

C

+

+

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



**Complete**

13.7. The seek( ) Member Function	65
13.8. The tell( ) 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 <code>queue</code> Container Adaptor	104
14.7. The <code>priority_queue</code> Container Adaptor	106

## CHAPTER 15

# ASSOCIATIVE CONTAINERS

15.1. The <code>set</code> Class Template	108
15.2. The <code>multiset</code> Class Template	115
15.3. The <code>map</code> Class Template	124
15.4. The <code>multimap</code> Class Template	133
15.5. User-Defined Predicates	141



**Complete**

## CHAPTER 16

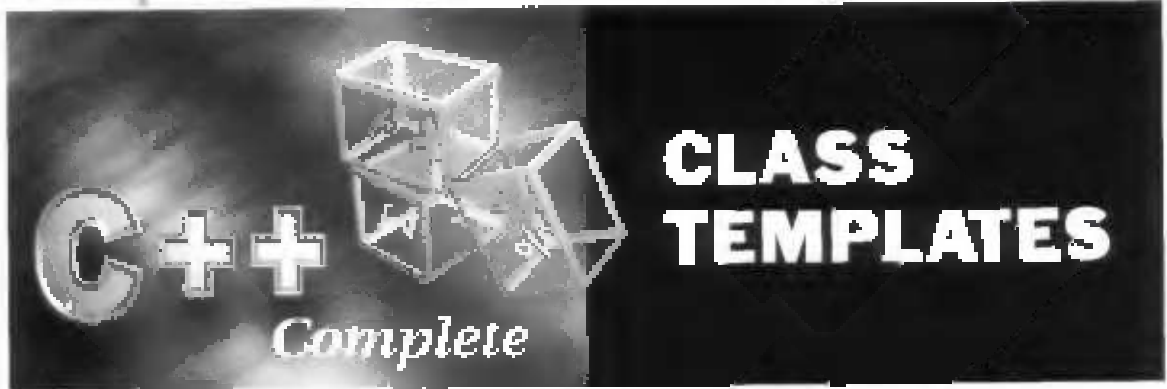
# GENERIC ALGORITHMS

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 တို့ကို create လုပ်ဆောင်ရာတွင် container ကဲ့သို့ပါး ခံနိုင် generic function ဆိုတာကတော့အားလုံး တူတူပဲ (data ပြောင်းဆဲလို့ပေါ်ပြီး ပာ သို့မဟုတ်နောက်ဆက်တွဲ function တစ်ခုဖြစ်ပုံကတော့) generic function တစ်ခုကို execute လုပ်မယ်ဆိုလို့ပဲ compiler ကနေ ဆင့်တင်တဲ့ data type ကိုအခြေအနေနဲ့ အသုံးပြုပေးမယ်။ ဒီဥပမာ generic function မဟုတ်ဘဲ အတိတ်ကတည်းက overload လုပ်လို့ပေးတာကတော့ပါ။ မဟုတ်ဘဲအားလုံး generic function တစ်ခုကို create လုပ်ဆောင် template ဆိုတဲ့ keyword ကိုအသုံးပြုရမယ်။ တစ်ခု ပြောရမယ့် template ဆိုတာ generic function တစ်ခု အားလုံးလုပ်ဆောင်ဆုံးတန်းပေါ်ပြီး framework တစ်ခုလုံးပြုလုပ် ပျက်စီးတဲ့ program တစ်ခုလုံးအားလုံးကို အားလုံးဆိုင်ရာလုပ်ဆောင် လုပ်ဆောင်တာကတော့ပါ။ compiler မှဲ့အားလုံးဖြစ်ပုံကတော့ template တစ်ခုလုံးပုံကတော့ အမှတ်ပါ။

```
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 = 110, i2 = 510;
    float  f1 = 25.55, f2 = 75.34;

    cout << "\n\nBEFORE SWAP "
         << "\n\n";
    cout << "i1: First integer number = " << i1
         << "\n\n";
    cout << "i2: Second integer number = " << i2
         << "\n\n";
    cout << "f1: First float number = " << f1
         << "\n\n";
    cout << "f2: Second float number = " << f2 << endl;

    swap (i1, i2);           // swapping integers
    swap (f1, f2);          // swapping floats

    cout << "\n\nAFTER SWAP "
         << "\n\n";
    cout << "i1: First integer number = " << i1
         << "\n\n";
    cout << "i2: Second integer number = " << i2
         << "\n\n";
    cout << "f1: First float number = " << f1
         << "\n\n";
    cout << "f2: Second float number = " << f2 << endl;

    return 0;
}
```





```

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

int main()
{
    cout << "Larger integer number = " << Larger(12,345)
    << "\nLarger float number = " << Larger(12.14, -56.73)
    << "\nLarger character = " << Larger('W', 'V')
    << endl;
    return 0;
}

```

Ex1102.cpp program  run                                  

```

Ex1103.cpp

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

#include <iostream>
using namespace std;

struct myTemp { int a; double b; };

int main()
{
    cout << "Enter a and b: ";
    myTemp t;
    t.a = 123456;
    t.b = 7.6789;

    cout << endl;

    myTemp t1(0, COMPLETE_C);
    myTemp t2(0, 123456, 7.6789);

    return 0;
}

```

Figure 11.3

Ex1103.cpp program is run as follows: Figure 11.4: Execution Results

```

Quincy OS
a = 0
b = COMPLETE_C

a = 123456
b = 7.6789

Any key to return to Quincy...

```

Figure 11.4





# A Simple Class Template

generic function  $\rightarrow$  define  $\rightarrow$  max/min  $\rightarrow$   $\rightarrow$  generic class  $\rightarrow$   $\rightarrow$  generic class  $\rightarrow$  generic class  $\rightarrow$  generic form  $\rightarrow$  max/min

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

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

```
template <class I1, class I2>
class Area
{
    I1 length;
    I2 width;

public:
    Area (I1 l, I2 w)
        : length = l, width = w {}

    float display()
    {
        cout << "length = " << length << " inches",
        cout << "width = " << width << " feet";
        return length*width;
    }
};
```

```

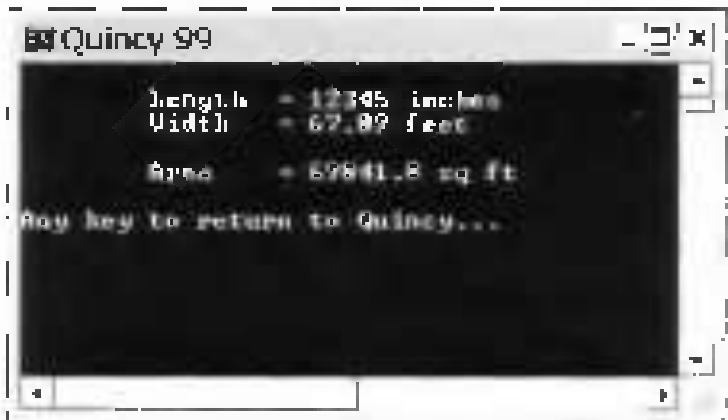
int main( )
{
    float    a = 12345;    // inches
    float    b = 67.89;   // feet

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

    return 0;
}

```

Ex1:05.cpp program ၏ run လုပ်ကိစ္စပုံစံနှင့် ပုံ (၁၁.၆) မှာပြထားတဲ့အတိုင်း ပြန်ရမှာပါ။



ပုံ (၁၁.၆)

## ၁၁.၄ Default Values for Parameters of a Specific Type

ရှေ့ပိုင်းကဲ့သို့ class template တစ်ခုကို create လုပ်တဲ့အခါမှာ template parameter list ထဲမှာ default argument အစီအစဉ် တစ်ခုခုပေးနိုင်အောင်အားပေးတို့ရန်အတွက် ပုံစံမှာ Ex1:06.cpp program ကာန်နော်

Ex1105.cpp program of (C++ Code Snippets): template parameter list of constructor argument of member of type `int` of value of 10 of constructor constructor

// Listing 11.6. Default values for parameters of a specific type  
#include <string>

```
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 << "\nLength = " << length << " inches";
```

```
        cout << "\nWidth = " << width << " feet";
```

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

```
    }
```

```
};
```

```
int main() {
```

```
{
```

```
    float a = 12145; // inches
```

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

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

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

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

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

```
    return 0;
```

```
}
```



Ex1106.cpp program ၏ ဆောင်ရွက်ပုံမှာ main() function မှာ  $a = 12.45$  ၊  $b = 67.89$  ၊ ၎် initialize လုပ်ပေးသော `mi1(a, b)` ၎် template function ၎် call ခံရပြီးနောက် ၎် class Area ၎်ကိုင်ရာ `length = l + add-default = 12.45 + 10 = 124.5 ၊ width = w + add-default = 67.89 + 10 = 77.89 ၎် assign လုပ်ပေးပါ default value ၎် ကိုယ်စားပြုပေးသော mi2(a, b) ၎် call ခံရပြီးနောက် length = l + add = 12345 + 100 = 12445 ၊ width = w + add = 57.85 + 100 = 157.85 ၎် assign လုပ်ပေးပါ။ template definition ၎် default value ၎် ကိုယ်စားပြုပေးပါ။ Ex1106.cpp program ၎် run ၎်ခံရင် နံနက် ၁၁:၀၀ ၎် မှတ်တမ်း ၎် ပြန်ရာ ၎်ပြန်ပါ။`



ဇ် (၁၁.၅)

## ၁၁.၅ Using a Linked-list Template

```

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

```

int main( )
{
  
```



```

// 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* preventry;
    ListEntry(T& entry);
    friend class LinkedList<T>;
};

template <class T>
// Construct a linked list entry.
ListEntry<T>::ListEntry(T &entry)
{
    thisentry = entry;
    nextentry = 0;
    preventry = 0;
}

template <class T>
// The linked list
class LinkedList
{
    // The list head.
    ListEntry<T>* firstentry;
    ListEntry<T>* lastentry;
    ListEntry<T>* kranke;
    void RemoveEntry(ListEntry<T> *nentry);
    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()
{
    iterstor = 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;
    entry = newentry;
}

```

```

template <class T>

```

```

// Remove an entry from the linked list.

```

```

void LinkedList<T>::RemoveEntry(ListEntry<T>* lentry)

```

```

{
    if (lentry == 0)    return;
    if (lentry == nullptr)
        return;
    delete lentry->preentry;

    // Repeat 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.

```

```

// LinkedList::operator[](int pos) const
{
    if (pos != -1)
    {
        iterator = firstEntry;

        if (!iterator)
        {
            while (pos > 0)
                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::PreEntry( )
{
    if (iterator == 0)
        iterator = lastentry;
    else
        iterator = iterator->preventry;

    return CurrentEntry();
}
}
#endif

```

Ex1207.cpp program to run: `g++ Ex1207.cpp -std=c++11 -c` and then `g++ Ex1207.o -std=c++11 -c`

```

Quincy 99
1 2 3 4 6 7 9
1 2 3 4 6 7 9
Any key to return to Quincy...

```

g++ Ex1207.cpp



# Partial Template Specialization

template partial specialization ဆိုတာ primary class template ရဲ့ အသံထွက်ပုံများထက်ပို၍ parameter ကွဲပြား ထည့်သွင်းတဲ့ special parameter အကွဲအပြားတွေကို template ရဲ့အသံထွက်ပုံများထက်ပို၍ primary class template ထက်ပို၍ object (2) နဲ့ parameter အကွဲအပြားနဲ့ pass ထည့်သွင်းနိုင်ဖို့ကို object အကွဲအပြားတွေမှာ display ထုတ်ဖော်ပေးနိုင်ဖို့ အသံထွက်ပုံမှာ object အကွဲအပြား char value ထည့်သွင်းနိုင်ဖို့ကိုပါ integer value မှာပဲ object ဖဲ့ ထည့်သွင်းအသံထွက်ပုံကို template ရဲ့ကို create လုပ်ပုံပုံကို ချိန်ညှိတဲ့ template partial specialization လို့ခေါ်ပါတယ်။ Ex1108. cpp program ကိုအောက်ဖော်ပြပါ။

// Listing 11.B: 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 << "My OBJECT DISPLAY\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 << "My OBJECT DISPLAY\n"
        << "-----\n"
        << "Object 1: " << obj1 << endl
        << "Object 2: " << obj2 << endl
        << endl;
}
}

```

};

```

int main()
{

```

```

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

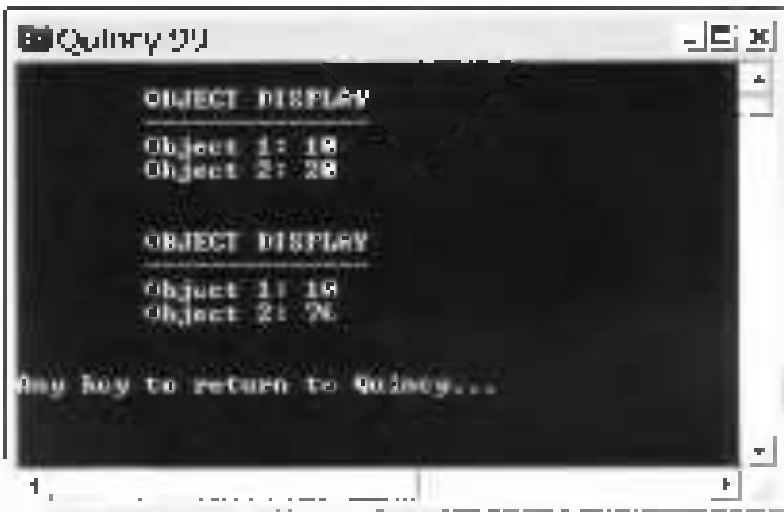
```

```

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

```

Ex1113.cpp program ⇒ run ⇒  (a x 2)



(a x 2)

# Chapter 12



BASIC language ကိုထင်မသွားတဲ့သူများက C language ကို ဆက်လက်လေ့လာကြည့်မှ အဲဒီ C မှာ string operator လေးကွဲကွဲပဲတဲ့ strcpy( ) နဲ့ strcmp( ) function ကိုရ (2) မှာအိပ်မယ့် ဆိုက်သွားတဲ့ အားပေးကမ်းခြေနဲ့ကြည့်တယ်။ သို့သော်လည်း C ကဲ့သို့ C++ ကို ဆက်လက်လေ့လာကြည့်တဲ့အခါမှာ ဒီဘာသာစကားမှာအားပေးတဲ့ လက်ကွပ်ကွားကားကိုတွေ့ရတယ်။ C++ မှာပါပဲတဲ့ string class ကာဆီပဲ string object လေးနဲ့ပတ်သက်ပြီး ကြည့်ရုံဆိုလျှင်အခန်း ၅၅ မှာ၊ ဥပမာ string object လေးကို construct လုပ်မယ်၊ assign လုပ်မယ်၊ concatenate လုပ်မယ်၊ compare လုပ်မယ်၊ search လုပ်မယ်ဆိုရင် <string> header ကို program မှာ include လုပ်သတင်းနဲ့ ကာရန်လုပ်သင့်တယ်လို့မယ်။

## ၁၂.၁ The string Class

၀၁။ ပုံ (၁၂.၁) မှာပါကြည့်သော Ex1201.cpp program မှာဆိုရင် string ကို construct လုပ်မယ်၊ ကာရန်နဲ့လေ့လာတဲ့ program ကာရန်ပဲလဲလဲ။ လေးခုမှာ string s1; ကိုလေးခုတဲ့ statement ကဲ့သို့ empty string လေးခုကို construct လုပ်မယ့်အခါ၊ နောက်တစ်ခုမှာ string s2("This is a string");

1. `string s2` is a string object. `string s3(ch)` constructs a string object `s3` from a character array `ch`. `string s3(ch)` constructs a string object `s3` from a character array `ch`.

```

Ex1201.cpp
// Listing 12.1 Constructing strings
#include <string>
#include <string.h>

int main()
{
    string s1;

    string s2("This is a string");
    cout << s2 << endl;

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

    return 0;
}
    
```

Figure 12.1

Ex1201.cpp program of run screenshot of (Fig. 12.1) program screenshot is given as



Figure 12.2

## Assigning Strings

• A string object may be constructed explicitly, or may be assigned from an existing string (e.g. `string s2("This is a string");`), or constructed implicitly and assigned (e.g. `string s1 = s2;` may assign `s1` from `s2`). Ex1202.cpp program is given below.

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

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

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

    return 0;
}
```

• Ex1202.cpp program is run and the output is given below.



```
Quincy 59
s2 = This is a string
s1 = This is a string
s3 = A different string
Any key to return to Quincy...
```

Figure 12.2

# Concatenating Strings

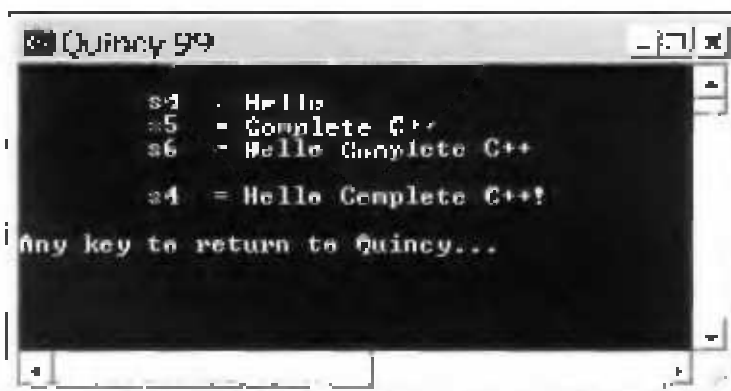
• `string` object မှာ `+` ကို concatenation operator အဖြစ် `+` သုံးပြီး အသုံးပြုနိုင်ပြီး စာပိုဒ်များကို ပေါင်းစပ်ပေးနိုင်ပါသည်။ Ex1203.cpp program ကိုမေးမြန်းကြည့်ရအောင်။

```
// Listing 12.3: Concatenating strings
#include <iostream>
#include <string>

int main( )
{
    string s4("Hello ");
    string s5("Complete C++");
    string s6 = s4 + s5;

    cout << "in\ts4 = " << s4 << "in\ts5 = " << s5
         << "\n\ts6 = " << s6 << endl;
    s4 += s5;
    s4 += '!';
    cout << "in\ts4 = " << s4 << endl;
    return 0;
}
```

• Ex1203.cpp program ကို run လိုက်မယ်ဆိုရင် နံ (၁၂-၄) မှာပြထားတဲ့ပုံတိုင်း ပြရပါမယ်။



နံ (၁၂-၄)

## Subscripting Strings

• string object contains single character objects. You can access character using subscript operator `[]`. `at()` member function also can be used. Ex1204.cpp program demonstrates it.

Ex1204.cpp: 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 << "ch1 = " << ch1 << "ch2 = " << ch2
        << "\ns4 = " << s4 << endl;
    s4.at(0) = 'Z';
    cout << "\ns4 = " << s4 << endl;
    return 0;
}
```

Ex1204.cpp program is run. It prints the following output:



```
Quincy 99
ch1 = e
ch2 = l
s4 = Hello,Complete C++
s4 = Hello,Complete C++
Any key to return to Quincy...
```

# Substrings

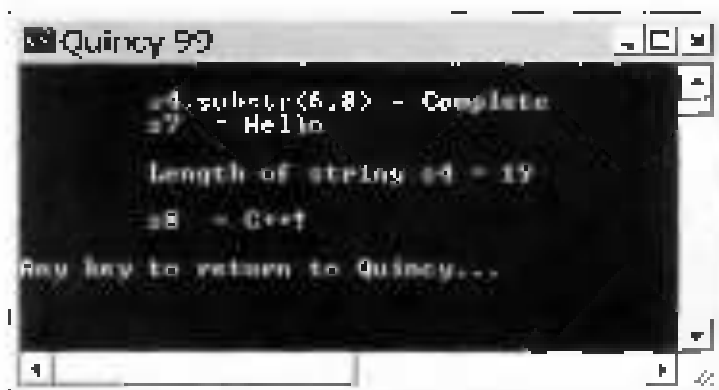
၁၂ string object တစ်ခုကနေ substring တစ်ခုကို နှိုင်းစစ်ယူရန်ရင် substr( ) function ကိုအသုံးပြုရမည်။ Ex1205.cpp program ကိုအောက်ဖော်ပြချက်အတိုင်း-

```
// Listing 12.5: Substrings
#include <iostream>
#include <string>

int main( )
{
    string s4("Hello,Complete C++!");

    cout << "\n"s4.substr(6,8) = " << s4.substr(6,8);
    string s7(s4.substr(0,5));
    cout << "\n"s7 = " << s7 << endl;
    cout << "\n"Length of string s4 = "
        << s4.length( ) << endl;
    string s8(s4.substr(s4.length( )-4,4));
    cout << "\n"s8 = " << s8 << endl;
    return 0;
}
```

၂ Ex1205.cpp program ကို run ကိုယ်စမ်းဆိုရင် မှ (၁၂-၆) ဖြစ်ပေါ်လာမည်။ ပြသချက်။



မှ (၁၂-၆)



## Searching Strings

The standard C++ library `string` class has `find()`, `rfind()` and `find_all()` overloaded functions. The `find()` function returns the index of the first occurrence of a matching substring, single character or character array in a string. It uses a naive forward search algorithm. A `-1` value is returned if the search argument is not found. The `rfind()` function returns the index of the last occurrence of a search argument. Example program is given below.

// 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 << "Index = " << n << endl;
    n = s4.find('!');
    cout << "Index = " << n << endl;
    n = s4.find(" ");
    cout << "Index = " << n << endl;
    n = s4.find("xyz");
    cout << "Index = " << n << endl;

    return 0;
}
```

Ex1206.cpp program ကို run လုပ်ရမည်ဆိုပါက ပုံ (၁၂-၅) ကဲ့သို့ အောက်ဖော်ပြပါပုံ ပေါ်လာမည်။



ပုံ (၁၂-၅)

## Comparing Strings

Ex1207.cpp program မှာ string object မှာ နှိုင်းယှဉ်စဉ်းကွင်း program တစ်ခု မျက်နှာပြင်ပေါ်တွင် ကျောင်းပြင်ပေါ်။

```
// Listing 12.7: Comparing strings
```

```
#include <iostream>
```

```
#include <string>
```

```
int main()
```

```
{
```

```
    string s1("C++ ");
```

```
    string s2("Programming ");
```

```
    if ("Bye" < s1)
```

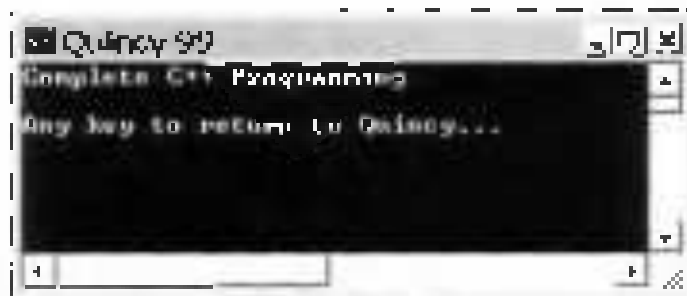
```
        if (s2 == "Programming ")
```

```

        if (s2 > s1)
        {
            string s( "Complete " );
            string s4 = s3 + s1 + s2;
            cout << s4 << endl;
        }
    }
    return 0;
}

```

Ex1207.cpp program ၏ run နံ့ရလိမ့်မည် ၏ (၂၂၃) ၏ ပုံရိပ်ကူးပေးထား ပြန်ရမည်။



ပုံ (၂၂၃)

## ၁၂.၂ The string Member Functions

၁။ standard C++ string class ၏ member function (၁) နှိုင်းယှဉ်ခြင်း (၂) clear ( ) (၃) empty ( ) (၄) length ( ) (၅) data ( ) function တို့ဖြစ်သည်။ clear ( ) function သည် string object မှာရှိ zero length ကိုပြန်လှူပေးပေး clear ပေးရန်ဖြစ်သည်။ empty ( ) function သည် string object မှာရှိ zero length ဖြစ်ခြင်းကို check ပေးပေးပေး empty ဖြစ်နေ true false data ၏ return ပေးသည်။ empty မှာသည် false နှင့် length ( ) function သည် string object မှာရှိ character ကျားများကို ချုပ်နှောင်ပေးသည်။ string data buffer ကိုရှာဖွေရာမှ ပထမဆုံး non-zero character ကိုရှာဖွေပြီး return ပေးသည်။ data ( ) member function ၏ အသုံးပြုနည်းမှာ function မှာ အသုံးပြုနည်းကို Ex1208.cpp program မှာ ကြည့်ရှုနိုင်သည်။

```

// Listing 12.0: 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\n";
    }
}

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

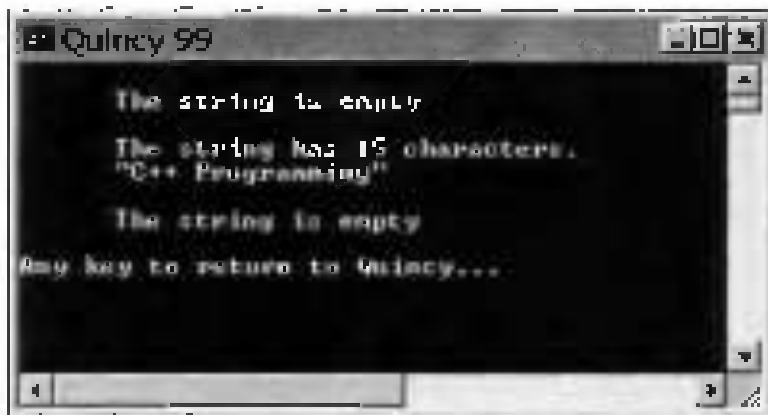
    str = "C++ Programming";
    test(str);

    str.clear( );
    test(str);

    return 0;
}

```

Ex1200.cpp program ını run edebilirsiniz. (2.5.2) bölümde açıklanmıştır.



ပုံ (၁၂.၉)

## ၁၂.၃ Formatted Output

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

» C++ program တစ်ခုကို run လုပ်ခိုင်း output ကို fixed column width တစ်ခုခုကို သတ်မှတ်ပေးရန် `width( )` member function ကိုအသုံးပြုရပါသည်။ Ex: 1209.cpp program မှ array `x[ ]` value တွေကို column width (10) ရှိတဲ့ column များ fixed format တွေမှာ scientific format နဲ့ right-align တပ်ပြီး display လုပ်ပြနိုင်အောင်တောင်းဆိုပါ။

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

```

int main( )
{
    static double x[ ] =
        { 0.00000017, 1.73, 345.678, 56789012.34 };

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

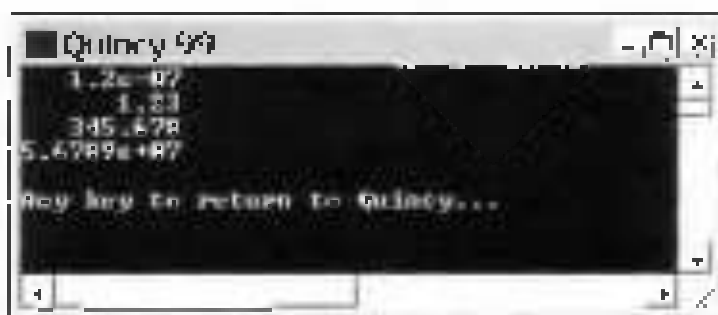
```

    {
        cout.width(10),
        cout << d[i] << endl,
    }

    return 0;
}

```

၂။ Ex1210.cpp program သို့ run လိုက်ပါက အောက်ဖော်ပြပါအတိုင်း argument value ၁၀ ဝေမျှပေးသော နံပါတ် သင်္ချာပုံစံဖြင့် scientific format နှင့် display လိုက်ပြီး argument value တို့၏ column width ဝေမျှပေးပေးပါ။ fixed format နှင့် display လိုက်ပါက column width သတ်မှတ်ပေးရမည်။ argument value ကို round (6 decimal places) လိုက်ပေးပါ။



ပုံ (၁၂-၁)

## The setw( ) Manipulator

၂။ data display လိုက်ပြီး table form ဝေမျှပေးခြင်းအား setw( ) manipulator ကိုအသုံးပြု၍ column width ကိုကိတ်ချခြင်း adjust ပေးပေးပါ။ ဤနေရာတွင် setw( ) function ကိုအသုံးပြုရန်တွင် <iomanip> header ကို program မှ include လိုက်ပေးရမည်။ cout.setf(ios::fixed) ကိုအသုံးပြု၍ argument value ၁၀ သိန်းထက်ပို၍ပါး ပေးပေးရမည် ဖြစ်ပါသည်။ Ex1210.cpp program သို့ အသုံးပြုရပါ။

```
// Listing 17.10: The setw( ) manipulator
#include <iostream>
#include <iomanip>

int main( )
{
    static double x[ ] =
        { 0.0000012, 1.23, 345.678, 56789012.34 };

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

    cout << fixed;
    cout << endl;
    for (int i = 0; i < 4; i++)
        cout << setw(10) << ch[i]
            << setw(20) << x[i] << endl;
    return 0;
}
```

Ex1710.cpp program ၏ run ရုပ်ပုံသည် ( ၁၇.၁၀ ) မှ [ကွေးကွေး] မှ ဖြစ်ပေါ်သည်။



ရုပ်ပုံ ၁၇.၁၀

၁၇.၁၀ cout << fixed) နှင့် cout << scientific) ကို အသုံးပြု၍ ဖိတ်ခေါ်မှု ( ၁၇.၁၀ ) မှ မှန်ကန်စွာ ဖြစ်ပေါ်သည်။

```

Quincy 99
ZORR-I      1.200000e+07
BIRMAX     1.210000e+08
AVELUPIN   3.456789e+02
POZAN      5.678901e+07

Any key to return to Quincy...

```

Figure 12.10

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

• In program `g++ fill( )` function ကို အသုံးပြုဆောင်ရွက်ရန် `output` နှင့် `display` နှစ်ခုလုံး `value` ကိုရန် ရေပန်းစက် `g++ fill( )` function argument မှာ အောက်ဖော်ပြပါအတိုင်း ဖြစ်စေရမည်။ Ex1211.cpp program ကို အသုံးပြုပါ။

```

// 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) Ex1711.cpp program ကို run လုပ်ဆောင်ရာမှ (၂၂, ၁၃) မှာ `cout` ကိုယ်စားပြုသည့် ခြေရာကွက်



၆ (၂၂, ၁၃)

## Output Justification

၁) `cout` မှာ `<<left>` မှာ ခြေရာကွက် program output မှာ ပုံစံပြောင်းလဲမှု left-justified လုပ်ဆောင်ရန် `setiosflags (ios::left)` manipulator ကို အသုံးပြုပေးနိုင်ပါသည်။ Ex1712.cpp program ကို အောက်ဖြေတွင်

```
// Using 12.12: The output justification
#include <iostream>
#include <iomanip>

int main()
{
    static double x[] =
        { 0.0000012, 1.23, 345.678, 55789012.34 };

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

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



```

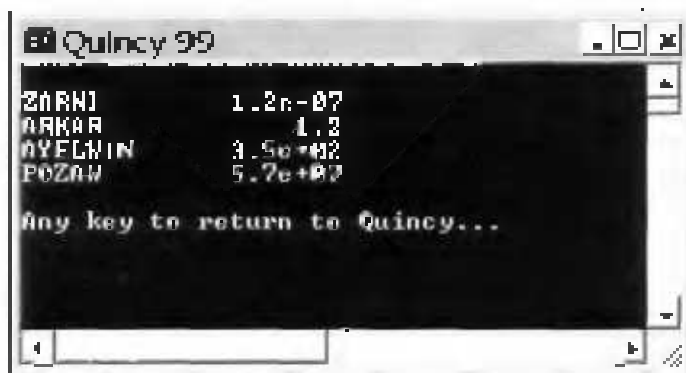
static char* ch[] =
    {"ZARNI", "ARKAR", "AYELWIN", "POZAW"};

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;
}

```

Ex1213.cpp program ကို run လုပ်သည့်အခါမှာ မှ (၁၂, ၁၅) မှုန်ပြောင်းပေးရမည်။ ပြင်ရမည်။



မှ (၁၂, ၁၅)

## Aligning Decimal Points

မှ (၁၂, ၁၅) မှုန်ပြောင်းတဲ့ program output မှာ ခေတ်ယခင်ကကဲ့သို့ floating-point value မှုန် decimal-aligned လုပ်ပေးရန် Ex1214.cpp program မှုန် ပြောင်းပေးရမည်။

```

// Listing 12.14: Scientific and fixed notation
#include <iostream>
#include <omanip>

int main()
{
    static double x[ ] =
        { 0.00000012, 1.23, 345 678, 56789012.34 };
    static char* ch[ ] =
        {"ZARNI", "ARKAR", "AYELWIN", "POZAN"};

    cout << endl;
    for (int i = 0, i < 4; i++)
    {
        cout << setw(10) << ch[i]
            << "\t" << setiosflags(ios::left) << setiosflags(ios::fixed)
            << setiosflags(ios::right) << setw(12)
            << setprecision(1) << x[i] << endl;
    }
    return 0;
}

```

Ex1714.cpp program → run → (Screenshot) → (Screenshot) → (Screenshot)

```

Quincy 93
ZARNI          0.0
ARKAR          1.2
AYELWIN       345.7
POZAN         56789012.3
any key to return to Quincy...

```

→ (Screenshot)





Figure 12.16

## Output Member Functions

- `ostream` class defines single character display `put()` member function & output object `cout`'s display `<<` overloaded `<<` insertion operator `<< (k, v)`
- `Ex1216.cpp` program displays `<<` results

```

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

```

```

int main()
{
    cout.put('C');
    cout.put('-');
    cout.put('!');
    cout << endl;
}

```

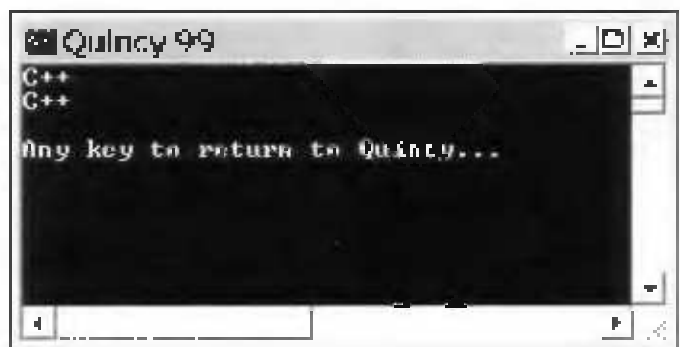
```

    cout << 'C',
    cout << '\n';
    cout << '+',
    cout << endl;

    return 0;
}

```

Ex1216.cpp program ကို run လိုက်သောအခါ (၁)၊ (၂)၊ (၃) မှာပါအတိုင်းပုံစံဖြင့် ပြသရမည်။



ပုံ (၁၂.၁၈)

write( ) member function သုံးသွား output data ကို binary format အနေနဲ့ stream သဲ ထည့်ပေးရမည်။ Ex1217.cpp program ကို run လိုက်ရင် message ကို display အောင်ယောင်မြန်စွာ alarm ပြုတ်ပေးတဲ့အတိုင်းပုံစံဖြင့် next line ကို အလိုလိုထည့်သွင်းပေးမည်။

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

```

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

```

```

    } msg = ("Complete C++ Programming", sizeof msg);

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

```

၂. Ex1217.cpp program ကို run လုပ်ရန်အတွက် နောက်ဖော်ပြပါအတိုင်း ပြုလုပ်ပါ။

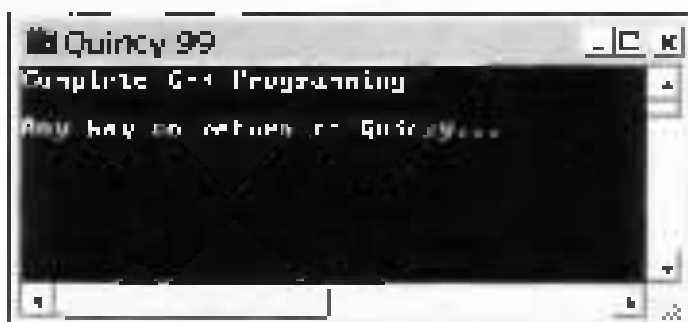


Figure 12.17

## ၁၂-၄ Input Member Functions

- ၁။ C++ program များတွင် extraction operator ကို ဝါးစွဲဖြင့် data input လုပ်ဆောင်နိုင်ရန်အတွက် အောက်ဖော်ပြပါ data input နှစ်ခုပါဝင်သော data ကိုယူရင်း အရာရှိကို အသုံးပြုနိုင်ရန်အတွက် ဖော်ပြပါ get() member function ကို အသုံးပြုရပါမည်။ Ex1218.cpp program ကို အောက်ဖော်ပြပါအတိုင်း ရေးသားပြီး အောက်ဖော်ပြပါအတိုင်း အောင်မြင်ပါ။

```

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

```

```

int main()
{

```





ပုံ ၁၂-၁၉ ကို C++ program ကို run လုပ်ရန်: input data ကို ENTER နှိပ်ခံပြီး output display ပေါ်ပြောင်ရန် Ex1219.cpp program မှာ မှတ်တမ်းတင်ရန်အားပြုပါ။

```
// 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;
}
```

ပုံ ၁၂-၂၀ ကို input data မှ password ပေါ်ပြောင်ရန်အားပြုပြီး output display ပေါ်ပြောင်ရန် Ex1220.cpp program မှာ မှတ်တမ်းတင်ရန်အားပြုပါ။

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

int main( )
{
    char line[40];

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

    return 0;
}
```

၅၉ Ex1220.cpp program ကို run ထိုက်ခင်ဆိုရင် (၁၃၂) ကို မှည့်ထားတာပဲလို့ ပြင်ရပါမယ်။



ပုံ (၁၃၂) ကို

## More on the getch( ) Function

```
// Listing 12-21 More on the getch( ) function
#include <iostream>
int main( )
{
    char ch;

    while ( (ch = cin.getch( )) != EOF)
        cout << "ch = " << ch << endl,
        cout << endl << "Bye!" << endl;

    return 0;
}
```

Ex1221.cpp program ကို run ထိုက်ခင်ဆိုရင် (၁၃၂) ကို မှည့်ထားတာပဲလို့ ပြင်ရပါမယ်။ program ထဲမှာပေးထားတဲ့ cin.getch( ) ကိုသုံးရင် string object မဟုတ် character မှာပေးရင် display ထိုက်ပြောရင်ပဲ ENTER key ထိုးမှ မဟုတ် ch = <blank> ဟု ပြန်ပေးတာပဲ။ data

input `CTRL + Z` +ENTER key `Bye!` message `display` program stop `Any key to return to Quincy...`



Figure 12.22

## Using the `get( )` Function with Parameters

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

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

```

cout << "\n(ch1 = " << ch1 << endl
     << "\n(ch2 = " << ch2 << endl
     << "\n(ch3 = " << ch3 << endl;

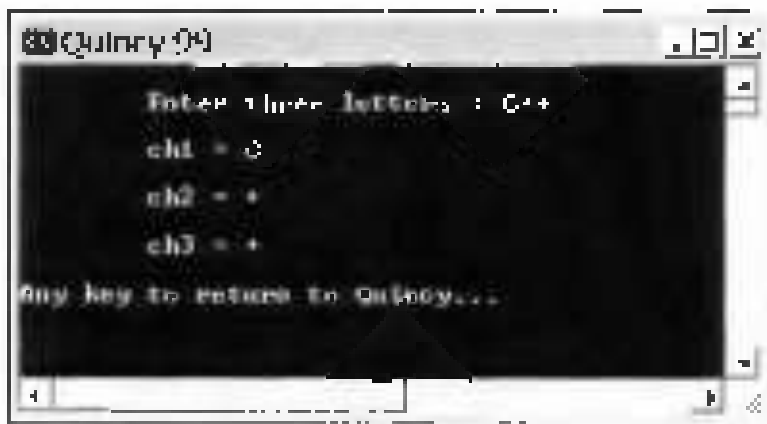
```

```

return 0
}

```

Ex0222.cpp program          



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

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

```

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

```

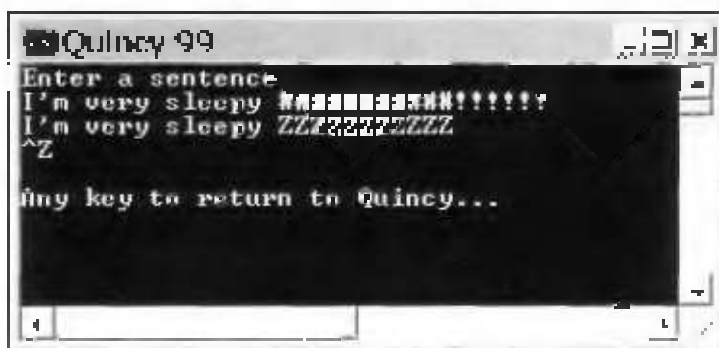
```

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

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

```

Ex1223.cpp program ကို run လုပ်သောအခိုက်မှာ ပုံ (၁၂-၂၆) မှာပြထားတဲ့အတိုင်းဖြစ်ရပါမည်။



ပုံ (၁၂-၂၆)



## 29.3 The ofstream Class

- 1. Create a program called `exit01.cpp` program to test the output of the string
- 2. Write the following program to test the output of the string

Listing 29.1 Writing strings to a file

```
#include <fstream>
```

```
int main()
```

```
{
```

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

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

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

```
    outfile << "\nReaching new heights in global education.\n";
```

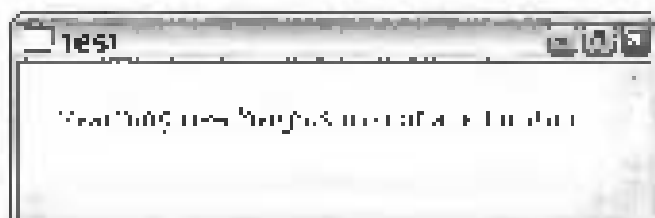
```
    return 0;
```

```
}
```

The `exit01.cpp` program creates a file named `outfile` of the `ofstream` class object instance and constructs it. The object of the `ofstream` class is `outfile`. The `ofstream` object is initialized to write to the file `test.txt`. The program's output is shown in the screen capture. The `test.txt` file is created and contains the following text:



(a)





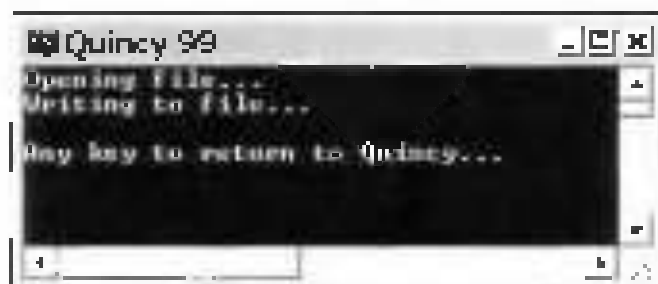
## ၁၃.၂ Appending to an Output File

၁၂။ ၁၃.၂ (၁၀) မှာ အောက်ဖော်ပြပါ test.txt output file string သိမ်းဆွဲရန်အတွက် ကုဒ်အပိုင်းကို ရေးသားရမည်။ Ex1302.cpp program ကို ရေးသားကြည့်ရန်။

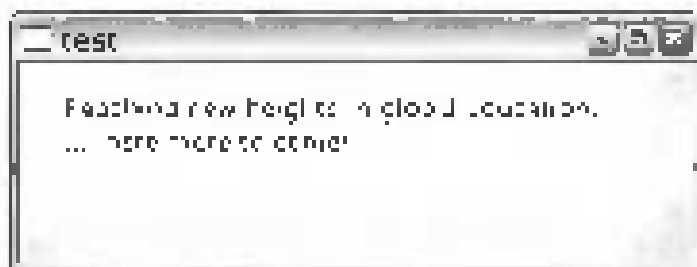
```
// Listing 13.2: Appending to an output file
#include <fstream>

int main()
{
    cout << "Opening file...\n" ;
    ofstream outfile("test.txt", ios::app);
    cout << "Writing to file...\n" ;
    outfile << "Bye... here more to come!\n";
    return 0;
}
```

၁၃။ Ex1302.cpp program ကို run ရန်အတွက် ကွန်ပိုက်လုပ် ထုတ်ပေးမှုကို screen မှာပါပုံများ test.txt ဖိုင်ကို string သိမ်းဆွဲရန် အားပေးသည့်အတွက် ပုံ (၁၃.၂) ကိုကြည့်ရန်။



ပုံ (၁၃.၂)





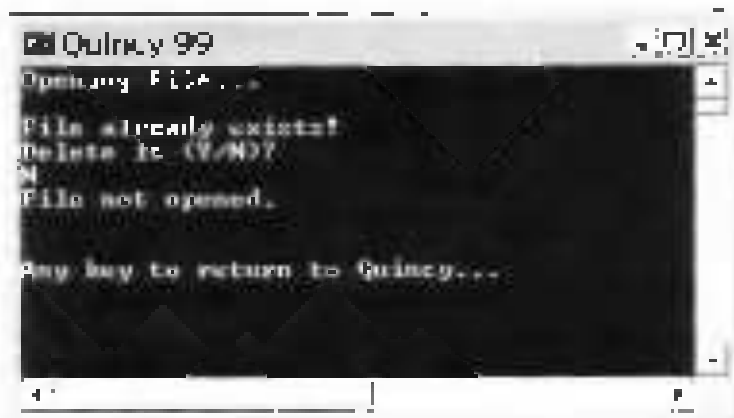


Figure 13.4

## 13.4 The ofstream( ) Function

■ In Example 13.4, the program uses the file pointer `fprintf( )` and `printf( )` function of `stdio.h` to write the input data to a file. In this example, we will use the `ofstream` class to write the input data to a file. The program will display the following output:

Figure 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<19; ++x)
{
    cout << "File pointer: " << outfile.tell(),
    cout << put(str[x]);
    cout << " " << str[x] << endl;
}

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

return 0;
}

```

Ex1304.cpp program is run (Screenshot) is shown below. File pointer is incremented by 1 and the character is displayed on the screen. This is a test program to demonstrate the use of the file pointer.



Figure 9

# Reading Strings from a File

Ex104.cpp program ကို run လုပ်ရန်အတွက် test.txt ချဲ့ဝင် data ကွင်း read ပြန်ပေးရန် အမှတ်ပေးပါ။ အမှတ်ပေးရန်အတွက် Ex105.cpp ကိုအသုံးပြုပါ။

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

```
int main()
{
    const int MAX = 81;
    char buffer[MAX];

    ifstream infile("test.txt");
    while (infile)
    {
        infile.getline(buffer, MAX);
        cout << buffer;
    }
    cout << endl;
    return 0;
}
```

Ex105.cpp program ကို run လုပ်ရန်အတွက် Ex104.cpp ကို အသုံးပြုပါ။ test.txt ချဲ့ဝင် data ကို program ကနေ ဖတ်၍ screen ပေါ် display ပြန်ရပါမည်။

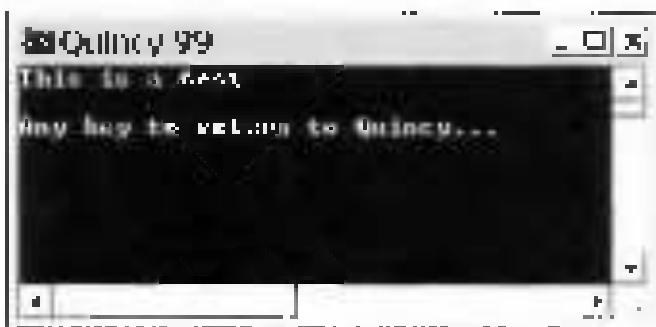


Figure 13.5

## ၁၃.၆ Reading until End-of-File

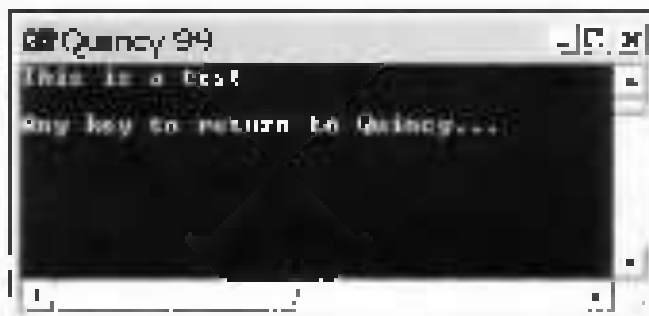
ဒီနေ့ကျွန်ုပ်တို့ `open` နည်းနဲ့ ဖိုင်ဖွင့်ပြီးမှ `isstream` ကိုသုံးပြီး `endl` ; member function ကိုသုံးပြီး `Ex1306.cpp` program ဖို့ အခြေခံရေးဆွဲထားတာပဲ။ နောက်ကျွန်ုပ်တို့ `Ex1308.cpp` program ကို `min` - ဖိုင်ဖွင့်ပြီးမှ `is` (ဒါပေမဲ့ `is` နဲ့ `isstream` ကိုသုံးပြီးမှ) ကိုသုံးပြီး ရေးဆွဲပါမယ်။

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

int main()
{
    ifstream outfile("infile.txt");

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

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

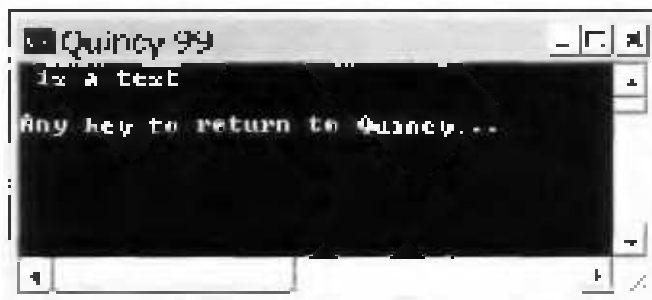


ရုပ်ပုံ ၆.၆

## ၁၃.၇ The seekg( ) Member Function

ဖိုင်တစ်ခုကို open လုပ်ပြီးနောက် information သွားတဲ့ user ကိုယ်တိုင်ကနေတစ်ခုခုပေးပြီး screen မှာ display လုပ်နိုင်ရန်နှင့် seekg( ) member function ကို အသုံးပြုရပါသည်။ Ex1307.cpp program ကို အောက်ဖော်ပြပါ ၅ program ကို run လုပ်ပါက အောက်ဖော်ပြပါ (၁၃-၇) ပုံကဲ့သို့ အသုံးပြုနိုင်ရပါသည်။

```
// 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)

# The tellg( ) Member Function

For more information on file streams, see the following white space: [File Streams](#)  
[tellg\( \) member function](#) (by David E. Estlin) [Ex1308.cpp](#) program (by David E. Estlin)

// Listing 14.8: 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;
        stringstream here = f2.tellg();
        f2.get(c2);
        if (c2 == ' ');
            cout << "Position " << here << " is a space!\n";
    }
    cout << endl;
    return 0;
}
```

▶ [Ex1308.cpp](#) program if run will output the following: [\[Screenshot\]](#)







# Opening and Closing a Stream File

• C++ program တွင်မှာ ifstream သို့မဟုတ် ofstream object တို့ကို စတင်ဖန်တီးသူ declare လုပ်ပေးရမည်။ ထိုသို့ declare လုပ်ပေးခြင်း object ကိုဖန်တီးပေးသူ object သို့မဟုတ် stream ကို စတင်ဖန်တီးသူ ထံသို့ အာရုံစိုက်ပေးရန် open() member function ကိုအသုံးပြုရမည်။ ထိုနောက်တွင် အာရုံစိုက်မှုကို ရှင်းထုတ် close() member function ကိုအသုံးပြုရမည်။ ၁၃.၁၀.၁၀.cpp program မှာ outfile သို့မဟုတ် ofstream object တစ်ခုကို declare လုပ်ပြီး test1.txt ၊ test2.txt မှာ (၂) ခုခန့် တစ်ခုချင်းချင်းဖန်တီးပေးထားလုပ်ပေးရန် ပြင်ဆင်ထားတယ် ၊ အောက်တွင်ညှိပါ။

```
// Listing 13.10: The open() and close() member functions
#include <fstream>
#include <<ctype>

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;
}
```

Ex1310.cpp program ကို run လုပ်ရာ output on screen မှာပေးထားတဲ့ ပြောပုံကို display လုပ်ပေးနိုင်အောင် code ကို (fig. 10) တွဲပြောပါ။

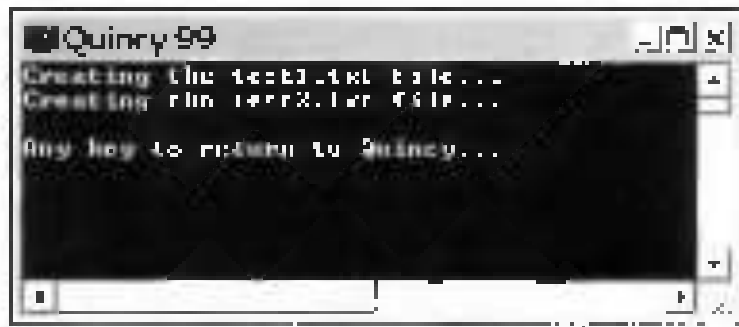


Fig. 10

test1.txt နဲ့ test2.txt နဲ့ဆက်သွယ်အောင် write လုပ်ပေးရန်အတွက် code ကို (fig. 11) တွဲပြောပါ။



Fig. 11

## ၁၃.၁၁ Objects I/O

တစ်ခဏအတွက် အောက်က object ၁ ခုကို ဖွဲ့ဆောင်ပေးရန် write လုပ်ပေး အခြေခံအားဖြင့် အောက်က code ကို ရေးဆွဲပေးရပါမည်။ Ex1311.cpp program ကို class history object ကိုဆွဲ၊ ရှိတဲ့ data ကို history.txt နဲ့ဆွဲ write လုပ်ပေးရန် screen on display ပြန်ပေးရန်အတွက် code ကို (fig. 12) တွဲပြောပါ။

```

// 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.close();
}

```

```

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

```

Ex1311.cpp program ን run ን ነገሩ data ነገሩ history.dat ትክኑሩ. write ነገሩ ትክኑሩ screen ላይ echo ትክኑሩ/ታምገራ/ክፍሉ ን ትክኑሩ/ታምገራ/ክፍሉ ን



ሪ (20. 2.0)

၃၁ Ex3012.cpp program သည် history.dat ဖိုင်ထဲမှ person data ဝဏ္ဏာ ရှိနေသည့်အခါ သံသရာရှာဖွေနိုင်ရန် information အိတ်ဂရုဏ်း screen မှ display ပြန်ပြန်ပေးနိုင်ရန်။

```
// 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 : ";
    int i;
    while (i < n)
```

```

    int pos = (i-1)*sizeof (person);

    ifile.seekg(pos);
    ifile.readf("%s", &person, sizeof person);
    person.showData( );
    cout << endl;

    return 0;
}

```

Figure 15.12: Running the Ex1512.cpp program. The user enters person number : 2. The history data is displayed on the screen as shown in the figure.

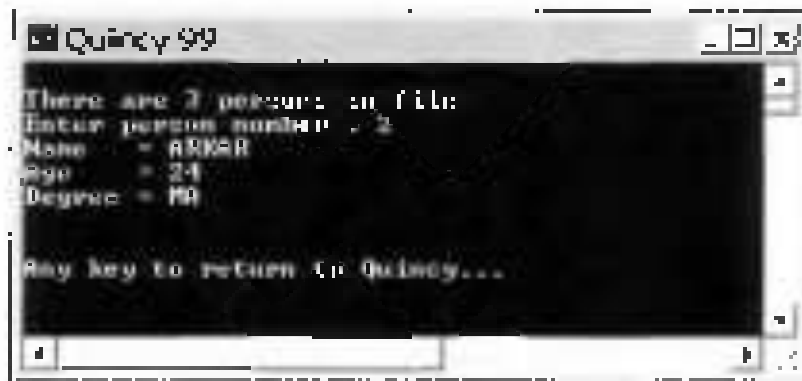


Figure 15.12







```

Quincy 99
Element #0: 15
Element #1: 15
Element #2: 15
Element #3: 15
Element #4: 15
Element #5: 15
Element #6: 15
Element #7: 15
Element #8: 15
Element #9: 15
Any key to return to Quincy...

```

Figure 14.1

## Adding Elements to a Vector

Listing 14.2 shows the `Ex1402.cpp` program, an empty vector object, `v`, is a sequence of new elements, each `push_back()` member function call. The `insert()` program inserts a new element into the vector.

```

// Listing 14.2: Adding elements to a vector
#include <iostream>
#include <vector>

int main()
{
    // create an empty vector object
    vector<char> v(10,0);

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

```

```

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

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

// display the content of the vector object
for (iter = charVec.begin( ); iter != charVec.end( ); iter++)
    cout << "Element #" << kount1+ << ": " << *iter << endl;

return 0;
}

```

၂။ Ex1402.cpp program ကိုထုတ်ကုန်ခြင်းပုံစံအရ ဆိုလိုရာ charVec သို့မဟုတ် <char> type vector object ကိုဆွဲ၍ create လုပ်ပြီး element များကို 'A' through 'J' အတိုင်း character များ assign လုပ်ပါ။ push\_back( ) member function ကိုအသုံးပြု၍ အတိုင်းအတာအရ data ကို first element အထိထည့်သွင်းပါ။ display လုပ်ရန်အတွက် အောက်ဖော်ပြပါအတိုင်း ဆွဲယူရာမှဆိုင်ရာပုံစံ program ကို run ပြီးနောက်ဆိုင်ရာ ပုံ (ပုံ ၂) မှာပြသထားသည့်အတိုင်း vector object ရဲ့ content များကိုလည်း အောက်ဖော်ပြပါအတိုင်း ပြသနိုင်မည်။



ပုံ (၁၄) ၂။

## Inserting Elements anywhere in a Vector

is the `insert()` member function on a `vector` object. It takes a sequence of element `insert` arguments and inserts them into the `vector`. Ex1403.cpp program demonstrates this.

// 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('5' + j);

    // Display the starting vector
    cout << "\nOriginal vector : ";
    vector<char>::iterator iter;
    for (iter = charVec.begin(); iter != charVec.end(); iter++)
        cout << *iter;
    cout << endl << endl;

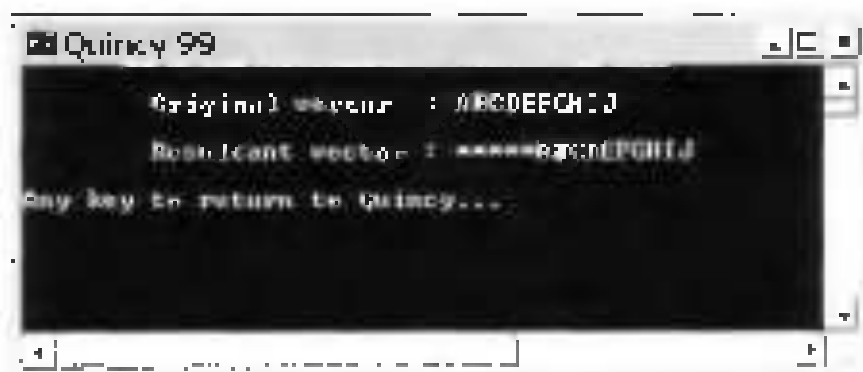
    // Insert five Xs into the vector starting from the front
    vector<char>::iterator start = charVec.begin();
    charVec.insert(start, 5, 'X');

    // Display the result
    cout << "\nResultant vector : ";
    for (iter = charVec.begin(); iter != charVec.end(); iter++)
        cout << *iter;
    cout << endl;
    return 0;
}
```

၂) Ex1403.cpp program ၎် insert( ) member function မှန်ကန်စွာ အသုံးပြုခြင်းကို

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

original vector မှာ ရှိတဲ့ မှန်ကန်စွာ start = charVec.begin( ); အသုံးပြုခြင်းနှင့် asterisk '\*' (5) ကို start မှနေတိုင်း insert ပြုစုခြင်းကို program မှတစ်ဆင့် Ex1403.cpp program ကို run ပြုစုခြင်းနှင့် ပုံ (၁၄.၃) မှာ ပြသထားသည်။ Sequence မှာ ရှိတဲ့ မှန်ကန်စွာ asterisk '\*' မှာ မှန်ကန်စွာ အသုံးပြုခြင်းကို ပြသထားသည်။



ပုံ (၁၄.၃)

## Removing Elements from a Vector

၁) ဘယ်လိုအသုံးပြုသလဲ; Ex1404.cpp program မှာတော့ ABCDEFGHIJ ကို; sequence မှာ မှန်ကန်စွာ element မှန်ကန်စွာ မြှုပ်နှံမှုမှတစ်ဆင့် program မှတစ်ဆင့် pop\_back( ) function မှန်ကန်စွာ အသုံးပြုခြင်းကို

```
// 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('A' + j);

    int size = charVec.size();
    cout << endl;
    for (int k=0; k<size; ++k)
    {
        charVec.pop_back();
        vector<char>::iterator iter,
        cout << "0";
        for (iter = charVec.begin(); iter != charVec.end(); iter++)
            cout << *iter;
        cout << endl;
    }
    return 0;
}

```

- (24. 2) Ex1404.cpp program ın run ıçin, lıko ın



(24. 2)

## Removing Elements Anywhere within a Vector

၁. `pop_back()` member function ဝင်အားပြင်ရာထက် `erase()` member function ပြန်ပါ  
တည်းကောင်းကောင်းပြောင်းပါ။ Ex1405.cpp program မှ ABCDEFGHIJ ဆိုတဲ့ sequence မှ ၃ ရှေ့ဆုံးကနေ  
element အိမ်မှန်ကို ချစ်ကောင်းတဲ့ program ပါ။ `erase()` function ကိုအသုံးပြုတဲ့ ကုဒ်ကောက်ကြည့်ပါ။

// 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. ဒီ (code) မှ Ex1405.cpp program ကို run ပြောနိုင်တယ်။





Figure 14.6

## Comparing Vectors

Figure 14.6 shows the output of the program `Ex1406.cpp` from Listing 14.6. The program compares two `vector<char>` objects (2) and prints the result of the comparison. The program also prints the contents of the two vectors.

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

// Compare the vectors.
if (charVec1 == charVec2)
    cout << "\n(vector1 == vector)",
else if (charVec1 < charVec2)
    cout << "\n(vector1 < vector)",
else
    cout << "\n(vector1 > vector)",
cout << endl;
return 0;
}

```

• (ص. 5) : Ex:406.cpp program نى run قىلىش.

```

Quincy 99
Vector 1: ABCDEFGHIJ
Vector 2: HGFEDGHIJK
vector1 < vector2
Any key to return to Quincy...

```

• (ص. 6)

## Sorting a Vector of Integers

• `sort()` is used in the `Ex1407.cpp` program. The `rand()` function generates a pseudo-random integer sequence. `rand()` is a function of the `stdlib.h` header. `sort()` is a function of the `algorithm` header.

□ Listing 14.7: Sorting a vector of integers

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

int main()
{
    int n,
        i,
        cout << "n: How many integers? ";
    cin >> n;

    vector<int> mVec;

    for (i=0; i < n; i++)
        mVec.insert(mVec.end(), rand() );

    cout << "n: Unsorted ---\n";
    vector<int>::iterator iter;
    int k=0;

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

```

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

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

cout << endl;
return 0;
}

```

For C++ program to run: How many integers? prompt will ask for 12 random generate numbers and display them sorted.



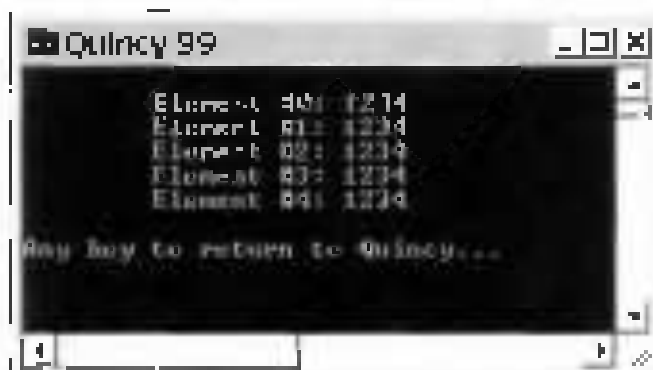
Figure 1.10

## 99.2 The deque Class Template

• deque class template is a vector-like sequence of elements of element type T. It is similar to vector and array element access, but it is not like vector. It can dynamically resize. Example: 408.cpp program - create deque object, create iterator, print screen, display program output. (99.2) program is run successfully.

```
// Listing 14.9 Creating a simple deque
#include <iostream>
#include <deque>

int main()
{
    deque<int> intDeq(5, 1234);
    int count = 0;
    deque<int>::iterator iter;
    cout << endl;
    for (iter = intDeq.begin(); iter != intDeq.end(); iter++)
        cout << "(Element " << count++ << "): "
            << *iter << endl;
    return 0;
}
```



```
Quincy 99
Element 00: 1234
Element 01: 1234
Element 02: 1234
Element 03: 1234
Element 04: 1234
Any key to return to Quincy...
```

(99.2)

## Adding Elements to a Deque

Listing 14.9 shows the Ex1409.cpp program. An empty deque object is created, and a sequence of new elements is added by the `push_front()` member function. The program prints the data of last element, and the program is run.

( Listing 14.9: Adding elements to a deque

```
#include <iostream>
#include <deque>

int main()
{
    deque<char> charDeq;
    int count = 0;
    for (int i=0; i<5; ++i)
        charDeq.push_front('E' + i);

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

( Listing 14.9)

## Inserting Elements anywhere in a Deque

• `deque::insert()` member function: Inserts a deque object into a sequence of element objects. `insert` copies the deque object into the sequence. Ex1410.cpp program illustrates the use of `insert()`.

Ex1410.cpp program: Inserting elements anywhere within a deque

```
#include <iostream>
#include <deque>

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

    cout << "\n(Original 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, 'f');

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

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

Ex1410.cpp program: Inserting elements anywhere within a deque


## Adding Elements to a Deque

Now we'll modify the program in Ex1409.cpp program to use an empty deque object and a sequence of new elements using the `push_front()` member function to add the data of last element using the `insert` function of program. The following is the code for Ex1409.cpp program if you run it you will see:

```
/* Listing 14.9: Adding elements to a deque
#include <string.h>
#include <deque>

int main()
{
    deque<char> charDeq;
    int count= 0;
    for (int i=0, i<5; ++i)
        charDeq.push_front('5' + i);

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



```
Quincy 99
Element #0: 5
Element #1: 6
Element #2: 7
Element #3: 8
Element #4: 9
Any key to return to Quincy...
```

Figure 14.9



## Inserting Elements anywhere In a Deque

• The `insert()` member function of a deque object inserts a sequence of element objects before the element specified in the program. Example: Ex1410.cpp

```
// 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(55 + x);

    cout << "Initial 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 << "Resultant deque: ",
    for (iter = charDeq.begin(); iter != charDeq.end(); iter++)
        cout << *iter;

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

Ex1410.cpp program is run. Output is shown below:



(Fig. 14)

## Removing Elements from a Deque

● In the previous example, the program uses the `UNORDERED` deque sequence and element using `push_front()` function. In this program, we create a `deque` sequence and use the `pop_back()` function to remove elements from the back of the deque.

// Listing 14-11: Removing elements from a deque

```
#include <iostream>
#include <deque>

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

    int size = charDeque.size();
    cout << endl;
    for (int x=0; x<size; ++x)
    {
        charDeque.pop_back();
    }
}
```

```

        cout << '\n';
        deque<char>::iterator iter;
        for (iter = charDeq.begin(); iter != charDeq.end(); iter++)
            cout << *iter;
        cout << endl;
    }
    return 0;
}

```

Ex1411.cpp program के पूरे कोड को नीचे दिए गए लिंक से देखें।



Ex1411.cpp

## Removing Elements Anywhere within a Deque

`push front()` व `erase()` member function को `deque` object में create किए गए element को `deque` में `erase()` Ex1412.cpp program को देखें। नीचे दिए गए लिंक से program को पूरा कोड को देखें।

```

// 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 << "X";
        deque<char>::iterator iter;
        for (iter = charDeq.begin( ); iter != charDeq.end( ); iter++)
            cout << *iter;

        cout << endl;
    }
    return 0;
}

```

↳ (top = 0)

## Comparing Deques

Listing 14.13 shows the `Ex1413.cpp` program for `char` vector object (2). The program compares two `deque` objects (1) (i.e., `q1`) of program 1) run `g++ ex1413.cpp`.

// Listing 14.13. Comparing deques

```
#include <string>
```

```
#include <deque>
```

```
int main( )
```

```
{
```

```
    deque<char> charDeque1;
```

```
    for (int i=0; i<10; i++) charDeque1.push_back('A' + i);
```

```
    deque<char> charDeque2;
```

```
    for (int i=0; i<10; i++) charDeque2.push_front('A' + i);
```

```
    cout << "\nDeque 1: ";
```

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

```
    for (iter = charDeque1.begin(); iter != charDeque1.end(); iter++)
```

```
        cout << *iter,
```

```
    cout << endl;
```

```
    cout << "\nDeque 2: ";
```

```
    for (iter = charDeque2.begin(); iter != charDeque2.end(); iter++)
```

```
        cout << *iter,
```

```
    cout << endl;
```

```
    if (charDeque1 == charDeque2)
```

```
        cout << "\nDeque1 == deque2";
```

```
    else if (charDeque1 < charDeque2)
```

```
        cout << "\nDeque1 < deque2";
```

```
    else if (charDeque1 > charDeque2)
```

```
        cout << "\nDeque1 > deque2";
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```



Figure 14.14

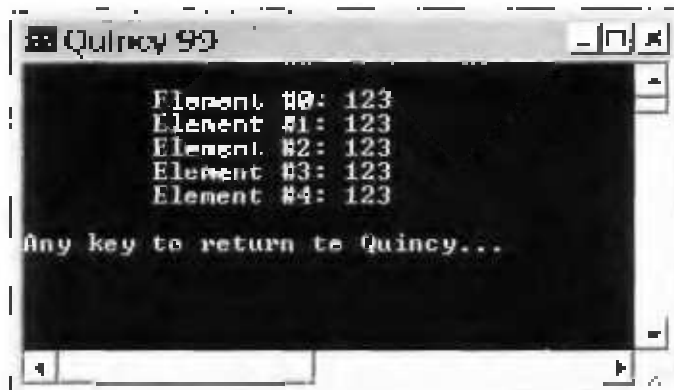
## 14.9 The list Class Template

Listing 14.14 shows an Ex1414.cpp program that uses the `rand()` function to generate a random integer list sequence. The program also uses the `sort()` function to sort the sequence. Listing 14.15 shows a program that uses the `sort()` function to sort a sequence of integers.

// Listing 14.14. Creating a single list

```
#include <iostream>
#include <list>

int main()
{
    list<int> intList(5, 123);
    int count = 0;
    for<int> iter(int);
    cout << endl;
    for (iter = intList.begin(); iter != intList.end(); iter++)
        cout << "Element #" << count << " is "
            << *iter << endl;
    return 0;
}
```



↳ (39, 29)

## Adding Elements to a List

• `Ex1415.cpp` program `l1` empty list object `charList` sequence  
 • new element `65 + i` `push_front()` member function `insert()` program  
 • `cout` `endl` `begin()` `end()` `iter++` `cout` `endl` `return 0;`

// 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<?; ++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;
```

```
}
```



Figure 14.16

## Inserting Elements anywhere in a List

✎ `std::list::insert()` member function of a `std::list` object inserts a sequence of blank elements at a specific position. Ex 14.16.cpp program shows the result.

```
// 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_back('5' + x);

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



```

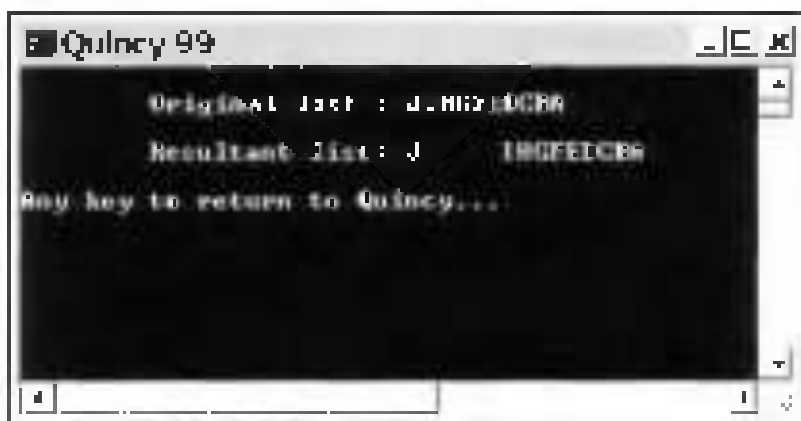
cout << endl;

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

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

```

Ex 14166) Ex14166.cpp program 5) run 5)com,look



( Ex. 16 )

## Removing Elements from a List

Ex14170) program which ABCDEFGH sequence and elements 'E' & 'B' are  
 by using pop() program ( push back() function & remove() function )

```

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

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

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

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

```

➤ `g++ Ex1417.cpp` program `&& run { gcc } gcc`

```

Quincy 99
Original list : 0123456
Resultant list : 02456
Any key to return to Quincy...

```

➤ (14-17)





```

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

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

if (charList1 == charList2)
    cout << "\nlist1 == list2";
else if (charList1 < charList2)
    cout << "\nlist1 < list2";
else if (charList1 > charList2)
    cout << "\nlist1 > list2";

cout << endl;
return 0;
}

```

Ex1419.cpp program:  $\text{list1} = \text{ABCDEFGHIJ}$ ,  $\text{list2} = \text{KLMNOPQR}$  (Figure 14.19)

```

Quincy 99
list 1: ABCDEFGHIJ
list 2: KLMNOPQR
list1 < list2
One key to return to Quincy...

```

Figure 14.19

## ၁၄.၅ The stack Container Adaptor

Ex1420.cpp program use push() နှင့် pop() function များကို တွက်ချက်ပြီး stack sequence မှာပါရှိသော အမှတ်အသား ပြုလုပ်ထားတဲ့ program ရှိသည်။

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

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

    cout << "Initial values pushed onto stack:\n";
    for (int x=1; x<=7; ++x)
    {
        intStack.push(x*100);
        cout << "x" << x*100 << endl;
    }

    cout << "Initial values popped from stack:\n";
    int size = intStack.size();
    for (int x=0; x<size; ++x)
    {
        cout << "!" << intStack.top() << endl;
        intStack.pop();
    }
    cout << endl;
    return 0;
}
```

Ex1420.cpp program ကို run လုပ်နိုင်စေရန် (၁၄.၂၀) မှာရှိသော အမှတ်အသား ပြုလုပ်ရပါမည်။

```

Quincy 99
Values pushed onto stack:
100
200
300
400
500
600

Values popped from stack:
600
500
400
300
200
100

Any key to return to Quincy...

```

☐ (☐ ☐ ☐)

## Sorting a Stack of Integers

Listing 14-21 shows a program that randomly generates a stack sequence and sorts it using the `sort` function. The `push()` & `pop()` function calls are used to generate the stack.

// Listing 14-21: Sorting an array of integers

```

#include <ostream>
#include <cstdlib>
#include <stack>

```

```

int main()
{
    int n;

```

```

cout << "\n!(How 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!nt!--- 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;
}

```

▶ (၁၄-၂၃) ရှိ Ex1421.cpp program ကို run လုပ်ကြည့်ပါ။

## ၁၄.၆ The queue Container Adaptor

▶ (၁၄-၂၄) ရှိ Ex1422.cpp program ကို push( ) function ကိုသုံးပြီး generate လုပ်သော queue sequence ကို remove လုပ်ဖို့ pop( ) function နှင့် remove လုပ်ကြည့်ပါ။



```

Quincy 99
How many integers? 12
----- Pushing -----
  41    18457    5334    26586
19169   15724   11478   29356
26962   24464    5795    28145

----- Popping -----
 28145    5795    24464    26962
29356   11478   15724   19169
26586    5334   18467     41

Any key to return to Quincy...

```

Figure 14.22

```

// Listing 14.22 Managing a queue
#include <iostream>
#include <list>
#include <queue>

int main()
{
    queue<int, list<int>> mQueue;

    cout << "Any values pushed onto queue:\n";
    for (int x=1; x<7; ++x)
    {
        mQueue.push(x*100);
        cout << "x: " << x << "x*100: " << endl;
    }

    cout << "Any values removed from queue:\n";
    int size = mQueue.size();
}

```

```

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

cout << "\n\n!-- 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!-- 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;
}

```

▶ (29- 32) Ex1421.cpp program ରେ run କିରାକରାନ୍ତୁ

## ୦୫.6 The queue Container Adaptor

▶ କାର୍ଯ୍ୟକାରୀ କିରାକରା Ex1422.cpp program ରେ push( ) function ରେ କିରାକରା କିରା generate କରାକରା queue sequence ରେ କାର୍ଯ୍ୟକାରୀ pop( ) function କି remove କିରାକରାକରାକରାକରା.

```

Quincy 90
How many integers? 12
- - Pushing - -
  41    18467    6334    26588
19169   15724   11478   29358
36762   24464    5786    28145
- - Popping - -
  28145    4786    24464    26962
29358    11478   15724    19169
26588    6334    18467    41
Any key to return to Quincy...

```

Figure 14.22

Listing 14.22: Managing a queue

```

#include <ostream>
#include <list>
#include <queue>

int main()
{
    queue<int, list<int>> intQueue;

    cout << "\n"; Values pushed into queue:\n";
    for (int x=1; x<=7; ++x)
    {
        intQueue.push(x*100);
        cout << "%d" << x*100 << endl;
    }

    cout << "\n"; Values removed from queue:\n";
    int size = intQueue.size();

```

```

for (int x=0; x<size; ++x)
{
    cout << "E" << intQueue.front() << endl;
    intQueue.pop();
}
cout << endl;
return 0;
}

```

➤ `g++ 09_11.cpp Ex1422.cpp` program `Ex1422.cpp` program `Ex1422.cpp` program `Ex1422.cpp` program



`g++ 09_11.cpp`

## 09.7 The priority\_queue Container Adaptor

➤ `g++ 09_11.cpp Ex1423.cpp` program `Ex1423.cpp` program `Ex1423.cpp` program `Ex1423.cpp` program `Ex1423.cpp` program `Ex1423.cpp` program

```

// Listing 14.25: 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 << "The values removed from priority queue is",
    int size = intPQue.size();
    for (int x=0; x<size; ++x)
    {
        cout << " " << intPQue.top() << endl;
        intPQue.pop();
    }
    cout << endl;
    return 0;
}

```

➤ (iii) To run Ex1425.cpp program → run `g++ Ex1425.cpp`

```

Bd Quincy %
The values removed from priority queue is
 400
 500
 300
 200
 100

Any key to return to Galaxy...

```

➤ (iv) To

# Chapter 15



STL associative container တွေက sequence တွေနဲ့တူပြီး container ထဲက element တွေကိုအတိုင်းပဲ ထည့်ထားတာ၊ key တွေကိုအတိုင်းပဲ element တွေကို locate တဲ့ပုံစံမှာပဲ။ associative container type တွေကို ကို နောက်ခေါ်တယ်။ (၁) set (၂) multiset (၃) map (၄) multimap (၅) bitset type တို့ဖြစ်တယ်။ အတိအကျ တွေ့ရအောင် set class template ကိုဖြင့် ခပ်နားနားပေးဖွယ် ဖြည့်ပေးတယ်။

## ၁၅.၁ The set Class Template

set class object တစ်ခုကို တည်ဆောက်ရာမှာ sorted order ဖြစ်အောင် program မှာ ထည့်သွင်း အညွှန်းပေးရတယ်။ set ကိုပဲပဲ ordered list တွေရဲ့ မင်တရီလေ။ set element များက stored data ကနေမိလ်တယ်။ data ကျားကတော့ key တွေကိုဖြင့် အညွှန်းပေးထားတာပဲ။ (Part 50) C++ ကနေရင် set object တွေကို create တဲ့အခါ content တွေကို screen မှာ display တဲ့အဖြစ်ပဲ။ program တစ်ခု ဖြစ်တယ်။ အောက်ဖွဲ့ရတယ်။

```

// 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 << "Contents of set:\n";
    set<int>::iterator iter;
    for (iter=intSet.begin(); iter!=intSet.end(); iter++)
        cout << *iter << " ";

    return 0;
}

```

Ex 15.1 code program ကို run လုပ်လျှင် set object သို့ insert လုပ်သော element တွေကို အောက်ဖော်ပြပါ အပြုအမူများကို တွေ့ရပါမည်။ (၁၅.၁) ပုံပြန်ရန်။



(၁၅.၁)

## Adding Elements to a set

Listing 15.2: Adding char elements to a set

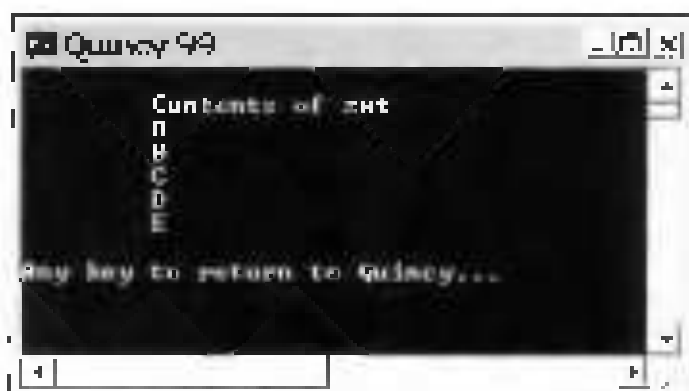
;; Listing 15.2: Adding char elements to a set

```
#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(Contents of set):\n";
    set<char>::iterator iter;
    for (iter = charSet.begin(); iter != charSet.end(); iter++)
        cout << *iter << " ";

    return 0;
}
```





## Removing Elements anywhere within a set

```
1 // Listing 15.3: Removing elements anywhere within a set
2 #include <string>
3 #include <set>
4
5 int main( )
6 {
7     std::set<char> charSet;
8
9     charSet.insert('E');
10    charSet.insert('D');
11    charSet.insert('C');
12    charSet.insert('B');
13    charSet.insert('A');
14
15    // Display the contents of the set.
16    cout << "\n\nContents of set:\n";
17    set<char>.iterator iter;
18    for (iter = charSet.begin( ); iter != charSet.end( ); iter++)
19        cout << "E" << *iter << endl;
20
21    iter = charSet.begin( );
22    charSet.erase(++iter);
23
24    cout << "\n\nContents of new set:\n";
25    for (iter = charSet.begin( ); iter != charSet.end( ); iter++)
26        cout << "E" << *iter << endl;
27    return 0;
28 }
```

// Listing 15.3: Removing elements anywhere within a set

```
#include <string>
#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\nContents of set:\n";
```

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

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

```
    cout << "E" << *iter << endl;
```

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

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

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

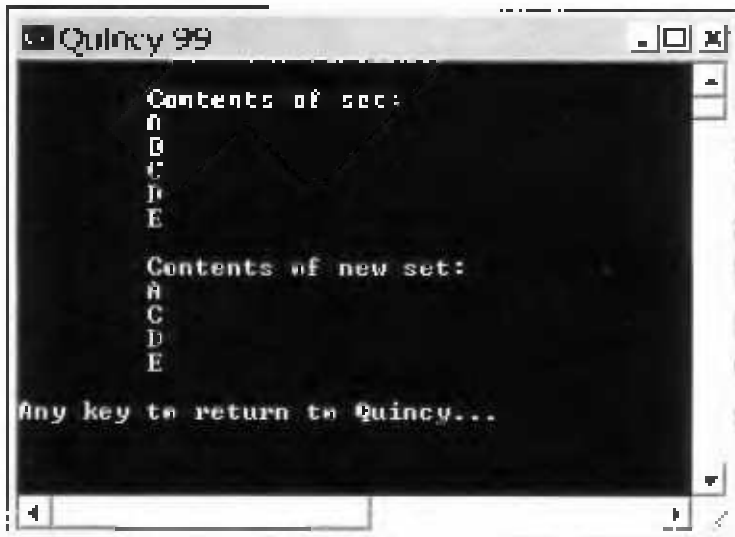
```
for (iter = charSet.begin( ); iter != charSet.end( ); iter++)
```

```
    cout << "E" << *iter << endl;
```

```
return 0;
```

```
}
```

၂၂ Ex1503.cpp program မှ run လိုက်ပါက အောက်ပါအတိုင်း sorted set element များကို display လုပ်ပြီး second element (B) ကို erase လုပ်ပြီးနောက် new element များကို display လုပ်ခြင်းကို မြေပုံဖော်ပါ။



ပုံ (၁၅-၅)

## Searching a set

၂၃ အောက်ဖော်ပြပါအတိုင်း Ex1504.cpp program ကို run လိုက်ပါက set element များထဲမှ ဖြစ်ပါက element တစ်ခုကို ရှာဖွေတဲ့ program ဖြစ်ပါကပါ။ element ရှိနေပါက Element found. လို့ပေးပြီးနောက် ပြန်မပါကပါ။ Element not found. လို့ပေးပြီးနောက်ပါ။

```

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

int main()
{
  
```

```

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

    cout << "\n(Contents of set):\n";
    set<char>::iterator iter;
    for (iter = charSet.begin(); iter != charSet.end(); iter++)
        cout << *iter << " ";

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

```

Ex1504.cpp program to run (set of ABCDE) set use 'D' (Element found) Element found D (display) (Element found: D) (set of ABCDE)

```

Quincy 99
Contents of set:
A
B
C
D
E

Element found: D

Any key to return to Quincy...

```

## Comparing sets

Listing 15-5: Comparing sets

```
// Listing 15-5: Comparing sets
```

```
#include <iostream>
```

```
#include <set>
```

```
int main( )
```

```
{
```

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

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

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

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

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

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

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

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

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

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

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

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

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

```
    charSet2.insert('T');
```

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

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

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

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

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

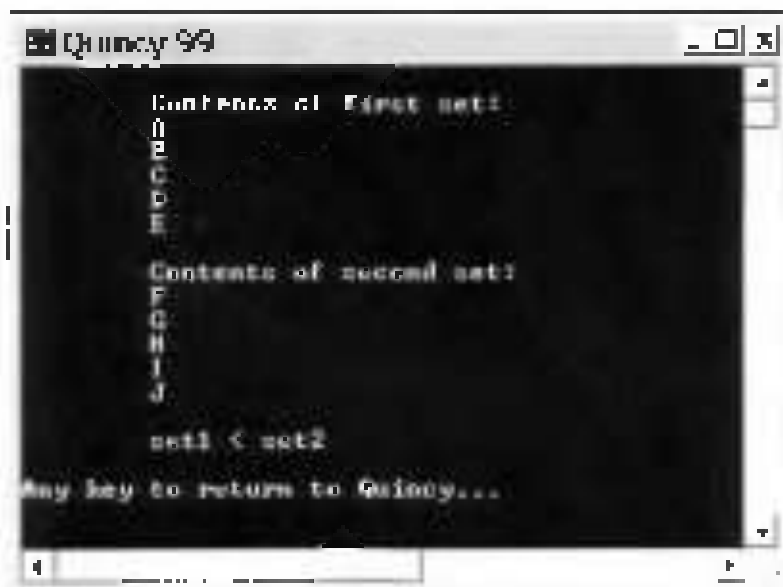
```

for (iter= charSet1.begin(); iter != charSet1.end(); iter++)
    cout << *iter << endl,
cout << endl;

// Compare the sets
if (charSet1 == charSet2)
    cout << "set1 == set2";
else if (charSet1 < charSet2)
    cout << "set1 < set2";
else
    cout << "set1 > set2";
cout << endl;
return 0;
}

```

Ex11113.cpp program ko run karne ke liye ABCDE ko set (1) aur FGHIJ ko set (2) ke liye set1 ko set2 se compare karke display karne ke liye run karke dekhiye.



↳ (11.9)

# ၁၅.၂ The multiset Class Template

• multiset object တစ်ခုခုကို အစိုးအစိုးအစိုးနဲ့ sorted under `lt` program မှာ အညွှန်းတမ်း အစိုးအစိုးအစိုးအစိုး multiset တစ်ခုကို set မှာ အစိုးအစိုးအစိုးအစိုး multiset element များကို duplicate မှာ အစိုးအစိုးအစိုးအစိုး `Ex1506.cpp` မှာ အစိုးအစိုး multiset object တစ်ခုကို create လုပ်ပြီး content များကို display လုပ်ပြီးအစိုးအစိုး program တစ်ခုဖြစ်အောင် မှာ အစိုးအစိုးအစိုးအစိုး

// 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(1);
    intMultiset.insert(8);
    intMultiset.insert(5);
    intMultiset.insert(8);

    // Display the contents of the multiset.
    cout << "\n(Contents of multiset):\n";
    multiset<int>::iterator iter;
    for (iter = intMultiset.begin(); iter != intMultiset.end(); iter++)
        cout << *iter << " ";
    return 0;
}
```

• `Ex1506.cpp` program ကို run လုပ် လုပ်အစိုးအစိုး `g++ (prog.cpp) -std=c++11` မှာ အစိုးအစိုးအစိုးအစိုး 1 3 5 5 8 8 10 မှာ multiset တစ်ခုကို create လုပ်ပြီး content များကို display လုပ်ပြီးအစိုးအစိုးအစိုးအစိုး

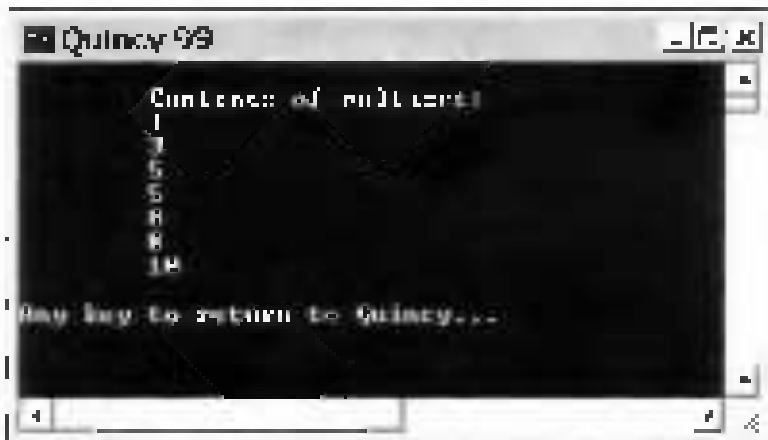


Figure 15.7

## Inserting multiset Elements

Listing 15.7 shows how to insert elements into a multiset. The program prompts the user to enter a character and then inserts that character into the multiset. The program then prompts the user to enter a key to return to the prompt.

// Listing 15.7: Adding elements to a multiset

```
#include <ostream>
#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 << "\nContents of multiset:\n";
multiset<char>.iterator iter;
for (iter= charMultiset.begin(); iter != charMultiset.end(); iter++)
    cout << *iter << " ";
return 0;
}

```

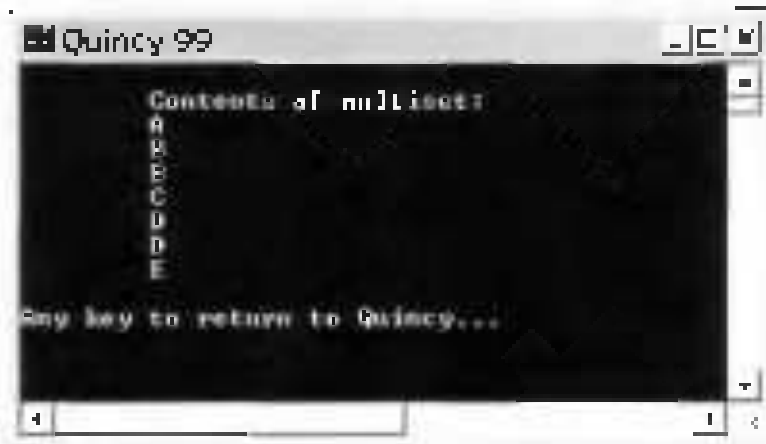


Figure 15.9

## Removing multiset Elements

Figure 15.9 shows the output of the EX1508.cpp program. Initially, multiset contains the second element 'B'. Erase creates a new set with ABCDDE. Some programs may need to remove

```

// Listing 15.9. Removing multiset elements
#include <iostream>
#include <set>

```

```
int main()

```



```

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

    // Populate the multiset with values.
    charMultiset.insert( 'C' );
    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\nContents of multiset:\n";
    multiset<char>::iterator iter;
    for ( iter = charMultiset.begin( ); iter != charMultiset.end( ), iter++; )
        cout << " " << *iter << endl;

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

    // Display the new contents of the multiset.
    cout << "\n\nContents of new set:\n";
    for ( iter = charMultiset.begin( ); iter != charMultiset.end( ), iter++; )
        cout << " " << *iter << endl;
    return 0;
}

```

Figure 15-10: Example 15-10: Erasing an element from a sorted multiset. The program first displays the contents of the multiset, which are C, D, C, B, A, B, D. Then it erases the second element (D), and displays the new contents of the multiset, which are C, C, B, A, B.

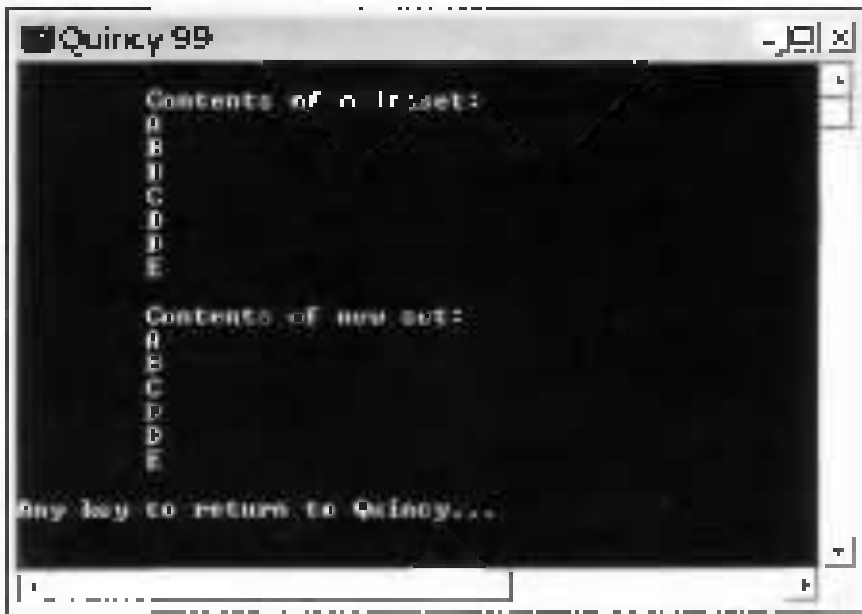


Figure 15.9

## Searching a multiset

Listing 15.9 shows the `Ex1509.cpp` program that searches a multiset element by element. The program prints the element if found. Element found: A. Next element: B. Element not found: C. Next element: D. Element found: D. Element not found: E.

// Listing 15.9. Searching a multiset

```
#include <ostream>
#include <set>

int main()
{
    multiset<char> charMultiset;
    charMultiset.insert('A');
    charMultiset.insert('D');
    charMultiset.insert('C');
}
```

```

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

cout << "\n";
cout << "Contents of multiset:\n";
multiset<char>::iterator iter;
for (iter = charMultiset.begin(); iter != charMultiset.end(); iter++)
    cout << *iter << " ";
cout << endl;

// Find the first D
iter = charMultiset.find('D');
if (iter == charMultiset.end())
    cout << "Element not found.\n";
else
{
    cout << "Element found: " << *iter++ << endl;
    cout << "Next element: " << *iter;
}
cout << endl;
return 0;
}

```

```

Quincey 99
Contents of multiset:
A
B
B
B
C
D
D
E

Element found: D
Next element: D

Any key to return to Quincey...

```

Figure 19-8

## Comparing multisets

> Example 15.10 (pp program) compares multiset objects. The program outputs the following:

// 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\nContents of first multiset:\n";
```

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

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

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

```
    cout << endl;
```

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

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

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

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

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

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

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

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

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

```

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

// Compare the sets
if (charMultiset1 == charMultiset2)
    cout << "set1 == set2";
else if (charMultiset1 < charMultiset2)
    cout << "set1 < set2";
else
    cout << "set1 > set2";
cout << endl;
return 0;
}

```

Ex1510.cpp program ನ್ನು ಸುಸ್ಥಿರವಾಗಿ ABBCDDE & FGGHIJJ ಒಂದು multiset 1) & ಎರಡನೇ set1 & set2 ಮೇಲಿನ ಎರಡನೇ ಸೆಟ್ ಮೇಲಿನ ಎರಡನೇ display ಮಾಡಿಕೊಡುವ (Fig. 10) ಕೆಲಸವನ್ನು.

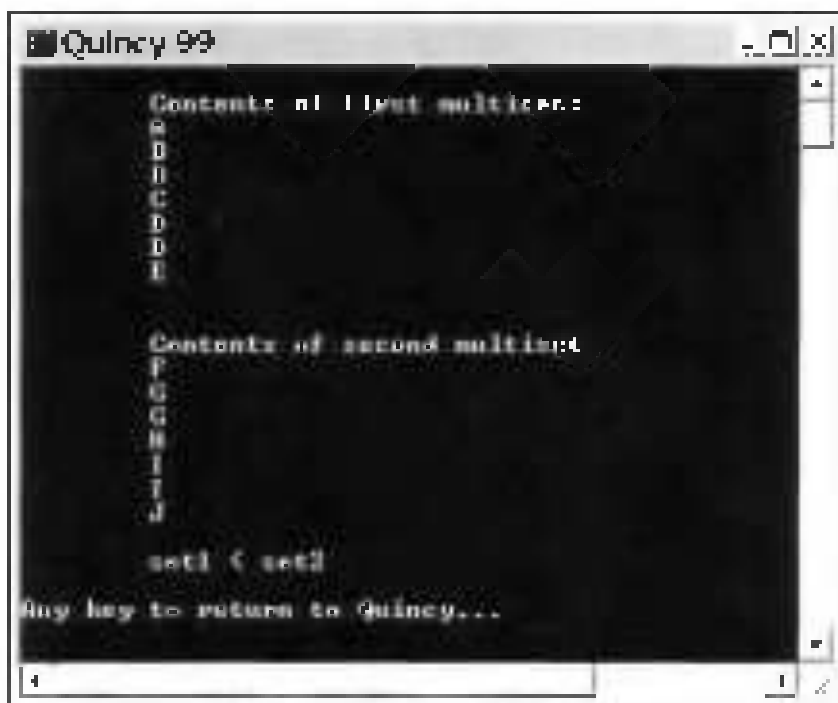


Fig. 10

## ၁၅.၃ The map Class Template

၁. map class object ကိုအသုံးပြု၍ sorted order ပြန်ကပ် program မှတည်ဆောက်ပြီးတော့၊ map element ကိုလည်း stored data ကိုအသုံးပြု၍ data ကိုရှာဖွေရာတွင် search key ကိုအသုံးပြု၍ အသုံးပြုနိုင်ပါသည်။ EX1511.cpp ကိုအသုံးပြု၍ simple map object ကိုဆောက်တည်ပြီး content ကို display လုပ်ပြုနိုင်ရန် program ကိုအသုံးပြုနိုင်ပါသည်။ (၁၅.၁၁) ကိုအသုံးပြုခြင်း

```
// 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(Contents of map:\n";
    map<int, char>::iterator iter;
    for (iter= charMap.begin( ); iter != charMap.end( ); iter++)
    {
        cout << "A" << (*iter).first << " --> ";
        cout << (*iter).second << endl;
    }
    return 0;
}
```



```

charMap.insert(MYMAP::value_type(5,'E')),
charMap.insert(MYMAP::value_type(4,'D'));

// Display the contents of the map.
cout << "\n\nContents of map:\n";
MYMAP::iterator iter;
for (iter= charMap.begin( ),iter != charMap.end( ); iter++)
{
    cout << "!" << (*iter).first << " -> ";
    cout << (*iter).second << endl;
}
return 0;
}

```

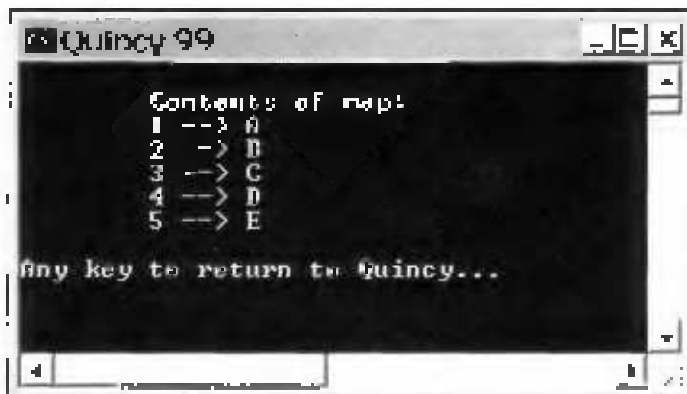


Figure 15.13

## Adding Elements to a map Using the [ ] Operator

၁။ map class template မှာပါသော [ ] operator ၏ အသုံးပြုနည်းကို insert( ) function ၏ အသုံးပြုနည်း element သို့မဟုတ် map object ပေါ်တွင် insert လုပ်ဆောင်နိုင်စေရန်၊ များသောအားဖြင့်အားဖြင့် Ex15.13 မှာ ပိုမိုအသုံးပြုနိုင်စေရန်အတွက် အသုံးပြုပါသည်။





## Removing map Elements

Listing 15.14 shows the `Ex1514.cpp` program. An empty map object is created and a new element is added using the `[]` operator. The `charMap.erase(iterator)` statement erases the second element and a new element is displayed. When the program is run, the sorted set elements are displayed: second element is 'B', it is erased, and new element is displayed. The program is shown in Listing 15.14.

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] = 'F';
    charMap[3] = 'C';

    cout << "Map contents of map:\n";
    MYMAP::iterator iter;
    for (iter = charMap.begin(); iter != charMap.end(); iter++)
    {
        cout << " " << (*iter).first << " -> ";
        cout << (*iter).second << endl;
    }

    iter = charMap.begin();
    charMap.erase(++iter);
}
```

```

// Display the new contents of the map.
cout << "\n\nContents of new map:\n";
for (iter= chaMap.begin( ), iter != chaMap.end( ); iter++)
{
    cout << "iter << (*iter).first << " -> ";
    cout << "iter << (*iter).second << endl;
}
return 0;
}

```



Figure 15.14

## Searching a map

Ex15.15.cpp program uses map object to store key-value pairs. Program searches for element. If key of Element found, program prints Element found. If key of Element not found, program prints Element not found.

```

// 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] = 'W';
    charMap[4] = 'D';
    charMap[2] = 'B';
    charMap[5] = 'E';
    charMap[3] = 'C';

    cout << "PrintContents of map:\n";
    MYMAP::iterator iter;
    for (iter= charMap.begin( ),iter != charMap.end( ); iter != )
    {
        cout << "First: ("<iter>).first << " => ";
        cout << ("iter).second << endl;
    }

    // find the D.
    MYMAP::iterator pos = charMap.find(4);

    if (pos == charMap.end())
        cout << "NoElement not found",
    else
        cout << "NoElement found: " << (*pos).second;

    cout << endl;
    return 0;
}

```

Ex15:5.cpp program is run. ABCDE is the map object. D is the element found. D is display under the cursor.

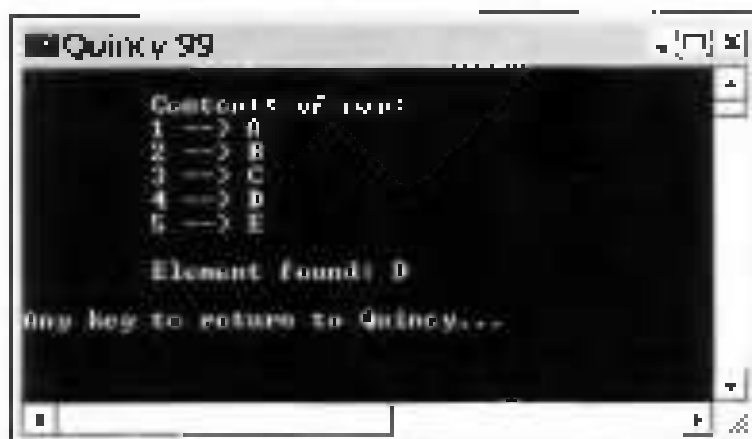


Figure 15-16

## Comparing maps

Example 15-16 program compares two map objects (2) and displays the contents of the first map object. The program uses the `map` object (2) and the `map` object (2) to compare the contents of the two maps and displays the results of the comparison.

```

// Listing 15-16: Comparing maps
#include <istream>
#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\nContents of first map:\n";
MYMAP iterator iter;
for (iter = charMap1.begin(); iter != charMap1.end(); iter++)
{
    cout << "it << (" * iter).first << " -> ";
    cout << (*iter).second << endl;
}
cout << endl;

// Create the second map object.
MYMAP charMap2;
charMap2[1] = 'F';
charMap2[4] = 'J';
charMap2[7] = 'G';
charMap2[5] = 'D';
charMap2[3] = 'H';

cout << "\n\nContents of second map:\n";
for (iter = charMap2.begin(); iter != charMap2.end(); iter++)
{
    cout << "it << (" * iter).first << " -> ";
    cout << (*iter).second << endl;
}
cout << endl;

// Compare the maps
if (charMap1 == charMap2)
    cout << "map1 == map2";
else if (charMap1 < charMap2)
    cout << "map1 < map2";
else
    cout << "map1 > map2";

cout << endl;
return 0;
}

```

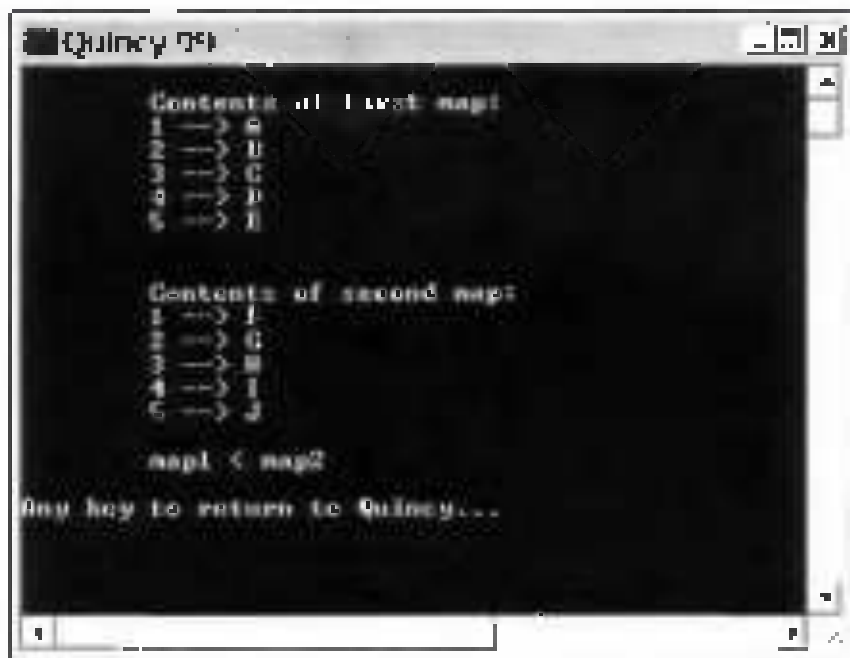


Figure 15.16

## 15.9 The multimap Class Template

A `multimap` object stores multiple elements in sorted order. In this program, you will use a `multimap` to store multiple elements. `multimap` is a generalization of `map` that allows for multiple elements for each key. In Listing 15.17, the program creates a `multimap` and displays its contents. The output shows the following: `A B B C C D E`. Note that the `multimap` stores multiple elements for each key.

// 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[Contents of multimap:]\n";
    MYMAP::iterator iter;
    for (iter = charMultimap.begin(); iter != charMultimap.end(); iter++)
    {
        cout << "k: " << (*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...

```

.. (25 / 37)



## Removing multimap Elements

❶ `erase` ကို အသုံးပြု၍ `Ex15.18.cpp` program ထဲထဲက `ABBCDDE` မှာ `multimap` ထဲထဲက create လုပ်ပြီး second element 'B' ကို erase လုပ်ပစ်ရင်းကားဖြင့် new set က `ABCDDE` ဖြစ်ပေါ်လာမည် လုပ်ပေးတဲ့ program ဖြစ်ပါတယ် ။ လေးလက်ပြန်ပေး။

```
// 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\nContents of multimap:\n";
    MYMAP::iterator iter;
    for (iter= charMultimap.begin( ); iter != charMultimap.end( ); iter++)
    {
        cout << "i" << (*iter).first << " --> ";
        cout << (*iter).second << endl;
    }

    // Erase the multimap's second element.
    iter = charMultimap.begin( );
```



```
first element: 10, second: 15, third: 20, fourth: 25, fifth: 30, sixth: 35, seventh: 40, eighth: 45, ninth: 50, tenth: 55, eleventh: 60, twelfth: 65, thirteenth: 70, fourteenth: 75, fifteenth: 80, sixteenth: 85, seventeenth: 90, eighteenth: 95, nineteenth: 100, twentieth: 105, twenty-first: 110, twenty-second: 115, twenty-third: 120, twenty-fourth: 125, twenty-fifth: 130, twenty-sixth: 135, twenty-seventh: 140, twenty-eighth: 145, twenty-ninth: 150, thirtieth: 155, thirty-first: 160, thirty-second: 165, thirty-third: 170, thirty-fourth: 175, thirty-fifth: 180, thirty-sixth: 185, thirty-seventh: 190, thirty-eighth: 195, thirty-ninth: 200, fortieth: 205, forty-first: 210, forty-second: 215, forty-third: 220, forty-fourth: 225, forty-fifth: 230, forty-sixth: 235, forty-seventh: 240, forty-eighth: 245, forty-ninth: 250, fiftieth: 255, fifty-first: 260, fifty-second: 265, fifty-third: 270, fifty-fourth: 275, fifty-fifth: 280, fifty-sixth: 285, fifty-seventh: 290, fifty-eighth: 295, fifty-ninth: 300, sixtieth: 305, sixty-first: 310, sixty-second: 315, sixty-third: 320, sixty-fourth: 325, sixty-fifth: 330, sixty-sixth: 335, sixty-seventh: 340, sixty-eighth: 345, sixty-ninth: 350, seventieth: 355, seventy-first: 360, seventy-second: 365, seventy-third: 370, seventy-fourth: 375, seventy-fifth: 380, seventy-sixth: 385, seventy-seventh: 390, seventy-eighth: 395, seventy-ninth: 400, eightieth: 405, eighty-first: 410, eighty-second: 415, eighty-third: 420, eighty-fourth: 425, eighty-fifth: 430, eighty-sixth: 435, eighty-seventh: 440, eighty-eighth: 445, eighty-ninth: 450, ninetieth: 455, ninety-first: 460, ninety-second: 465, ninety-third: 470, ninety-fourth: 475, ninety-fifth: 480, ninety-sixth: 485, ninety-seventh: 490, ninety-eighth: 495, ninety-ninth: 500, one hundred: 505, one hundred and one: 510, one hundred and two: 515, one hundred and three: 520, one hundred and four: 525, one hundred and five: 530, one hundred and six: 535, one hundred and seven: 540, one hundred and eight: 545, one hundred and nine: 550, one hundred and ten: 555, one hundred and eleven: 560, one hundred and twelve: 565, one hundred and thirteen: 570, one hundred and fourteen: 575, one hundred and fifteen: 580, one hundred and sixteen: 585, one hundred and seventeen: 590, one hundred and eighteen: 595, one hundred and nineteen: 600, one hundred and twenty: 605, one hundred and twenty-one: 610, one hundred and twenty-two: 615, one hundred and twenty-three: 620, one hundred and twenty-four: 625, one hundred and twenty-five: 630, one hundred and twenty-six: 635, one hundred and twenty-seven: 640, one hundred and twenty-eight: 645, one hundred and twenty-nine: 650, one hundred and thirty: 655, one hundred and thirty-one: 660, one hundred and thirty-two: 665, one hundred and thirty-three: 670, one hundred and thirty-four: 675, one hundred and thirty-five: 680, one hundred and thirty-six: 685, one hundred and thirty-seven: 690, one hundred and thirty-eight: 695, one hundred and thirty-nine: 700, one hundred and forty: 705, one hundred and forty-one: 710, one hundred and forty-two: 715, one hundred and forty-three: 720, one hundred and forty-four: 725, one hundred and forty-five: 730, one hundred and forty-six: 735, one hundred and forty-seven: 740, one hundred and forty-eight: 745, one hundred and forty-nine: 750, one hundred and fifty: 755, one hundred and fifty-one: 760, one hundred and fifty-two: 765, one hundred and fifty-three: 770, one hundred and fifty-four: 775, one hundred and fifty-five: 780, one hundred and fifty-six: 785, one hundred and fifty-seven: 790, one hundred and fifty-eight: 795, one hundred and fifty-nine: 800, one hundred and sixty: 805, one hundred and sixty-one: 810, one hundred and sixty-two: 815, one hundred and sixty-three: 820, one hundred and sixty-four: 825, one hundred and sixty-five: 830, one hundred and sixty-six: 835, one hundred and sixty-seven: 840, one hundred and sixty-eight: 845, one hundred and sixty-nine: 850, one hundred and seventy: 855, one hundred and seventy-one: 860, one hundred and seventy-two: 865, one hundred and seventy-three: 870, one hundred and seventy-four: 875, one hundred and seventy-five: 880, one hundred and seventy-six: 885, one hundred and seventy-seven: 890, one hundred and seventy-eight: 895, one hundred and seventy-nine: 900, one hundred and eighty: 905, one hundred and eighty-one: 910, one hundred and eighty-two: 915, one hundred and eighty-three: 920, one hundred and eighty-four: 925, one hundred and eighty-five: 930, one hundred and eighty-six: 935, one hundred and eighty-seven: 940, one hundred and eighty-eight: 945, one hundred and eighty-nine: 950, one hundred and ninety: 955, one hundred and ninety-one: 960, one hundred and ninety-two: 965, one hundred and ninety-three: 970, one hundred and ninety-four: 975, one hundred and ninety-five: 980, one hundred and ninety-six: 985, one hundred and ninety-seven: 990, one hundred and ninety-eight: 995, one hundred and ninety-nine: 1000, two hundred: 1005, two hundred and one: 1010, two hundred and two: 1015, two hundred and three: 1020, two hundred and four: 1025, two hundred and five: 1030, two hundred and six: 1035, two hundred and seven: 1040, two hundred and eight: 1045, two hundred and nine: 1050, two hundred and ten: 1055, two hundred and eleven: 1060, two hundred and twelve: 1065, two hundred and thirteen: 1070, two hundred and fourteen: 1075, two hundred and fifteen: 1080, two hundred and sixteen: 1085, two hundred and seventeen: 1090, two hundred and eighteen: 1095, two hundred and nineteen: 1100, two hundred and twenty: 1105, two hundred and twenty-one: 1110, two hundred and twenty-two: 1115, two hundred and twenty-three: 1120, two hundred and twenty-four: 1125, two hundred and twenty-five: 1130, two hundred and twenty-six: 1135, two hundred and twenty-seven: 1140, two hundred and twenty-eight: 1145, two hundred and twenty-nine: 1150, two hundred and thirty: 1155, two hundred and thirty-one: 1160, two hundred and thirty-two: 1165, two hundred and thirty-three: 1170, two hundred and thirty-four: 1175, two hundred and thirty-five: 1180, two hundred and thirty-six: 1185, two hundred and thirty-seven: 1190, two hundred and thirty-eight: 1195, two hundred and thirty-nine: 1200, two hundred and forty: 1205, two hundred and forty-one: 1210, two hundred and forty-two: 1215, two hundred and forty-three: 1220, two hundred and forty-four: 1225, two hundred and forty-five: 1230, two hundred and forty-six: 1235, two hundred and forty-seven: 1240, two hundred and forty-eight: 1245, two hundred and forty-nine: 1250, two hundred and fifty: 1255, two hundred and fifty-one: 1260, two hundred and fifty-two: 1265, two hundred and fifty-three: 1270, two hundred and fifty-four: 1275, two hundred and fifty-five: 1280, two hundred and fifty-six: 1285, two hundred and fifty-seven: 1290, two hundred and fifty-eight: 1295, two hundred and fifty-nine: 1300, two hundred and sixty: 1305, two hundred and sixty-one: 1310, two hundred and sixty-two: 1315, two hundred and sixty-three: 1320, two hundred and sixty-four: 1325, two hundred and sixty-five: 1330, two hundred and sixty-six: 1335, two hundred and sixty-seven: 1340, two hundred and sixty-eight: 1345, two hundred and sixty-nine: 1350, two hundred and seventy: 1355, two hundred and seventy-one: 1360, two hundred and seventy-two: 1365, two hundred and seventy-three: 1370, two hundred and seventy-four: 1375, two hundred and seventy-five: 1380, two hundred and seventy-six: 1385, two hundred and seventy-seven: 1390, two hundred and seventy-eight: 1395, two hundred and seventy-nine: 1400, two hundred and eighty: 1405, two hundred and eighty-one: 1410, two hundred and eighty-two: 1415, two hundred and eighty-three: 1420, two hundred and eighty-four: 1425, two hundred and eighty-five: 1430, two hundred and eighty-six: 1435, two hundred and eighty-seven: 1440, two hundred and eighty-eight: 1445, two hundred and eighty-nine: 1450, two hundred and ninety: 1455, two hundred and ninety-one: 1460, two hundred and ninety-two: 1465, two hundred and ninety-three: 1470, two hundred and ninety-four: 1475, two hundred and ninety-five: 1480, two hundred and ninety-six: 1485, two hundred and ninety-seven: 1490, two hundred and ninety-eight: 1495, two hundred and ninety-nine: 1500, three hundred: 1505, three hundred and one: 1510, three hundred and two: 1515, three hundred and three: 1520, three hundred and four: 1525, three hundred and five: 1530, three hundred and six: 1535, three hundred and seven: 1540, three hundred and eight: 1545, three hundred and nine: 1550, three hundred and ten: 1555, three hundred and eleven: 1560, three hundred and twelve: 1565, three hundred and thirteen: 1570, three hundred and fourteen: 1575, three hundred and fifteen: 1580, three hundred and sixteen: 1585, three hundred and seventeen: 1590, three hundred and eighteen: 1595, three hundred and nineteen: 1600, three hundred and twenty: 1605, three hundred and twenty-one: 1610, three hundred and twenty-two: 1615, three hundred and twenty-three: 1620, three hundred and twenty-four: 1625, three hundred and twenty-five: 1630, three hundred and twenty-six: 1635, three hundred and twenty-seven: 1640, three hundred and twenty-eight: 1645, three hundred and twenty-nine: 1650, three hundred and thirty: 1655, three hundred and thirty-one: 1660, three hundred and thirty-two: 1665, three hundred and thirty-three: 1670, three hundred and thirty-four: 1675, three hundred and thirty-five: 1680, three hundred and thirty-six: 1685, three hundred and thirty-seven: 1690, three hundred and thirty-eight: 1695, three hundred and thirty-nine: 1700, three hundred and forty: 1705, three hundred and forty-one: 1710, three hundred and forty-two: 1715, three hundred and forty-three: 1720, three hundred and forty-four: 1725, three hundred and forty-five: 1730, three hundred and forty-six: 1735, three hundred and forty-seven: 1740, three hundred and forty-eight: 1745, three hundred and forty-nine: 1750, three hundred and fifty: 1755, three hundred and fifty-one: 1760, three hundred and fifty-two: 1765, three hundred and fifty-three: 1770, three hundred and fifty-four: 1775, three hundred and fifty-five: 1780, three hundred and fifty-six: 1785, three hundred and fifty-seven: 1790, three hundred and fifty-eight: 1795, three hundred and fifty-nine: 1800, three hundred and sixty: 1805, three hundred and sixty-one: 1810, three hundred and sixty-two: 1815, three hundred and sixty-three: 1820, three hundred and sixty-four: 1825, three hundred and sixty-five: 1830, three hundred and sixty-six: 1835, three hundred and sixty-seven: 1840, three hundred and sixty-eight: 1845, three hundred and sixty-nine: 1850, three hundred and seventy: 1855, three hundred and seventy-one: 1860, three hundred and seventy-two: 1865, three hundred and seventy-three: 1870, three hundred and seventy-four: 1875, three hundred and seventy-five: 1880, three hundred and seventy-six: 1885, three hundred and seventy-seven: 1890, three hundred and seventy-eight: 1895, three hundred and seventy-nine: 1900, three hundred and eighty: 1905, three hundred and eighty-one: 1910, three hundred and eighty-two: 1915, three hundred and eighty-three: 1920, three hundred and eighty-four: 1925, three hundred and eighty-five: 1930, three hundred and eighty-six: 1935, three hundred and eighty-seven: 1940, three hundred and eighty-eight: 1945, three hundred and eighty-nine: 1950, three hundred and ninety: 1955, three hundred and ninety-one: 1960, three hundred and ninety-two: 1965, three hundred and ninety-three: 1970, three hundred and ninety-four: 1975, three hundred and ninety-five: 1980, three hundred and ninety-six: 1985, three hundred and ninety-seven: 1990, three hundred and ninety-eight: 1995, four hundred: 2000, four hundred and one: 2005, four hundred and two: 2010, four hundred and three: 2015, four hundred and four: 2020, four hundred and five: 2025, four hundred and six: 2030, four hundred and seven: 2035, four hundred and eight: 2040, four hundred and nine: 2045, four hundred and ten: 2050, four hundred and eleven: 2055, four hundred and twelve: 2060, four hundred and thirteen: 2065, four hundred and fourteen: 2070, four hundred and fifteen: 2075, four hundred and sixteen: 2080, four hundred and seventeen: 2085, four hundred and eighteen: 2090, four hundred and nineteen: 2095, four hundred and twenty: 2100, four hundred and twenty-one: 2105, four hundred and twenty-two: 2110, four hundred and twenty-three: 2115, four hundred and twenty-four: 2120, four hundred and twenty-five: 2125, four hundred and twenty-six: 2130, four hundred and twenty-seven: 2135, four hundred and twenty-eight: 2140, four hundred and twenty-nine: 2145, four hundred and thirty: 2150, four hundred and thirty-one: 2155, four hundred and thirty-two: 2160, four hundred and thirty-three: 2165, four hundred and thirty-four: 2170, four hundred and thirty-five: 2175, four hundred and thirty-six: 2180, four hundred and thirty-seven: 2185, four hundred and thirty-eight: 2190, four hundred and thirty-nine: 2195, four hundred and forty: 2200, four hundred and forty-one: 2205, four hundred and forty-two: 2210, four hundred and forty-three: 2215, four hundred and forty-four: 2220, four hundred and forty-five: 2225, four hundred and forty-six: 2230, four hundred and forty-seven: 2235, four hundred and forty-eight: 2240, four hundred and forty-nine: 2245, four hundred and fifty: 2250, four hundred and fifty-one: 2255, four hundred and fifty-two: 2260, four hundred and fifty-three: 2265, four hundred and fifty-four: 2270, four hundred and fifty-five: 2275, four hundred and fifty-six: 2280, four hundred and fifty-seven: 2285, four hundred and fifty-eight: 2290, four hundred and fifty-nine: 2295, four hundred and sixty: 2300, four hundred and sixty-one: 2305, four hundred and sixty-two: 2310, four hundred and sixty-three: 2315, four hundred and sixty-four: 2320, four hundred and sixty-five: 2325, four hundred and sixty-six: 2330, four hundred and sixty-seven: 2335, four hundred and sixty-eight: 2340, four hundred and sixty-nine: 2345, four hundred and seventy: 2350, four hundred and seventy-one: 2355, four hundred and seventy-two: 2360, four hundred and seventy-three: 2365, four hundred and seventy-four: 2370, four hundred and seventy-five: 2375, four hundred and seventy-six: 2380, four hundred and seventy-seven: 2385, four hundred and seventy-eight: 2390, four hundred and seventy-nine: 2395, four hundred and eighty: 2400, four hundred and eighty-one: 2405, four hundred and eighty-two: 2410, four hundred and eighty-three: 2415, four hundred and eighty-four: 2420, four hundred and eighty-five: 2425, four hundred and eighty-six: 2430, four hundred and eighty-seven: 2435, four hundred and eighty-eight: 2440, four hundred and eighty-nine: 2445, four hundred and ninety: 2450, four hundred and ninety-one: 2455, four hundred and ninety-two: 2460, four hundred and ninety-three: 2465, four hundred and ninety-four: 2470, four hundred and ninety-five: 2475, four hundred and ninety-six: 2480, four hundred and ninety-seven: 2485, four hundred and ninety-eight: 2490, four hundred and ninety-nine: 2495, five hundred: 2500, five hundred and one: 2505, five hundred and two: 2510, five hundred and three: 2515, five hundred and four: 2520, five hundred and five: 2525, five hundred and six: 2530, five hundred and seven: 2535, five hundred and eight: 2540, five hundred and nine: 2545, five hundred and ten: 2550, five hundred and eleven: 2555, five hundred and twelve: 2560, five hundred and thirteen: 2565, five hundred and fourteen: 2570, five hundred and fifteen: 2575, five hundred and sixteen: 2580, five hundred and seventeen: 2585, five hundred and eighteen: 2590, five hundred and nineteen: 2595, five hundred and twenty: 2600, five hundred and twenty-one: 2605, five hundred and twenty-two: 2610, five hundred and twenty-three: 2615, five hundred and twenty-four: 2620, five hundred and twenty-five: 2625, five hundred and twenty-six: 2630, five hundred and twenty-seven: 2635, five hundred and twenty-eight: 2640, five hundred and twenty-nine: 2645, five hundred and thirty: 2650, five hundred and thirty-one: 2655, five hundred and thirty-two: 2660, five hundred and thirty-three: 2665, five hundred and thirty-four: 2670, five hundred and thirty-five: 2675, five hundred and thirty-six: 2680, five hundred and thirty-seven: 2685, five hundred and thirty-eight: 2690, five hundred and thirty-nine: 2695, five hundred and forty: 2700, five hundred and forty-one: 2705, five hundred and forty-two: 2710, five hundred and forty-three: 2715, five hundred and forty-four: 2720, five hundred and forty-five: 2725, five hundred and forty-six: 2730, five hundred and forty-seven: 2735, five hundred and forty-eight: 2740, five hundred and forty-nine: 2745, five hundred and fifty: 2750, five hundred and fifty-one: 2755, five hundred and fifty-two: 2760, five hundred and fifty-three: 2765, five hundred and fifty-four: 2770, five hundred and fifty-five: 2775, five hundred and fifty-six: 2780, five hundred and fifty-seven: 2785, five hundred and fifty-eight: 2790, five hundred and fifty-nine: 2795, five hundred and sixty: 2800, five hundred and sixty-one: 2805, five hundred and sixty-two: 2810, five hundred and sixty-three: 2815, five hundred and sixty-four: 2820, five hundred and sixty-five: 2825, five hundred and sixty-six: 2830, five hundred and sixty-seven: 2835, five hundred and sixty-eight: 2840, five hundred and sixty-nine: 2845, five hundred and seventy: 2850, five hundred and seventy-one: 2855, five hundred and seventy-two: 2860, five hundred and seventy-three: 2865, five hundred and seventy-four: 2870, five hundred and seventy-five: 2875, five hundred and seventy-six: 2880, five hundred and seventy-seven: 2885, five hundred and seventy-eight: 2890, five hundred and seventy-nine: 2895, five hundred and eighty: 2900, five hundred and eighty-one: 2905, five hundred and eighty-two: 2910, five hundred and eighty-three: 2915, five hundred and eighty-four: 2920, five hundred and eighty-five: 2925, five hundred and eighty-six: 2930, five hundred and eighty-seven: 2935, five hundred and eighty-eight: 2940, five hundred and eighty-nine: 2945, five hundred and ninety: 2950, five hundred and ninety-one: 2955, five hundred and ninety-two: 2960, five hundred and ninety-three: 2965, five hundred and ninety-four: 2970, five hundred and ninety-five: 2975, five hundred and ninety-six: 2980, five hundred and ninety-seven: 2985, five hundred and ninety-eight: 2990, five hundred and ninety-nine: 2995, six hundred: 3000, six hundred and one: 3005, six hundred and two: 3010, six hundred and three: 3015, six hundred and four: 3020, six hundred and five: 3025, six hundred and six: 3030, six hundred and seven: 3035, six hundred and eight: 3040, six hundred and nine: 3045, six hundred and ten: 3050, six hundred and eleven: 3055, six hundred and twelve: 3060, six hundred and thirteen: 3065, six hundred and fourteen: 3070, six hundred and fifteen: 3075, six hundred and sixteen: 3080, six hundred and seventeen: 3085, six hundred and eighteen: 3090, six hundred and nineteen: 3095, six hundred and twenty: 3100, six hundred and twenty-one: 3105, six hundred and twenty-two: 3110, six hundred and twenty-three: 3115, six hundred and twenty-four: 3120, six hundred and twenty-five: 3125, six hundred and twenty-six: 3130, six hundred and twenty-seven: 3135, six hundred and twenty-eight: 3140, six hundred and twenty-nine: 3145, six hundred and thirty: 3150, six hundred and thirty-one: 3155, six hundred and thirty-two: 3160, six hundred and thirty-three: 3165, six hundred and thirty-four: 3170, six hundred and thirty-five: 3175, six hundred and thirty-six: 3180, six hundred and thirty-seven: 3185, six hundred and thirty-eight: 3190, six hundred and thirty-nine: 3195, six hundred and forty: 3200, six hundred and forty-one: 3205, six hundred and forty-two: 3210, six hundred and forty-three: 3215, six hundred and forty-four: 3220, six hundred and forty-five: 3225, six hundred and forty-six: 3230, six hundred and forty-seven: 3235, six hundred and forty-eight: 3240, six hundred and forty-nine: 3245, six hundred and fifty: 3250, six hundred and fifty-one: 3255, six hundred and fifty-two: 3260, six hundred and fifty-three: 3265, six hundred and fifty-four: 3270, six hundred and fifty-five: 3275, six hundred and fifty-six: 3280, six hundred and fifty-seven: 3285, six hundred and fifty-eight: 3290, six hundred and fifty-nine: 3295, six hundred and sixty: 3300, six hundred and sixty-one: 3305, six hundred and sixty-two: 3310, six hundred and sixty-three: 3315, six hundred and sixty-four: 3320, six hundred and sixty-five: 3325, six hundred and sixty-six: 3330, six hundred and sixty-seven: 3335, six hundred and sixty-eight: 3340, six hundred and sixty-nine: 3345, six hundred and seventy: 3350, six hundred and seventy-one: 3355, six hundred and seventy-two: 3360, six hundred and seventy-three: 3365, six hundred and seventy-four: 3370, six hundred and seventy-five: 3375, six hundred and seventy-six: 3380, six hundred and seventy-seven: 3385, six hundred and seventy-eight: 3390, six hundred and seventy-nine: 3395, six hundred and eighty: 3400, six hundred and eighty-one: 3405, six hundred and eighty-two: 3410, six hundred and eighty-three: 3415, six hundred and eighty-four: 3420, six hundred and eighty-five: 3425, six hundred and eighty-six: 3430, six hundred and eighty-seven: 3435, six hundred and eighty-eight: 3440, six hundred and eighty-nine: 3445, six hundred and ninety: 3450, six hundred and ninety-one: 3455, six hundred and ninety-two: 3460, six hundred and ninety-three: 3465, six hundred and ninety-four: 3470, six hundred and ninety-five: 3475, six hundred and ninety-six: 3480, six hundred and ninety-seven: 3485, six hundred and ninety-eight: 3490, six hundred and ninety-nine: 3495, seven hundred: 3500, seven hundred and one: 3505, seven hundred and two: 3510, seven hundred and three: 3515, seven hundred and four: 3520, seven hundred and five: 3525, seven hundred and six: 3530, seven hundred and seven: 3535, seven hundred and eight: 3540, seven hundred and nine: 3545, seven hundred and ten: 3550, seven hundred and eleven: 3555, seven hundred and twelve: 3560, seven hundred and thirteen: 3565, seven hundred and fourteen: 3570, seven hundred and fifteen: 3575, seven hundred and sixteen: 3580, seven hundred and seventeen: 3585, seven hundred and eighteen: 3590, seven hundred and nineteen: 3595, seven hundred and twenty: 3600, seven hundred and twenty-one: 3605, seven hundred and twenty-two: 3610, seven hundred and twenty-three: 3615, seven hundred and twenty-four: 3620, seven hundred and twenty-five: 3625, seven hundred and twenty-six: 3630, seven hundred and twenty-seven: 3635, seven hundred and twenty-eight: 3640, seven hundred and twenty-nine: 3645, seven hundred and thirty: 3650, seven hundred and thirty-one: 3655, seven hundred and thirty-two: 3660, seven hundred and thirty-three: 3665, seven hundred and thirty-four: 3670, seven hundred and thirty-five: 3675, seven hundred and thirty-six: 3680, seven hundred and thirty-seven: 3685, seven hundred and thirty-eight: 3690, seven hundred and thirty-nine: 3695, seven hundred and forty: 3700, seven hundred and forty-one: 3705, seven hundred and forty-two: 3710, seven hundred and forty-three: 3715, seven hundred and forty-four: 3720, seven hundred and forty-five: 3725, seven hundred and forty-six: 3730, seven hundred and forty-seven: 3735, seven hundred and forty-eight: 3740, seven hundred and forty-nine: 3745, seven hundred and fifty: 3750, seven hundred and fifty-one: 3755, seven hundred and fifty-two: 3760, seven hundred and fifty-three: 3765, seven hundred and fifty-four: 3770, seven hundred and fifty-five: 3775, seven hundred and fifty-six: 3780, seven hundred and fifty-seven: 3785, seven hundred and fifty-eight: 3790, seven hundred and fifty-nine: 3795, seven hundred and sixty: 3800, seven hundred and sixty-one: 3805, seven hundred and sixty-two: 3810, seven hundred and sixty-three: 3815, seven hundred and sixty-four: 3820, seven hundred and sixty-five: 3825, seven hundred and sixty-six: 3830, seven hundred and sixty-seven: 3835, seven hundred and sixty-eight: 3840, seven hundred and sixty-nine: 3845, seven hundred and seventy: 3850, seven hundred and seventy-one: 3855, seven hundred and seventy-two: 3860, seven hundred and seventy-three: 3865, seven hundred and seventy-four: 3870, seven hundred and seventy-five: 3875, seven hundred and seventy-six: 3880, seven hundred and seventy-seven: 3885, seven hundred and seventy-eight: 3890, seven hundred and seventy-nine: 3895, seven hundred and eighty: 3900, seven hundred and eighty-one: 3905, seven hundred and eighty-two: 3910, seven hundred and eighty-three: 3915, seven hundred and eighty-four: 3920, seven hundred and eighty-five: 3925, seven hundred and eighty-six: 3930, seven hundred and eighty-seven: 3935, seven hundred and eighty-eight: 3940, seven hundred and eighty-nine: 3945, seven hundred and ninety: 3950, seven hundred and ninety-one: 3955, seven hundred and ninety-two: 3960, seven hundred and ninety-three: 3965, seven hundred and ninety-four: 3970, seven hundred and ninety-five: 3975, seven hundred and ninety-six: 3980, seven hundred and ninety-seven: 3985, seven hundred and ninety-eight: 3990, seven hundred and ninety-nine: 3995, eight hundred: 4000, eight hundred and one: 4005, eight hundred and two: 4010, eight hundred and three: 4015, eight hundred and four: 4020, eight hundred and five: 4025, eight hundred and six: 4030, eight hundred and seven: 4035, eight hundred and eight: 4040, eight hundred and nine: 4045, eight hundred and ten: 4050, eight hundred and eleven: 4055, eight hundred and twelve: 4060, eight hundred and thirteen: 4065, eight hundred and fourteen: 4070, eight hundred and fifteen: 4075, eight hundred and sixteen: 4080, eight hundred and seventeen: 4085, eight hundred and eighteen: 4090, eight hundred and nineteen: 4095, eight hundred and twenty: 4100, eight hundred and twenty-one: 4105, eight hundred and twenty-two: 4110, eight hundred and twenty-three: 4115, eight hundred and twenty-four: 4120, eight hundred and twenty-five: 4125, eight hundred and twenty-six: 4130, eight hundred and twenty-seven: 4135, eight hundred and twenty-eight: 4140, eight hundred and twenty-nine: 4145, eight hundred and thirty: 4150, eight hundred and thirty-one: 4155, eight hundred and thirty-two: 4160, eight hundred and thirty-three: 4165, eight hundred and thirty-four: 4170, eight hundred and thirty-five: 4175, eight hundred and thirty-six: 4180, eight hundred and thirty-seven: 4185, eight hundred and thirty-eight: 4190, eight hundred and thirty-nine: 4195, eight hundred and forty: 4200, eight hundred and forty-one: 4205, eight hundred and forty-two: 4210, eight hundred and forty-three: 4215, eight hundred and forty-four: 4220, eight hundred and forty-five: 4225, eight hundred and forty-six: 4230, eight hundred and forty-seven: 4235, eight hundred and forty-eight: 4240, eight hundred and forty-nine: 4245, eight hundred and fifty: 4250, eight hundred and fifty-one: 4255, eight hundred and fifty-two: 4260, eight hundred and fifty-three: 4265, eight hundred and fifty-four: 4270, eight hundred and fifty-five: 4275, eight hundred and fifty-six: 4280, eight hundred and fifty-seven: 4285, eight hundred and fifty-eight: 4290, eight hundred and fifty-nine: 4295, eight hundred and sixty: 4300, eight hundred and sixty-one: 4305, eight hundred and sixty-two: 4310, eight hundred and sixty-three: 4315, eight hundred and sixty-four: 4320, eight hundred and sixty-five: 4325, eight hundred and sixty-six: 4330, eight hundred and sixty-seven: 4335, eight hundred and sixty-eight: 4340, eight hundred and sixty-nine: 4345, eight hundred and seventy: 4350, eight hundred and seventy-one: 4355, eight hundred and seventy-two: 4360, eight hundred and seventy-three: 4365, eight hundred and seventy-four: 4370, eight hundred and seventy-five: 4375, eight hundred and seventy-six: 4380, eight hundred and seventy-seven: 4385, eight hundred and seventy-eight: 4390, eight hundred and seventy-nine: 4395, eight hundred and eighty: 4400, eight hundred and eighty-one: 4405, eight hundred and eighty-two: 4410, eight hundred and eighty-three: 4415, eight hundred and eighty-four: 4420, eight hundred and eighty-five: 4425, eight hundred and eighty-six: 4430, eight hundred and eighty-seven: 4435, eight hundred and eighty-eight: 4440, eight hundred and eighty-nine: 4445, eight hundred and ninety: 4450, eight hundred and ninety-one: 4455, eight hundred and ninety-two: 4460, eight hundred and ninety-three: 4465, eight hundred and ninety-four: 4470, eight hundred and ninety-five: 4475, eight hundred and ninety-six: 4480, eight hundred and ninety-seven: 4485, eight hundred and ninety-eight: 4490, eight hundred and ninety-nine: 4495, nine hundred: 4500, nine hundred and one: 4505, nine hundred and two: 4510, nine hundred and three: 4515, nine hundred and four: 4520, nine hundred and five: 4525, nine hundred and six: 4530, nine hundred and seven: 4535, nine hundred and eight: 4540, nine hundred and nine: 4545, nine hundred and ten: 4550, nine hundred and eleven: 4555, nine hundred and twelve: 4560, nine hundred and thirteen: 4565, nine hundred and fourteen: 4570, nine hundred and fifteen: 4575, nine hundred and sixteen: 4580, nine hundred and seventeen: 4585, nine hundred and eighteen: 4590, nine hundred and nineteen: 4595, nine hundred and twenty: 4600, nine hundred and twenty-one: 4605, nine hundred and twenty-two: 4610, nine hundred and twenty-three: 4615, nine hundred and twenty-four: 4620, nine hundred and twenty-five: 4625, nine hundred and twenty-six: 4630, nine hundred and twenty-seven: 4635, nine hundred and twenty-eight: 4640, nine hundred and twenty-nine: 4645, nine hundred and thirty: 4650, nine hundred and thirty-one: 4655, nine hundred and thirty-two: 4660, nine hundred and thirty-three: 4665, nine hundred and thirty-four: 4670, nine hundred and thirty-five: 4675, nine hundred and thirty-six: 4680, nine hundred and thirty-seven: 4685, nine hundred and thirty-eight: 4690, nine hundred and thirty-nine: 4695, nine hundred and forty: 4700, nine hundred and forty-one: 4705, nine hundred and forty-two: 4710, nine hundred and forty-three: 4715, nine hundred and forty-four: 4720, nine hundred and forty-five: 4725, nine hundred and forty-six: 4730, nine hundred and forty-seven: 4735, nine hundred and forty-eight: 4740, nine hundred and forty-nine: 4745, nine hundred and fifty: 4750, nine hundred and fifty-one: 4755, nine hundred and fifty-two: 4760, nine hundred and fifty-three: 4765, nine hundred and fifty-four: 4770, nine hundred and fifty-five: 4775, nine hundred and fifty-six: 4780, nine hundred and fifty-seven: 4785, nine hundred and fifty-eight: 4790, nine hundred and fifty-nine: 4795, nine hundred and sixty: 4800, nine hundred and sixty-one: 4805, nine hundred and sixty-two: 4810, nine hundred and sixty-three: 4815, nine hundred and sixty-four: 4820, nine hundred and sixty-five: 4825, nine hundred and sixty-six: 4830, nine hundred and sixty-seven: 4835, nine hundred and sixty-eight: 4840, nine hundred and sixty-nine: 4845, nine hundred and seventy: 4850, nine hundred and seventy-one: 4855, nine hundred and seventy-two: 4860, nine hundred and seventy-three: 4865, nine hundred and seventy-four: 4870, nine hundred and seventy-five: 4875, nine hundred and seventy-six: 4880, nine hundred and seventy-seven: 4885, nine hundred and seventy-eight: 4890, nine hundred and seventy-nine: 4895, nine hundred and eighty: 4900, nine hundred and eighty-one: 4905, nine hundred and eighty-two: 4910, nine hundred and eighty-three: 4915, nine hundred and eighty-four: 4920, nine hundred and eighty-five: 4925, nine hundred and eighty-six: 4930, nine hundred and eighty-seven: 4935, nine hundred and eighty-eight: 4940, nine hundred and eighty-nine: 4945, nine hundred and ninety: 4950, nine hundred and ninety-one: 4955, nine hundred and ninety-two: 4960, nine hundred and ninety-three: 4965, nine hundred and ninety-four: 4970, nine hundred and ninety-five: 4975, nine hundred and ninety-six: 4980, nine hundred and ninety-seven: 4985, nine hundred and ninety-eight: 4990, nine hundred and ninety-nine: 4995, ten thousand: 5000, ten thousand and one: 5005, ten thousand and two: 5010, ten thousand and three: 5015, ten thousand and four: 5020, ten thousand and five: 5025, ten thousand and six: 5030, ten thousand and seven: 5035, ten thousand and eight: 5040, ten thousand and nine: 5045, ten thousand and ten: 5050, ten thousand and eleven: 5055, ten thousand and twelve: 5060, ten thousand and thirteen: 5065, ten thousand and fourteen: 5070, ten thousand and fifteen: 5075, ten thousand and sixteen: 5080, ten thousand and seventeen: 5085, ten thousand and eighteen: 5090, ten thousand and nineteen: 5095, ten thousand and twenty: 5100, ten thousand and twenty-one: 5105, ten thousand and twenty-two: 5110, ten thousand and twenty-three: 5115, ten thousand and twenty-four: 5120, ten thousand and twenty-five: 5125, ten thousand and twenty-six: 5130, ten thousand and twenty-seven: 5135, ten thousand and twenty-eight: 5140, ten thousand and twenty-nine: 5145, ten thousand and thirty: 5150, ten thousand and thirty-one: 5155, ten thousand and thirty-two: 5160, ten thousand and thirty-three: 5165, ten thousand and thirty-four: 5170, ten thousand and thirty-five: 5175, ten thousand and thirty-six: 5180, ten thousand and thirty-seven: 5185, ten thousand and thirty-eight: 5190, ten thousand and thirty-nine: 5195, ten thousand and forty: 5200, ten thousand and forty-one: 5205, ten thousand and forty-two: 5210, ten thousand and forty-three: 5215, ten thousand and forty-four: 5220, ten thousand and forty-five: 5225, ten thousand and forty-six: 5230, ten thousand and forty-seven: 5235, ten thousand and forty-eight: 5240, ten thousand and forty-nine: 5245, ten thousand and fifty: 5250, ten thousand and fifty-one: 5255, ten thousand and fifty-two: 5260, ten thousand and fifty-three: 5265, ten thousand and fifty-four: 5270, ten thousand and fifty-five: 5275, ten thousand and fifty-six: 5280, ten thousand and fifty-seven: 5285, ten thousand and fifty-eight: 5290, ten thousand and fifty-nine: 5295, ten thousand and sixty: 5300, ten thousand and sixty-one: 5305, ten thousand and sixty-two: 5310, ten thousand and sixty-three: 5315, ten thousand and sixty-four: 5320, ten thousand and sixty-five: 5325, ten thousand and sixty-six: 5330, ten thousand and sixty-seven: 5335, ten thousand and sixty-eight: 5340, ten thousand and sixty-nine: 5345, ten thousand and seventy: 5350, ten thousand and seventy-one: 5355, ten thousand and seventy-two: 5360, ten thousand and seventy-three: 5365, ten thousand and seventy-four: 5370, ten thousand and seventy-five: 5375, ten thousand and seventy-six: 5380, ten thousand and seventy-seven: 5385, ten thousand and seventy-eight: 5390, ten thousand and seventy-nine: 5395, ten thousand and eighty: 5400, ten thousand and eighty-one: 5405, ten thousand and eighty-two: 5410, ten thousand and eighty-three: 5415, ten thousand and eighty-four: 5420, ten thousand and eighty-five: 5425, ten thousand and eighty-six: 5430, ten thousand and eighty-seven: 5435, ten thousand and eighty-eight: 5440, ten thousand and eighty-nine: 5445, ten thousand and ninety: 5450, ten thousand and ninety-one: 5455, ten thousand and ninety-two: 5460, ten thousand and ninety-three: 5465, ten thousand and ninety-four: 5470, ten thousand and ninety-five:
```

```

        cout << "Element found: ";
        cout << 'a' << (*iter).first << " -> ";
        cout << (*iter).second << endl;
        cout << "(Next element: ";
        cout << 'b' << (*iter).first << " -> ";
        cout << (*iter).second << endl;
    }
    cout << endl;
    return 0;
}

```

```

Quincy 99
Contents of multimap:
a -> 0
b -> 1
c -> 2
d -> 3
e -> 4
f -> 5
g -> 6
h -> 7

Element found: e -> 4
Next element: f -> 5

Any key to return to Quincy...

```

Figure 9-10

## Comparing multimaps

Example 9-11 shows a program that compares two objects of the `multimap` class. The program compares the objects `multimap1` and `multimap2` using the `ABBCDE` and `CDEFFGE` keys. The program uses the `multimap::multimap2` constructor to create the second `multimap` object. The program uses the `multimap::display` function to display the contents of each `multimap` object.

```

// Listing 15.20: Comparing multimaps
#include <sstream>
#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,'X'));
    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,'E'));
    charMultimap.insert(MYMAP::value_type(6,'D'));

    cout << "PrintContents of first multimap:\n";
    MYMAP::iterator iter;
    for (iter=charMultimap.begin(); iter != charMultimap.end(); iter++)
    {
        cout << "k: " << (*iter).first << " -> " << "v: " << (*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'));
}

```

```

Galaxy 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 -> E
5 -> F
6 -> G
7 -> E

multimap1 < multimap2
Any key to return to Galaxy...

```

Figure 10.10

```

charMultimap2.insert(MYMAP::value_type(6, 'G'));

cout << "\n\nContents of second multimap:\n";

for (iter = charMultimap2.begin(); iter != charMultimap2.end(); iter++)
{
    cout << "!" << (iter).first << " -> ";
    cout << (iter).second << endl;
}
cout << endl;

// Compare the multimaps
if (charMultimap == charMultimap2)
    cout << "charMultimap == multimap2";
else if (charMultimap < charMultimap2)

```

```

        cout << "multimap1 < multimap2";
    else
        cout << "multimap1 > multimap2";

    cout << endl;
    return 0;
}

```

## 29.9 User-Defined Predicates

Ex 15.21.cpp program uses compare class predicate to define custom map object compare element using compare\_and\_sort <S, compare> program. It is a predicate user class using class predicate name to compare using overloaded ( ) operator. It is a user-defined program to run the example of ( ) ( ) operator.

```

// 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(5, '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(6, 'E'));
charMap.insert(map<int, char>::value_type(1, 'A'));

// Display the contents of the map.
cout << endl << "Contents of map:\n";
map<int, char>::iterator iter;
for (iter = charMap.begin(); iter != charMap.end(); iter++)
{
    cout << (*iter).first << " -> " << (*iter).second << endl;
}
return 0;
}

```

```

Query (9)
Contents of map:
1 -> A
2 -> B
3 -> C
4 -> D
5 -> E
6 -> F
7 -> G
any key to return to Coling...

```

Figure 14-14





Standard Template Library များကိုပိုမိုကောင်းမွန်စေရန် ဆောင်ရွက်ပုံကို type ကိုက်ညီစေရန်နှင့် ပိုမိုကောင်းမွန်စေရန် ရည်ရွယ်ချက်ဖြင့် ဆောင်ရွက်ထားသည်။ ထို့အပြင် ဆောင်ရွက်ထားသည့် စည်းကမ်းချက်များကို အောက်ဖော်ပြပါအတိုင်း ဖော်ပြထားသည်။

၆.၂။ ထပ်မံဆောင်ရွက်ပုံများ generic algorithm အားဆောင်ရွက်ခြင်း (၆.၂)။ STL မှ generic algorithm များကို အုပ်စု (၄) ခုဖြင့် ညွှန်ကြားနိုင်သော စာညွှန်း ( ) Non-modifying sequence algorithms ( ) Modifying sequence algorithms ( ) Sorting algorithms ( ) (၄) Numeric algorithm ကိုဖြည့်ထည့်သည့် အားဖြင့် non-modifying sequence algorithms ကိုထပ်မံဆောင်ရွက်သည့် sequence ကို modify သည့် function apply ကိုထပ်မံဆောင်ရွက်သည့် count ( ) function ကိုဖြင့် sequence ကိုဆောင်ရွက်သည့် element ကိုထပ်မံဆောင်ရွက်သည့် ဆောင်ရွက်ပုံများကို sequence ကိုဖြင့်ဆောင်ရွက်သည့် ဆောင်ရွက်ပုံ။ STL မှ non-modifying sequence algorithm (၄) များကို ညွှန်ကြားနိုင်သော စာညွှန်း adjacent find( ) find( ) find\_end( ) find\_first( ) count( ) mismatch( ) equal( ) for\_each( ) search( ) algorithm ကိုဖြင့်ဆောင်ရွက်ပုံများကို ဆောင်ရွက်ပုံများကို အောက်ဖော်ပြပါအတိုင်း ဖော်ပြထားသည်။

## 16.6 Using the adjacent\_find( ) Function

Ex1601.cpp program constructs set object and inserts set of values using adjacent\_find( ) generic algorithm. It also prints the program [Figure 16-1] using algorithm on set object element copy, modify and delete elements.

```
// 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 << "intContents of set int'
    multiset<int, less<int> >::iterator i = intSet.begin( );
    for (int x=0; x < intSet.size( ); ++x)
        cout << "i' x << "j(++ i << endl;
    cout << endl;

    // Find the first pair of equal values
    cout << "i: first matching pair: {n";
    i = adjacent_find (intSet.begin( ), intSet.end( ));
```



## ၁၆.၂ Using the count( ) Function

၁၆.၂.၁ မူပိုင်ခွင့်ပြုထားတဲ့ Ext1602.cpp program သည် set object တစ်ခုကိုတည်ဆောက်ပြီး element တစ်ခုခု, အထူးသတိပြုထားတဲ့ program မှ count( ) algorithm ကိုသုံးပြီး အရေအတွက်ပြုပေးပါ။

```
// 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(Contents 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(Number of 8s - " << cnt << endl;

    return 0;
}
```

Ex1502.cpp program ၏ run ထိန်းချုပ်ချိန်တွင် create သာသော sequence ၇- (8) ကို ပြသပေးခြင်းဖြစ်ပြီး အမှတ်အသားပြင်ဆင်မှုများကို (ပုံ 16.2) ကိုကြည့်ပါ။



( ပုံ 16.2 )

## ၁၆.၃ Using the for\_each( ) Function

Ex1603.cpp program သည် for\_each( start, end, func\_call) ကိုကား (၁) ကိုကား (၂) ကို sequence အပေါ်မှ element တစ်ခုစီ sorted order မှာ display ပြုလုပ်ပေးပြီး program သည် count( ) algorithm သို့မဟုတ် စီမံခန့်ခွဲမှုကို ပြုလုပ်ပေးသည်။

```
// Using 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\nContents of set:\n";
    for_each(intSet.begin( ), intSet.end( ), showVal);

    return 0;
}

```

```

Quincy 99
Contents of set:
1
3
5
8
8
8
10

Any key to return to Quincy...

```

© (sk. c)

## ၁၆.၄ Using the fill( ) Function

sequence operation ဆောင်ရွက်ခြင်းတို့ကို container မှာ modify ပြုလုပ်ရာ၌ algorithm ရော့။ Mutating sequence algorithm သို့မဟုတ် ပုံမှန် sequence operation ဆောင်ရွက်ရာ၌ same sequence ရှိတိုင်း ဆောင်ရွက်ရာမှ copy ကို သုံးသပ်ပေးခြင်းကို sequence operation မဟုတ်။ container content မတူပဲ၊ ကိုယ်တိုင် သုံးသပ်ပေးခြင်းပင်။ Mutating sequence algorithm ဆောင်ရွက်ခြင်းတို့မှာ swap( ), copy( ), copy\_backward( ), fill( ), generate( ), partition( ), random\_shuffle( ), swap( ), swap\_ranges( ), remove( ), spaceof( ), rotate( ), reverse( ), transform( ), unique( ) တို့ဖြစ်ကြသည်။ Ex16.4.cpp program မှ fill( ) algorithm ကို အသုံးပြုခြင်းကို ကြည့်ပါ။

// Listing 16.4: Using the fill( ) function

```
#include <iostream>
#include <vector>
#include <algorithm>

void showVal(int val)
{
    cout << "!" << val << endl;
}

int main( )
{
    vector<int> intVector; // Create the vector object

    for (int x=0; x<6; ++x)
        intVector.push_back(x);
    cout << "int(Contents 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 << "int(Contents of vector):\n";
```





```

// Listing 16.5: Using the random_shuffle( ) function
#include <sstream>
#include <vector>
#include <algorithm>

void showVal(int val)
{
    cout << " " << 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 << "Original contents of vector:\n";
    for_each (intvector.begin( ), intvector.end( ), showVal);

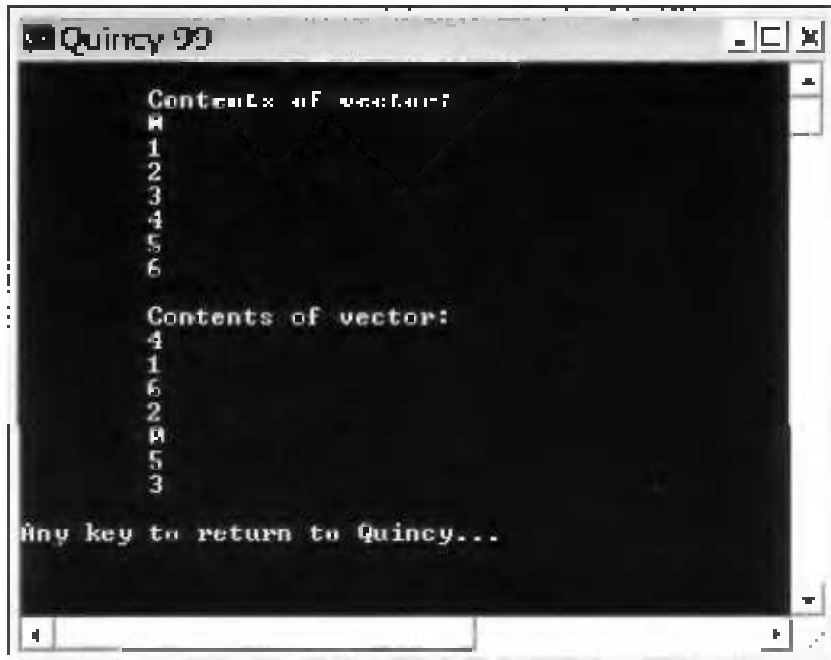
    // Shuffle the vector
    random_shuffle (intvector.begin( ), intvector.end( ));

    // Display the contents of the new vector.
    cout << "New contents of vector:\n";
    for_each (intvector.begin( ), intvector.end( ), showVal);

    return 0;
}

```

\* The `Ex06165.cpp` program is run as follows: `g++ Ex06165.cpp -std=c++11`. The program displays the original sequence and modify it by using `random_shuffle( )` function.



ရ (၁၆.၅)

## ၁၆.၆ Using the partition( ) Function

Ex1606.cpp program သာ sequence တစ်ခုမှာပင်တဲ့ element မတူသေး သင်ဆက်သားတဲ့ element ညွှန်လမ်းမှာရှိ သီးသန့်ညွှန်ပုဒ်ပြီး sequence ပုံတစ်ပုံအပေါ်ပိုင်းမှာ မတူသေးတဲ့ program မြင်ပါကပါ။ partition( ) algorithm တွေးကြည့်မှာရှိ အားတင်ပါ။ program မှာအဖွဲ့အစည်းပုံပါ။

```
// Listing 16.6: Using the partition( ) function
```

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
void showVa (int val)
```

```
{ cout << 't' << val << end; }
```

```

bool equals5 (int val)
{
    return (val == 5);
}

int main ()
{
    // Create the vector object
    vector<int> intVector;

    // Populate the vector with values
    intVector.push_back(6);
    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 << "x\n";
    // Display contents of vector 'x'
    for_each (intVector.begin(), intVector.end(), showVal);

    // Partition the vector
    part(1)(intVector.begin(), intVector.end(), equals5);

    // Display the contents of the new vector
    cout << "y\n";
    // Display contents of vector 'y'
    for_each (intVector.begin(), intVector.end(), showVal);

    return 0;
}

```

Ex1E06.cpp program will run as follows: `g++ Ex1E06.cpp` sequence element program (5) `./Ex1E06` sequence element program (5) `partition operation (5) (5) (1) (7) (5) (2) (5)` `partition operation (7) (5) (1) (2) (5)` `partition operation (5) (5) (2) (5)`

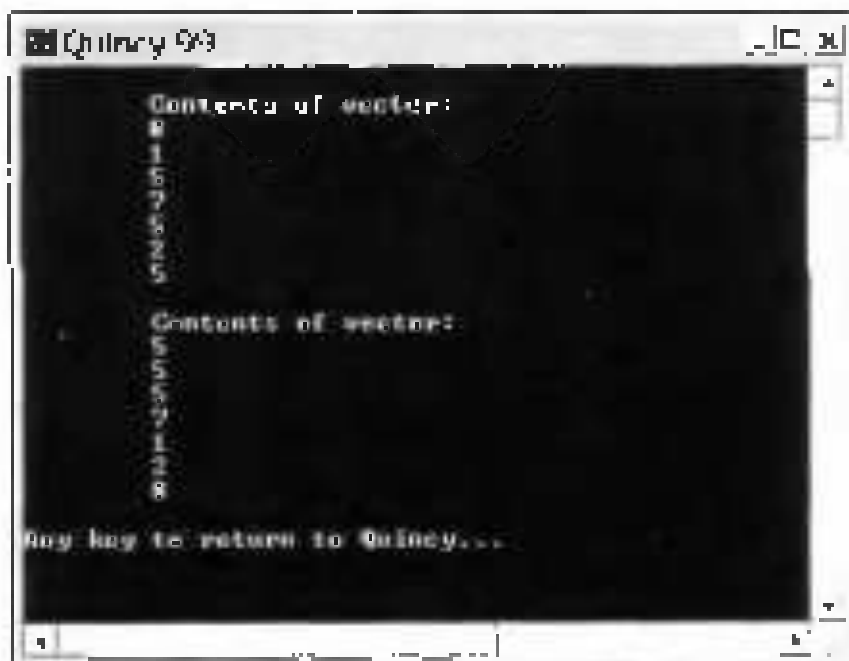


Figure 16.6

## 16.7 Using the rotate( ) Function

Listing 16.7 shows a program that rotates a sequence of characters. The element type is `char`, and the container is `std::string`. The program uses the `rotate( )` algorithm to rotate the sequence. The program is shown in Listing 16.7.

// Listing 16.7. Using the rotate( ) function

```
#include <string>
#include <vector>
#include <algorithm>

void showVec(char val)
{   cout << " " << val << " "; }
}
```

```

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 << "Initial contents 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 << "\nNew contents of vector:\n";
    for_each(charVector.begin(), charVector.end(), showVal);

    return 0;
}

```

The following program will run the same code as above to create a vector of THREE HE sequence and rotate 6 elements. After the rotate operation, the sequence will be IHE THREE.



Figure 16.8

## 16.8 Using the sort( ) Function

සමස්තවශයෙන්ම Sorting algorithm යැයි කියවන්නේ sort( ), partial\_sort( ), merge( ) යන සෑම algorithm යැයි කියවන්නේ Ex1608.cpp program යන සෑම sequence මඳායුතියේ char element යැයි කියවන්නේ sort( ) algorithm program යන සෑම A to Z කෙරෙහි 16.8.e) හි Ex1608.cpp program හි run කිරීමෙනි.

```

// 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('Q');
    charVector.push_back('O');
    charVector.push_back('C');
    charVector.push_back('H');
    charVector.push_back('M');
    charVector.push_back('Y');
    cout << "\n{Contents of vector:\n";
    for_each(charVector.begin(), charVector.end(), showVal);

    sort(charVector.begin(), charVector.end());

    // Display the contents of the new vector.
    cout << "\n{sorted contents of vector:\n";
    for_each(charVector.begin(), charVector.end(), showVal);
    cout << endl;
    return 0;
}

```

```

Quincy 09
Contents of vector
Z D F S Q O C H M Y

Contents of vector
A C D F G H Q S Y Z

Press any key to return to Quincy...

```

Figure 10-10

## Using the `partial_sort()` Function

Ex 16.9: C++ program that sorts (10) char element, sorted sequence of `vector` of char (5) elements. Use `partial_sort()` algorithm.

// Using 16.9 Using the `partial_sort()` function

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
#include <string>
```

```
void showVec(string val)
```

```
{ cout << " " << val << endl; }
```

```
int main() 
```

```
{
```

```
    // Create the vector object
```

```
    vector<string> strVector;
```

```
    // Populate the vector with values
```

```
    strVector.push_back("Zebra");
```

```
    strVector.push_back("Diner");
```

```
    strVector.push_back("Fish");
```

```
    strVector.push_back("Snake");
```

```
    strVector.push_back("Box");
```

```
    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 << "myContents 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 << "myContents of vector:\n";
for_each (strVector.begin( ), strVector.end( ), showVal);
return 0;

```

```

QtQuincy 99
Contents of vector:
Zebra
bear
Fish
Snake
Bat
Cat
Bird
Turtle
Horse
Cow

Contents of vector:
Bat
Bird
Cat
Cow
Deer
Zebra
Snake
Turtle
Horse
Fish

Any key to return to Galaxy...

```

☺ (26/0)

## 16.10 Using the merge( ) Function

Ex16.10.cpp program creates two vector objects (2) and concatenates program (3) elements merge sorted element into a single sorted (4). (5) program of run (6) execution.

```
// 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\nContents of vector1:\n";
```

```

for_each (strVector1.begin( ), strVector1.end( ), showVal);
cout << "\n\nContents of vector2:\n\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\nContents of vector3:\n\n";
for_each(strVector3.begin( ), strVector3.end( ), showVal);
return 0;
}

```

```

Quincy 99
Contents of vector1:
Zebra
Deer
Fish

Contents of vector2:
Cat
Bird
Turtle
Horse

Contents of vector3:
Bird
Cat
Deer
Fish
Horse
Turtle
Zebra

Any key to return to Quincy...

```

## 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 << 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("Bear");
    strVector1.push_back("Fish");
    strVector1.push_back("Snake");
    strVector1.push_back("Bat");

    strVector2.push_back("Lion");
    strVector2.push_back("Antelope");
    strVector2.push_back("Turtle");
    strVector2.push_back("Snake");
    strVector2.push_back("Sheep");

    // Sort the vectors.
    sort (strVector1.begin(), strVector1.end());
    sort (strVector2.begin(), strVector2.end());
}
```

```

// Display the contents of the vectors.
cout << "\n\n(Contents of vector1:\n";
for each (strVector1.begin( ), strVector1.end( ), showVal);
cout << endl;
cout << "\n\n(Contents 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 << "\n\nFound sorted range.\n";
else
    cout << "\n\nDid not find sorted range.\n";
return 0;
}

```

```

Quincy 99
Contents of vector1:
Rat
Deer
Fish
Snake
Zebra

Contents of vector2:
Antelope
Deer
Fish
Snake
Turtle

Found sorted range.

Any key to return to Quincy...

```

{ 33- }

## ၁၆.၁၁ Using the accumulate( ) Function

အောက်ဖော်ပြပါ accumulate( ) inner\_product( ) partial\_sum( ) adjacent\_difference( ) အသုံးပြု Numerical algorithm အဖြစ်သည့် Ex16.12.cpp program တွင် accumulate( ) algorithm အကဲဖြတ်ပြီး vector object ဝင်ရောက်မှုများကို ဝင်ရောက်မှုအားဖြင့် program ဖြစ်နိုင်သည့် ( ၁၆.၁၂ ) မှ program မှ run ပြုလုပ်ပါ။ အောက်ဖော်ပြပါ

```
// Using 16.12: Using the accumulate( ) function
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>

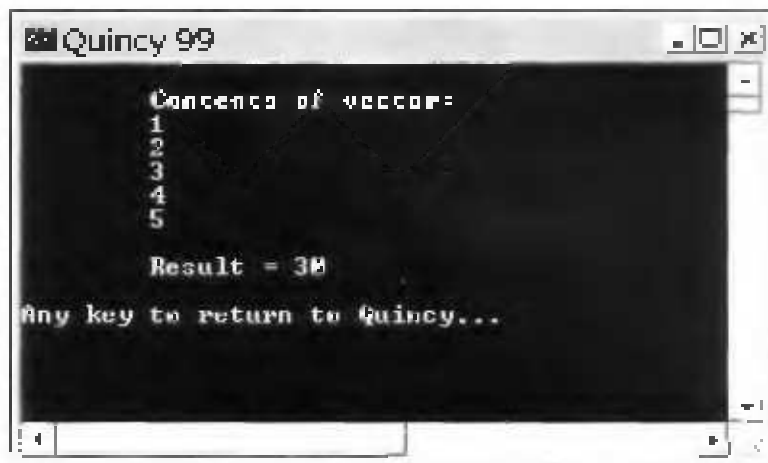
void showVal(int val)
{ cout << " " << val << endl; }

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 << "intVec contents 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 << "Result = " << result << endl;

    return 0;
}
```



(၂၆.၁၅)

## ၁၆.၁၂ Using the inner\_product( ) Function

ထောက်ပံ့သော vector object (၇) မှ ရလာ {0,1,2,3,4} နှင့် {2,3,4,5,6} တို့ကို inner product ရှင်းထုတ်ရန် (0\*2+1\*3+2\*4+3\*5+4\*6) တွေ့ရှိရသောအခါ ဝ) နံပါတ်ကို [x1613.cpp program မှာအစီအစဉ် (၂၆.၁၅) မှာ ကိုယ်တိုင် ရေးသားကြည့်ရပါ။

// 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\nContents of vector1:\n";
    for_each (intVector1.begin( ), intVector1.end( ), showVal);
    cout << "\n\nContents 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 << "\n\nResult = " << 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...

```

© (2006, 2007)



## Using the `partial_sum()` Function

Ex1614.cpp program uses `vector` object `mVec1` (i.e., {2,3,4,5,6}) and second vector `mVec2` and `partial_sum()` algorithm to calculate {2, 3+2, 4+(3+2), 5+(4+3+2), 6+(5+4+3+2)}. Create your program as follows: `g++ ex1614.cpp -std=c++11 -o program16_33` and execute it as `./program16_33`.

```
// Listing 16.14 Using the partial_sum() function
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>

void showVal(int val)
{
    cout << " " << val << endl;
}

int main()
{
    // Create the vector objects.
    vector<int> mVector1;
    vector<int> mVector2(4);

    // Populate the vector.
    for (int x=2; x<=6; ++x) mVector1.push_back(x);

    // Display the contents of the vector.
    cout << "Contents of vector1:\n";
    for_each(mVector1.begin(), mVector1.end(), showVal);

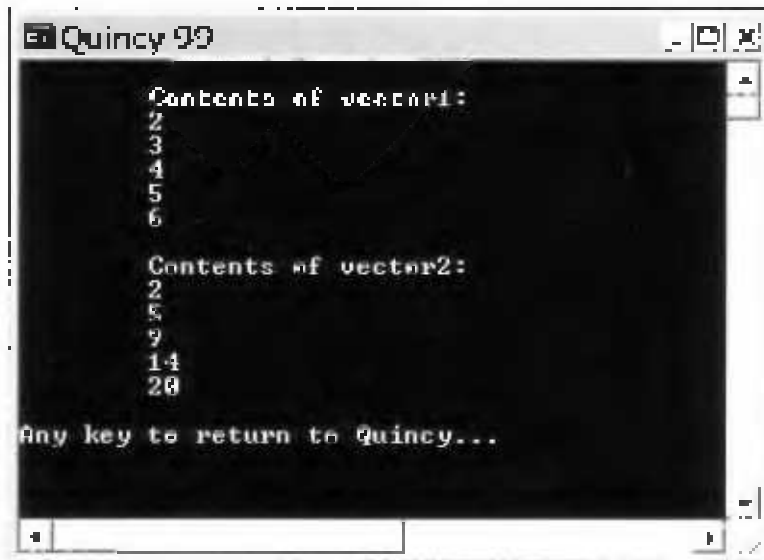
    // Calculate the partial sum.
    partial_sum(mVector1.begin(), mVector1.end(), mVector2.begin(),
```

```

// Display the contents of the resultant vector.
cout << "\nContents of vector2:\n";
for each (intVector2.begin( ), intVector2.end( ), showVal);

return 0;
}

```



ပုံ (၁၆.၁၄)

## ၁၆.၁၄ Using the adjacent\_difference( ) Function

အမှတ်အသားများနှင့် Ex16.15.cpp program ၏ vector object တစ်ခုမှာ {3, 4, 12, 6, 10} ၊ တစ်ခု second vector တစ်ခုကို partial difference( ) algorithm ၏ အသုံးပြုနည်း အရ {3, 4, 3, 12-4, 6-12, 10-6} create လုပ်ဆောင်ထားတဲ့ program ရေးကြည့်ရအောင်။ Ex16.15.cpp program ကို ပုံ (၁၆.၁၅) မှာ ကိုယ် ခြားကိုင်ဆောင် ရေးကြည့်ရအောင်။

```

// Listing 11.15: Using the adjacent_difference( ) function
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>

void showVal (int val)
{
    cout << " " << 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\nContents 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\nContents of vector2:\n";
}

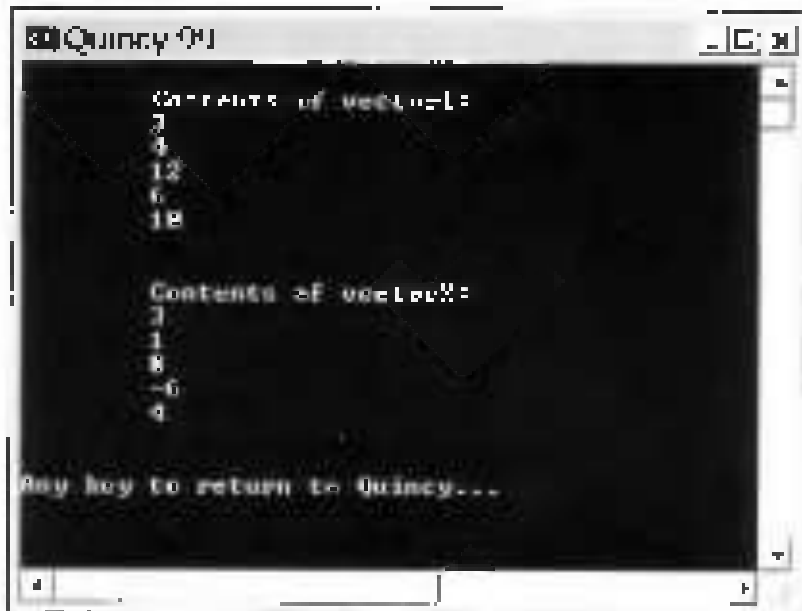
```

```

for each (itVector2.begin(), itVector2.end()), showVal);
cout << endl;

return 0;
}

```



{ 10, -1 }