C++ BASICS RELEVANT TO CSI 228

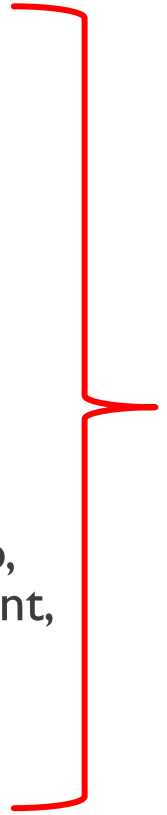
BASICS

- C++ language extends the C programming language with additional features such as type checking, object-oriented programming, exception handling etc.
  - C++ was developed by Bjarne Stroustrup in 1979.
  - File extension .cpp

- Why C++ in this course?
  - The Standard Template Library (STL) of C++ provides useful codes
  - STL is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators.

SIMILARITIES WITH C

- Variables, Operators
- struct
- Array
- Function
- Pointer
- Strings
- If, if…else-if statement, switch case, for loop, while loop, do-while loop, continue statement, break statement, goto statement
- Recursion



can use the same code as written in c

BASICS

C++

```cpp
#include<iostream>
using namespace std;

int main()
{
    cout<<"Hello World!";
    return 0;
}
```

for details explanation: https://beginnersbook.com/2017/08/first-cpp-program/

C

```c
#include<stdio.h>

int main()
{
    printf("Hello World!");
    return 0;
}
```

VARIABLES AND DATA TYPES

- int
- char
- bool
  - holds Boolean value true or false
- double
- float

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  int main()
6  {
7      bool b1 = true;
8      bool b2 = false;
9      if (b1) {
10         printf("inside first if\n");
11         if (b2) {
12             printf("inside first nested if\n");
13         }
14     }
15     printf("end\n");
16     return 0;
17 }
```

Output:
inside first if
end

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int main()
{
    bool b1 = true;
    bool b2 = false;
    bool b3 = 0;
    bool b4 = 1;
    if (b1) {
        printf("inside first if\n");
        if (b2) {
            printf("inside first nested if\n");
        }
        if (b3) {
            printf("inside second nested if\n");
        }
        if (b4) {
            printf("inside third nested if\n");
        }
    }
    printf("end\n");
    return 0;
}
```

Output:

inside first if
inside third nested if
end

scanf(), printf() EQUIVALENT

C++

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a;
    cin >> a;
    cout << a;
    return 0;
}
```

C

```c
#include <stdio.h>
int main()
{
    int a;
    scanf("%d",&a);
    printf("%d",a);
    return 0;
}
```

scanf(), printf() EQUIVALENT

which one is the C++ code?

```cpp
#include <iostream>
using namespace std;
int main()
{
    char str[100];
    /* input single word */
    cin >> str;
    cout << str;
    /* discards the input buffer */
    cin.sync();
    /* input a line */
    cin.get(str, 100);
    cout << str;
    return 0;
}
```

```cpp
#include <stdio.h>
int main()
{
    char str[100];
    /* input single word */
    scanf("%s", str);
    printf("%s", str);
    /* discards the input buffer */
    fflush(stdin);
    /* input a line */
    fgets(str, 100, stdin);
    printf("%s", str);
    return 0;
}
```

IF YOU PREFER scanf(), printf() OVER cin, cout

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main()
5  {
6      int a;
7      scanf("%d",&a);
8      printf("%d",a);
9      return 0;
10 }
```

STANDARD TEMPLATE LIBRARY (STL)

- STL is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators.

- We are going to use
  - Sorting
  - vector
  - priority queue

VECTOR

- #include <vector>
- To know more about vectors: https://www.edureka.co/blog/vectors-in-cpp/

Vector

```cpp
#include <iostream>
#include <cstdio>
#include <vector>
using namespace std;
int main()
{
    /* initialization - way 1 */
    vector<int> list2 = {1, 10, 200};
    /* initialization - way 2 */
    vector<int> list4;
    int x;
    for (int i = 0; i < 5; i++)
    {
        cin >> x;
        list4.push_back(x);
    }
    /* size of vector */
    printf("size: %d\n", list4.size());
    /* iterate over vector - way 1 */
    for (int i = 0; i < list4.size(); i++)
        printf("[%d] %d\n", i, list4[i]);
    /* delete from index i */
    int i = 2;
    list4.erase(list4.begin() + i);
    /* iterate over vector - way 2 */
    for (int x : list4)
        printf("%d\n", x);
}
```

Input:

```
11 753 2 8 91
```

Output:

```
size: 5
[0] 11
[1] 753
[2] 2
[3] 8
[4] 91
11
753
8
91
```

Vector

```cpp
#include <bits/stdc++.h>


using namespace std;
int main()
{
    /* initialization - way 1 */
    vector<int> list2 = {1, 10, 200};
    /* initialization - way 2 */
    vector<int> list4;
    int x;
    for (int i = 0; i < 5; i++)
    {
        cin >> x;
        list4.push_back(x);
    }
    /* size of vector */
    printf("size: %d\n", list4.size());
    /* iterate over vector - way 1 */
    for (int i = 0; i < list4.size(); i++)
        printf("[%d] %d\n", i, list4[i]);
    /* delete from index i */
    int i = 2;
    list4.erase(list4.begin() + i);
    /* iterate over vector - way 2 */
    for (int x : list4)
        printf("%d\n", x);
}
```

Input:

```
11 753 2 8 91
```

Output:

```
size: 5
[0] 11
[1] 753
[2] 2
[3] 8
[4] 91
11
753
8
91
```

SORTING

- sort array
- sort vector
- sort structure

SORT Array

```
1  #include <iostream>
2  #include <cstdio>
3  #include <bits/stdc++.h>
4  using namespace std;
5  int main()
6  {
7      int arr[] = {100, 512, 6, 724, 31, 14, 2, 0};
8      /* Length of the array */
9      int len = sizeof(arr) / sizeof(arr[0]);
10     /* print the array */
11     for (int i = 0; i < len; i++)
12         printf("%d ", arr[i]);
13     printf("\n");
14     /* sort the array */
15     sort(arr, arr + len);
16     /* print the array */
17     for (int i = 0; i < len; i++)
18         printf("%d ", arr[i]);
19     printf("\n");
20
21     return 0;
22 }
```

Output:

```
100 512 6 724 31 14 2 0
0 2 6 14 31 100 512 724
```

Default order/
Ascending order

SORT Array

```cpp
#include <iostream>
#include <cstdio>
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int arr[] = {100, 512, 6, 724, 31, 14, 2, 0};
    /* Length of the array */
    int len = sizeof(arr) / sizeof(arr[0]);
    /* print the array */
    for (int i = 0; i < len; i++)
        printf("%d ", arr[i]);
    printf("\n");
    /* sort the array */
    sort(arr, arr + len);
    /* print the array */
    for (int i = 0; i < len; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

Output:

```
100 512 6 724 31 14 2 0
0 2 6 14 31 100 512 724
```

Default order/
Ascending order

```cpp
#include <iostream>
#include <cstdio>
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int arr[] = {100, 512, 6, 724, 31, 14, 2, 0};
    /* Length of the array */
    int len = sizeof(arr) / sizeof(arr[0]);
    /* print the array */
    for (int i = 0; i < len; i++)
        printf("%d ", arr[i]);
    printf("\n");
    /* sort the array */
    sort(arr, arr + len, greater<int>());
    /* print the array */
    for (int i = 0; i < len; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

Output:

```
100 512 6 724 31 14 2 0
724 512 100 31 14 6 2 0
```

Descending order

SORT Array of struct

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Pair
{
    int a, b;
};
bool comp(Pair p1, Pair p2)
{
    return p1.b < p2.b;
}
int main()
{
    /* an array of struct */
    Pair arr[] = {{5, 100}, {3, 9}, {3, 12}, {1, 6}, {5, 5}, {8, 16}};
    int n = sizeof(arr) / sizeof(arr[0]);
    /* sort the array */
    sort(arr, arr + n, comp);
    /* print the array */
    for (int i = 0; i < n; i++)
    {
        printf("a:%d b:%d\n",arr[i].a, arr[i].b);
    }

    return 0;
}
```

Output:

```
a:5 b:5
a:1 b:6
a:3 b:9
a:3 b:12
a:8 b:16
a:5 b:100
```

this function is a must for sorting an array of struct

No default order. Order must be specified by a function

SORT Array of struct

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Pair
{
    int a, b;
};
bool comp2(Pair p1, Pair p2)
{
    return p1.b / p1.a > p2.b / p2.a;
}
int main()
{
    /* an array of struct */
    Pair arr[] = {{5, 100}, {3, 9}, {3, 12}, {1, 6}, {5, 5}, {8, 16}};
    int n = sizeof(arr) / sizeof(arr[0]);
    /* sort the array */
    sort(arr, arr + n, comp2);
    /* print the array */
    for (int i = 0; i < n; i++)
    {
        printf("a:%d b:%d ratio:%d\n",arr[i].a, arr[i].b, arr[i].b/arr[i].a);
    }

    return 0;
}
```

Output:

```
a:5 b:100 ratio:20
a:1 b:6 ratio:6
a:3 b:12 ratio:4
a:3 b:9 ratio:3
a:8 b:16 ratio:2
a:5 b:5 ratio:1
```

No default order. Order must be specified by a function

STRING

- In C++, string is an object of **std::string** class
- It stores a sequence of characters.
  - e.g. “hello world”

```cpp
#include <bits/stdc++.h>
using namespace std;

int main( ) {
    /// init1
    string str1 = "Hello world";
    cout<<str1<<endl;
    /// init2
    char chr_str[] = "hello universe";
    string str2 = string(chr_str);
    cout<<str2<<endl;

    /// length of a string
    cout << str1.length() << endl;

    /// accessing individual characters
    cout << "str1[0]:" << str1[0] << " str1[6]:" << str1[6] << endl;

    /// appending to a string
    str1 += ".";
    cout<<str1<<endl;
    str2 += ". ";
    str2 += str1;
    cout<<str2<<endl;

    /// input a string
    string word, line;
    cin >> word;
    cin.sync();
    cout << word << endl;
    getline(cin, line);
    cout << line << endl;
}
```

Input:

cpp strings

cpp strings

Output:

```
Hello world
hello universe
11
str1[0]:H str1[6]:w
Hello world.
hello universe. Hello world.
cpp
cpp strings
```

```cpp
#include <iostream>
using namespace std;
int main( ) {
    /// passing, returning, assigning strings
    string str1 = "hello";
    string str2 = str1; // makes a new copy
    str1[0] = 'y'; // changes str1, but not str2
    cout << str1 << " " << str2 << endl;

    /// comparing two strings
    if (str2 == str1)
        cout << "str1 and str2 both changed!!!";
    if (str2=="hello")
        cout << "str2 did not change!!!";

    /// You can use <, <=, >, and >= to compare strings as well.
    /// These operators compare strings lexicographically,
    ///     character by character and are case-sensitive.
}
```

Output:

```
yello hello
str2 did not change!!!
```

STRING

- Searching within a string.
  - search a character key: **`str.find(key)`**
    - returns the starting position of key
    - returns the constant **`string::npos`** if not found
  - search a string key: **`str.find(key)`**

```cpp
#include <iostream>
using namespace std;

int main() {
    string str = "Yes, we went to Gates after we left the dorm.";
    int first_we = str.find("we"); // finds the first "we"
    int sec_we = str.find("we", first_we + 1); // finds "we" in "went"
    int third_we = str.find("we", sec_we + 1); // finds the last "we"
    int gPos = str.find('G');
    int zPos = str.find('Z'); // returns string::npos
    cout << "First we: " << first_we << endl;
    cout << "Second we: " << sec_we << endl;
    cout << "Third we: " << third_we << endl;
    cout << "Is G there? ";
    cout << (gPos != string::npos ? "Yes!" : "No!") << endl;
    cout << "Is Z there? ";
    cout << (zPos != string::npos ? "Yes!" : "No!") << endl;
    return 0;
}
```

Output:

```
First we: 5
Second we: 8
Third we: 28
Is G there? Yes!
Is Z there? No!
```

STRING

- Extracting substrings
  - `str.substring(“hello”);`

```cpp
#include <iostream>
using namespace std;
int main() {
    string old_s = "Thank you very very much";
    cout << old_s << endl;
    int found = old_s.find("very");
    string new_s = old_s.substr(0, found);
    cout << new_s << endl;
    new_s += old_s.substr(found + 5);
    cout << new_s << endl;
    return 0;
}
```

Output:

```
Thank you very very much
Thank you
Thank you very much
```

PRIORITY QUEUE

- A priority queue in c++ is a type of container adapter, which processes only the highest priority element, i.e. the first element will be the maximum of all elements in the queue, and elements are in decreasing order.
  - `priority_queue<int> variableName;`
- `min queue`
  - `priority_queue <int, vector<int>, greater<int>> q;`

PRIORITY QUEUE

- `p.empty()`
- `p.size()`
- `p.push(10) // insert 10`
  - firstly, the element is added to the end of the queue, and simultaneously elements reorder themselves with priority. It takes value in the parameter.
- `p.pop()`
  - deletes the top element (highest priority) from the priority_queue
- p. top()
  - returns the top element (highest priority) from the priority_queue

```cpp
#include<bits/stdc++.h>
using namespace std;

struct Pair {
    int a, b;
};
struct comp_a {
    bool operator()(Pair const& p1, Pair const& p2)
    {
        return p1.a < p2.a;
    }
};
int main()
{
    priority_queue<Pair, vector<Pair>, comp_a> Q;

    Pair arr[] = {
        { 0, 5 },
        { 5, 5 },
        { 20, 6 },
        { 3, 6 },
        { 23, 5 }
    };
    for (int i = 0; i < 5; ++i) {
        Q.push(arr[i]);
    }
    while (!Q.empty()) {
        Pair p = Q.top();
        Q.pop();
        cout << p.a << " " << p.b << "\n";
    }
    return 0;
}
```

Output:

23 5
20 6
5 5
3 6
0 5

REFERENCES

- https://beginnersbook.com/2017/08/c-plus-plus-tutorial-for-beginners/
- https://www.edureka.co/blog/vectors-in-cpp/
- https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1132/handouts/08-C++-Strings.pdf
- https://www.mygreatlearning.com/blog/priority-queue-in-cpp/
- https://www.geeksforgeeks.org/stl-priority-queue-for-structure-or-class/