

UoA CP Lecture 1

Shifting to C++(or C with STL)

Qingchuan(Sam) Zhang

April 3, 2020

Contents

1	Background	3
2	Basic Framework	4
3	Basic Data Types	5
4	Input/Output	5
4.1	C++ style(recommended)	5
4.2	C style(not recommended)	5
5	if/for/while	6
5.1	if	6
5.2	for	6
5.3	while	6
6	Array	8
6.1	std::vector<Type>	8
6.2	Type a[]	8
7	Example Problem: Increasing Array	9
8	String	10
8.1	std::string	10
8.2	char s[]	10
9	Struct keyword	11
9.1	Definition	11
9.2	Example: Codeforces 479 C	12
10	Other Containers	13
10.1	std::queue	13

10.2	<code>std::priority_queue</code>	13
10.3	<code>std::stack</code>	13
10.4	<code>std::set<Type></code>	13
10.5	<code>std::map<Key,Value></code>	13
11	Function	13
12	Useful Library Functions	13
12.1	<code>std::max/min</code>	13
12.2	<code>std::sort</code>	13
12.3	<code>std::lower_bound/upper_bound</code>	13
12.4	<code>std::kth_element</code>	13

1 Background

In competitive programming, no expertise in C++/Java/Python is required. We only need to a small portion of the language(s) to convert our ideas to codes. Here I'll introduce the basic stuff needed for CP. Though there are more advanced stuff might be useful, the ones here can fulfill our needs for now.

The documentation of C++ is on [cppreference](#). Use the docs to find things you don't know how to use.

Let's start!

2 Basic Framework

```
1  #include <iostream>
2  #include <string>
3  #include <algorithm>
4  #include <vector>
5  #include <queue>
6  // or #include<bits/stdc++.h> if supported
7  using namespace std;
8
9  int main() {
10
11      return 0;
12 }
```

#include This is like `import` in Python, you need to include the libraries before using built-in functionalities.

using namespace std This line means you want to use the functions/containers residing in the `std` namespace as if they were written by yourself. If you don't put this line, you will need to add `std::` before the names. For example, `sort` versus `std::sort`

main This is where your program starts to run. The return type should always be `int` and it always return 0.

3 Basic Data Types

- `bool`: either `true` or `false`.
- `char`: an ASCII character, usually used to store `'a'-'z'` or `'A'-'Z'` or `'0'-'9'`.
- `int`: integers range from -2^{31} to $2^{31} - 1$, or just memorize $\pm 10^9$
- `long long`: integers range from -2^{63} to $2^{63} - 1$, or just memorize $\pm 10^{18}$
- `double`: decimal numbers of 15 digit precision.
- `std::string`: stores a sequence of characters like `"abcdefg"` or `"from1to2"`

4 Input/Output

4.1 C++ style(recommended)

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int a;
5      long long b;
6      double c;
7      cin >> a >> b >> c;
8      cout << a << " " << b << " " << c << endl; //endl = end of line
9      return 0;
10 }
```

4.2 C style(not recommended)

```
1  #include<stdio>
2  int main(){
3      int a;
4      long long b;
5      double c;
6      scanf("%d %lld %lf", &a, &b, &c);
7      printf("%d %lld %f", a, b, c);
8      return 0;
9  }
```

5 if/for/while

The following three structures all the most common ones used by almost every single c++ program.

5.1 if

```
1  //...
2  int main(){
3      int value;
4      cin>>value;
5      if(value == 0){
6          cout<<"the input is zero"<<endl;
7      }else{
8          cout<<"the input is not zero"<<endl;
9      }
10     return 0;
11 }
```

5.2 for

```
1  //...
2  int main(){
3      int n;
4      cin>>n;
5      int sum = 0;
6      for(int i = 1;i<=n;i++){
7          sum = sum + n;
8          //sum += n;
9      }
10     cout<<"The sum from"<<1<<" to n is "<<sum<<endl;
11     return 0;
12 }
```

5.3 while

```
1  //...
2  int main(){
3      int value;
4      cin>>value;
5      int step = 0;
6      //collatz conjecture
7      while(value != 1){
8          if(value % 2 == 0){
9              value /= 2;
10             }else{
```

```
11         value = value * 3 + 1;
12     }
13     step++;
14     //step += 1;
15 }
16 cout<<"The number of steps is "<<step<<endl;
17 return 0;
18 }
```

6 Array

6.1 `std::vector<Type>`

```
1  //..
2  #include<vector>
3  int main(){
4      vector<int> a(5,1); //{1,1,1,1,1}
5      a.insert(a.begin()+1,2); //{1,2,1,1,1}
6      //0-indexed
7      for(int i = 1; i<a.size(); i++){
8          a[i] = a[i-1];
9      }
10     int last = a.back();
11     bool is_empty = a.empty();
12
13
14     return 0;
15 }
```

6.2 Type `a[]`

```
1  //...
2
3  int a[100]; //{0,0,0,...}
4  const int maxn = 200;
5  int b[maxn];
6  int n;
7  int c[n]; //error
8  int main(){
9      int d[100]; // random values
10     int value = d[50];
11     cout<<value<<endl;
12     return 0;
13 }
```


7 Example Problem: Increasing Array

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXN=200000+5;
4  int n;
5  int nums[MAXN];
6
7  int main()
8  {
9      cin>>n;
10     for(int i=0; i<n; i++)
11         cin>>nums[i];
12     long long ans=0;
13     for(int i=1; i<n; i++)
14     {
15         if(nums[i]>=nums[i-1])
16             continue;
17         else
18         {
19             ans+=nums[i-1]-nums[i];
20             nums[i]=nums[i-1];
21         }
22     }
23     cout<<ans<<endl;
24     return 0;
25 }
```

8 String

8.1 std::string

```
1  //..  
2  #include<string>  
3  int main(){  
4      string a = "abcdefg"; // set to literal value  
5      string b; // set to ""  
6      string c;  
7      cin>>c; // input  
8      b = "xyz";  
9      c = c + a + b; // concatenation  
10     cout<<c<<" "<<c[0]<<endl;  
11     return 0;  
12 }
```

8.2 char s[]

```
1  //...  
2  int main(){  
3      char s[5] = "abcd"; // "abcd\0"  
4      char a[10];  
5      scanf("%s",a);  
6      printf("%s",a);  
7      return 0;  
8  }
```

9 Struct keyword

9.1 Definition

```
1  struct Point{
2      int x,y;
3      Point(int xx,int yy){
4          x = xx;
5          y = yy;
6      }
7      double disFromOrigin(){
8          return sqrt(x*x + y*y);
9      }
10 };
11
12 int main(){
13     Point p(10,20);
14     cout<< p.disFromOrigin() << endl;
15     return 0;
16 }
```

9.2 Example: Codeforces 479 C

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int MAXN = 5000 + 10;
5  int n;
6  struct exam_time
7  {
8      int atime, btime;
9      exam_time(){};
10     exam_time(int a, int b)
11     {
12         this->atime = a;
13         this->btime = b;
14     }
15     bool operator<(const exam_time &exam)
16     {
17         return atime < exam.atime || (atime == exam.atime && btime < exam.btime);
18     }
19 } exam[MAXN];
20 int main()
21 {
22     cin >> n;
23     for (int i = 0; i < n; i++)
24     {
25         int x, y;
26         cin >> x >> y;
27         exam[i] = exam_time(x, y);
28     }
29     sort(exam, exam + n);
30     int cur = 0;
31     for (int i = 0; i < n; i++)
32     {
33         if (exam[i].btime < cur)
34         {
35             cur = exam[i].atime;
36         }
37         else
38         {
39             cur = exam[i].btime;
40         }
41     }
42     cout << cur << endl;
43     return 0;
44 }
```

10 Other Containers

10.1 `std::queue`

10.2 `std::priority_queue`

10.3 `std::stack`

10.4 `std::set<Type>`

10.5 `std::map<Key,Value>`

11 Function

12 Useful Library Functions

12.1 `std::max/min`

12.2 `std::sort`

12.3 `std::lower_bound/upper_bound`

12.4 `std::kth_element`