

# Personal Competitive Programming Notebook

Juan David Gaviria Correa

June 4, 2022

## Contents

<b>1</b>	<b>C++</b>	<b>2</b>
1.1	C++ template . . . . .	2
<b>2</b>	<b>Type Conversion</b>	<b>2</b>
2.1	string to number . . . . .	2
2.2	number to string . . . . .	2
2.3	int to char . . . . .	2
2.4	char to int . . . . .	2
<b>3</b>	<b>Chars</b>	<b>2</b>
3.1	Change Case . . . . .	2
3.2	Check Case . . . . .	2
<b>4</b>	<b>Strings</b>	<b>2</b>
4.1	Substring . . . . .	2
4.2	Replace . . . . .	2
4.3	Replace All Matches . . . . .	3
4.4	Change Case . . . . .	3
4.5	Trim . . . . .	3
4.6	Split . . . . .	3
4.7	Compare Lexicographically . . . . .	3
<b>5</b>	<b>Useful Stuff</b>	<b>4</b>
5.1	Swap values . . . . .	4
5.2	Read vector . . . . .	4
5.3	Print vector . . . . .	4
5.4	Sort vector . . . . .	4

# 1 C++

## 1.1 C++ template

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

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    cout << setprecision(20) << fixed;
    return 0;
}
```

## 2 Type Conversion

### 2.1 string to number

```
// stof - float
// stod - double
// stold - long double

// stoi - int
// stol - long
// stoul - unsigned long
// stoll - long long
// stoull - unsigned long long

int x = stoi("789.19");
// Output: 789
```

### 2.2 number to string

```
string x = to_string(475.1);
// Output: 475.100000
```

### 2.3 int to char

```
// Any int
char x = 97;
// Output: a

// The int is a number from 0 to 9 and want to obtain the
// same number
char x = 5 + '0';
// Output: 5
```

### 2.4 char to int

```
// Any char
char y = 'a';
int x = y;
// Output: 97

// The char is a number and want the same number as int
char y = '5';
int x = y - '0';
// Output: 5
```

## 3 Chars

### 3.1 Change Case

```
char letter = tolower('A');
// Output: a

char letter = toupper('a');
// Output: A
```

### 3.2 Check Case

```
islower('a');
// Output: true

isupper('A');
// Output: true
```

## 4 Strings

### 4.1 Substring

```
string text = "Apple, Banana, Kiwi";
// Second param is optional, default text.length
text.substr(7, 6);
// Output: Banana
```

### 4.2 Replace

```
bool replace(string &str, const string &from, const
             string &to)
{
    size_t start_pos = str.find(from);
    if (start_pos == string::npos)
```

```

        return false;
    str.replace(start_pos, from.length(), to);
    return true;
}

string text = "Apple, Banana, Apple";
replace(text, "Apple", "Banana");
// Output: Banana, Banana, Apple

```

### 4.3 Replace All Matches

```

void replaceAll(string &str, const string &from, const
string &to)
{
    if (from.empty())
        return;
    size_t start_pos = 0;
    while ((start_pos = str.find(from, start_pos)) !=
string::npos)
    {
        str.replace(start_pos, from.length(), to);
        start_pos += to.length();
    }
}

string text = "Apple, Banana, Apple";
replaceAll(text, "Apple", "Banana");
// Output: Banana, Banana, Banana

```

### 4.4 Change Case

```

string text = "ApPlE";
// To lower case
transform(text.begin(), text.end(), text.begin(), ::
tolower);
// Output: apple
// To upper case
transform(text.begin(), text.end(), text.begin(), ::
toupper);
// Output: APPLE
// Capitalize
transform(text.begin(), text.end(), text.begin(), ::
tolower);
str[0] = toupper(str[0]);
// Output: Apple

```

### 4.5 Trim

```

void ltrim(string &s)
{
    s.erase(s.begin(), find_if(s.begin(), s.end(),
!::isspace));
}

void rtrim(string &s)
{
    s.erase(find_if(s.rbegin(), s.rend(), !::isspace)
.base(), s.end());
}

string text = " ' Apple ' ";
ltrim(text);
rtrim(text);
// Output: ' Apple '

```

### 4.6 Split

```

vector<string> split(string str, char del)
{
    vector<string> tokens;
    string token;
    stringstream ss(str);

    while (getline(ss, token, del))
    {
        tokens.push_back(token);
    }

    return tokens;
}

vector<string> tokens = split("Apple Banana Apple", ' ');
copy(tokens.begin(), tokens.end(), ostream_iterator<
string>(cout, ", "));
// Output: Apple,Banana,Apple,

```

### 4.7 Compare Lexicographically

```

string a = "abcd";
string b = "bcde";

if (a < b)
{
    cout << "a is first";
}
else if (a < b)
{
    cout << "b is first";
}
else
{
    cout << "same";
}

```

```
}  
// Output: a is first  
// Note: You can order lexicographically a vector with  
sort
```

---

## 5 Useful Stuff

### 5.1 Swap values

```
int a = 1, b = 2;  
swap(a, b);  
// A will be 2 and b will be 1
```

---

### 5.2 Read vector

```
copy_n(istream_iterator<type>(cin), times, back_inserter(  
arr));
```

---

### 5.3 Print vector

```
copy(arr.begin(), arr.end(), ostream_iterator<type>(cout,  
" "));
```

---

### 5.4 Sort vector

```
// This works with any data type  
vector<int> nums = {2, 3, 1, 4};  
// Ascending  
sort(nums.begin(), nums.end());  
// nums = 1, 2, 3, 4  
// Descending  
sort(nums.begin(), nums.end(), ::greater<int>());  
// nums = 4, 3, 2, 1
```

---