.push_back(6);
```

```
    intVector1.push_back(10);
```

```
    // Display the contents of the vector.
```

```
    cout << "\n\tContents of vector1:\n";
```

```
    for_each(intVector1.begin( ),intVector1.end( ), showVal);
```

```
    cout << endl;
```

```
    // Calculate the partial sum.
```

```
    adjacent_difference(intVector1.begin( ),
```

```
        intVector1.end( ), intVector2.begin( ));
```

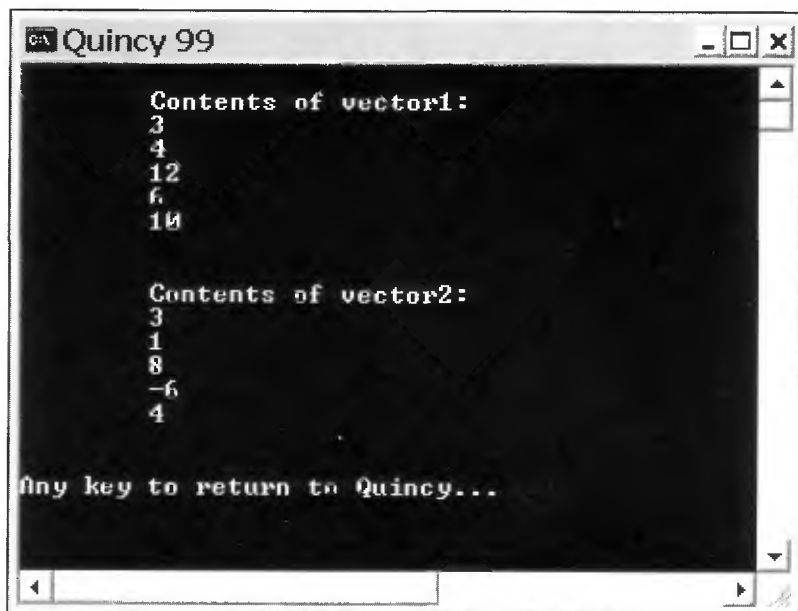
```
    // Display the contents of the resultant vector.
```

```
    cout << "\n\tContents of vector2:\n";
```

```
for_each (intVector2.begin( ),intVector2.end( ), showVal);  
cout << endl;
```

```
return 0;
```

```
}
```



```
Quincy 99  
Contents of vector1:  
3  
4  
12  
6  
10  
  
Contents of vector2:  
3  
1  
8  
-6  
4  
  
Any key to return to Quincy...
```

၁၆.၁၅