Module 04: Programming in C++

Sorting and Searching

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

Module Objectives

- Implementation of Sorting and Searching in C and C++

Module Outline

- Sorting in C and C++
  - Bubble Sort
  - Using Standard Library
- Searching in C and C++
  - Using Standard Library
- `algorithm` Library

Program 04.01: Bubble Sort

C Program

```
// FileName:Bubble_Sort.c:
#include <stdio.h>

int main() {
    int data[] = {32, 71, 12, 45, 26};
    int i, step, n = 5, temp;

    for(step = 0; step < n - 1; ++step)
        for(i = 0; i < n-step-1; ++i) {
            if(data[i] > data[i+1]) {
                temp = data[i];
                data[i] = data[i+1];
                data[i+1] = temp;
            }
        }

    for(i = 0; i < n; ++i)
        printf("%d  ", data[i]);

    return 0;
}
```

```
12 26 32 45 71
```

C++ Program

```
// FileName:Bubble_Sort.cpp:
#include <iostream>
using namespace std;
int main() {
    int data[] = {32, 71, 12, 45, 26};
    int n = 5, temp;

    for(int step = 0; step < n - 1; ++step)
        for(int i = 0;i < n-step-1; ++i) {
            if (data[i] > data[i+1]) {
                temp = data[i];
                data[i] = data[i+1];
                data[i+1] = temp;
            }
        }

     for(int i = 0; i < n; ++i)
         cout << data[i] << " ";

     return 0;
}
```

```
12 26 32 45 71
```

- Implementation is same in both C and C++ apart from the changes in basic header files, I/O functions explained in Module 02.

Program 04.02: Using sort from standard library

C Program (Desc order)

```
// FileName:qsort.c:
#include <stdio.h>
#include <stdlib.h>


// compare Function Pointer
int compare(const void *a, const void *b) {
    return (*(int*)a < *(int*)b);
}

int main () {
    int data[] = {32, 71, 12, 45, 26};

    // Start ptr, # elements, size, func. ptr
    qsort(data, 5, sizeof(int), compare);

    for(int i = 0; i < 5; i++)
        printf ("%d ", data[i]);

    return 0;
}
```

```
 71 45 32 26 12
```

- sizeof int, array passed in qsort

C++ Program (Desc order)

```
// FileName:Algorithm_Cust_c++.cpp:
#include <iostream>
#include <algorithm>
using namespace std;

// compare Function Pointer
bool compare (int i, int j) {
    return (i > j);
}

int main() {
    int data[] = {32, 71, 12, 45, 26};

    // Start ptr, end ptr, func. ptr
    sort (data, data+5, compare);

    for (int i = 0; i < 5; i++)
        cout << data[i] << " ";

    return 0;
}
```

```
71 45 32 26 12
```

- Size need not be passed.

Program 04.03: Using default sort of algorithm

C++ Program

```
// FileName:Algorithm_Cust_c++.cpp:
#include <iostream>
#include <algorithm>
using namespace std;

int main () {
    int data[] = {32, 71, 12, 45, 26};

    sort (data, data+5);

    for (int i = 0; i < 5; i++)
        cout << data[i] << " ";

    return 0;
}
```

```
12 26 32 45 71
```

- Sort using the default sort function of algorithm library which does the sorting in ascending order only.

Program 04.04: Binary Search

C Program

```
// FileName:Binary_Search.c:
#include <stdio.h>
#include <stdlib.h>


// compare Function Pointer
int compare (const void * a, const void * b) {
    if ( *(int*)a <  *(int*)b ) return -1;
    if ( *(int*)a == *(int*)b ) return 0;
    if ( *(int*)a >  *(int*)b ) return 1;
}

int main () {
    int data[] = {1, 2, 3, 4, 5};
    int key = 3;

    if (bsearch (&key, data, 5,
            sizeof(int), compare))
        cout << "found!\n";
    else
        cout << "not found.\n";

    return 0;
}
```

```
found!
```

C++ Program

```
// FileName:Binary_Search_c++.cpp:
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
    int data[] = {1, 2, 3, 4, 5};
    int key = 3;

    if (binary_search (data, data+5, key))

        cout << "found!\n";
    else
        cout << "not found.\n";

    return 0;
}
```

```
found!
```

The algorithm Library

The algorithm library of c++ helps us to easily implement commonly used complex functions. We discussed the functions for sort and search. Let us look at some more useful functions.

- Replace element in an array
- Rotates the order of the elements

Program 04.05: replace and rotate functions

Replace

```
// FileName:Replace.cpp:
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    int data[] = {1, 2, 3, 4, 5};

    replace (data, data+5, 3, 2);

    for(int i = 0; i < 5; ++i)
        cout << data[i] << " ";

    return 0;
}
```

```
1 2 2 4 5
```

- 3rd element replaced with 2

Rotate

```
// FileName:Rotate.cpp:
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    int data[] = {1, 2, 3, 4, 5};

    rotate (data, data+2, data+5);

    for(int i = 0; i < 5; ++i)
        cout << data[i] <<" ";

    return 0;
}
```

```
3 4 5 1 2
```

- Array circular shifted around 3rd element.

Module Summary

- Flexibility of defining *customised* sort algorithms to be passed as parameter to sort and search functions defined in the `algorithm` library.
- Predefined optimised versions of these sort and search functions can also be used.
- There are a number of useful functions like rotate, replace, merge, swap, remove etc in `algorithm` library.

Instructor and TAs

Name	Mail	Mobile
Partha Pratim Das, *Instructor*	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, *TA*	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, *TA*	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, *TA*	himadribhuyan@gmail.com	9438911655