Data Structures & Programming

A Reminder on C++

Golnar Sheikhshab

Saying hello to the world!

```
#include <iostream>

using namespace std;

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

Compile & Run

compiling:

```
g++ -o hello_world.o hello_world.cpp
```

running:

```
./hello_world.o
```

output:

```
Hello World!
```

Greeting somebody

```
#include <iostream>

using namespace std;

int main(){
	cout << "Hello. What is your name?" << endl;
	string name;
	cin >> name;
	cout << "Nice to meet you " << name << endl;
	return 0;
}
```

Greeting somebody (2)

```
#include <iostream>

using namespace std;

int main(int argc, char** argv){
	if (argc != 2){
		cout << "Usage: executable.o name\n";
		return 1;
	}
	string name = argv[1];
	cout << "Hello " << name << ". Nice to meet you." << endl;
	return 0;
}
```

Compile & Run

compiling:

```
g++ -o greetings2.o greetings2.cpp
```

running:

```
./greetings2.o
```

output:

```
Usage: executable.o  name
```

running:

```
./greetings2.o  Golnar
```

output:

```
Hello Golnar. Nice to meet you.
```

Counting

```
#include <iostream>

using namespace std;

int main(){
	int n = 10;
	for (int i = 0; i < n; i++)
		cout << i;
	return 0;
}
```

Counting (correction)

```
#include <iostream>

using namespace std;

int main(){
	int n = 10;
	for (int i = 0; i < n; i++)
		cout << i << " " ;  // without space the output will be 0123456789
	cout << endl;
	return 0;
}
```

Playing With the User

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main(){
	int iSecret = rand() % 10 + 1, guess;
	cout << "Guess what number I'm thinking..." << endl;
	cin >> guess;
	while( guess != iSecret ) {
		cout << "Nope! Guess again." << endl;
		cin >> guess;
	}
	cout << "Wow! You have superpowers! \n";
	return 0;
}
```

Does it work?

Copy and paste the code to a cpp file

Compile and run it

Does it work?

Are you sure?

What's wrong?

How to solve it?

rand() and srand()

rand() generates pseudo random numbers

it will use the same seed every time unless otherwise specified

seed can be changed using srand()

add the following line before the first time of using rand and see what happens (Also include time.h)

```
srand (time(NULL));
```

Is the program perfect now?

- What happens if the user enters something that's not an integer?
  - It loops forever because the bad input stays there in a buffer and it fails to parse the input again and again, resulting in a 0 again and again. To avoid a second failure you can add

    ```
    cin.clear();
    cin.ignore(10000, '\n');
    ```

  - but nevermind that.

- Let's try to change the code so we handle non-integer input correctly.

```
int getGuess(string message){
	int guess;
	cout << message << endl;
	string strGuess;
	cin >> strGuess;
	try {
		guess = boost::lexical_cast<int>( strGuess );
	}
	catch( boost::bad_lexical_cast const& ) {
		guess = getGuess("Input not acceptable. Please enter an integer.");
	}
	return guess;
}
```

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <boost/lexical_cast.hpp>
using namespace std;

int main(){
	srand (time(NULL));
	int iSecret = rand() % 10 + 1, guess;
	guess = getGuess("Guess what number I'm thinking...");
	while( guess != iSecret )
		guess = getGuess("Nope! Guess again.");
	cout << "Wow! You have superpowers! \n";
	return 0;
}
```

Sorting

An inefficient sorting algorithm: insertion sort

Idea: Read items one at a time and insert each item in its rightful place thus far

Illustration: Input : 64 90 12 51 3

Step	Sorted_until_now	Item Read	Sorted_until_after_item_read
1	[]	64	[64]
2	[64]	90	[64 90]
3	[64 90]	12	[12 64 90]
4	[12 64 90]	51	[12 51 64 90]
5	[12 51 64 90]	3	[3 12 51 64 90]

Sorting Program

- Let's keep the sorted list in an array

```
const int max_num_items = 100;
int sorted[max_num_items];
int n=0;
```

- Read positive integers from standard input and update the array

```
int x=-100;
while ((n < max_num_items) && (x != -1)){
        cin >> x;
        if (x > 0 )
                update(sorted, n, x);
        cin.clear();
        cin.ignore(1000, '\n');
}
```

Sorting Program - Updating the Array

```
void update(int* sorted_array, int & n, int x )
{
	// finding the rightful place of x
	int x_index = right_place(sorted_array, n, x);

	n++;

	// shift everything starting from x_index to the right to open a space for x
	for (int i= n-2; i>= x_index; i--)
		sorted_array[i+1] = sorted_array[i];

	sorted_array[x_index] = x;
}
```

Sorting Program - Finding the Rightful Place of x

```
int right_place(int* sorted_array, int & n, int x){
        int x_index = n;
        for (int i = 0; i<n; i++){
                if (sorted_array[i] >= x){
                        x_index = i;
                        break;
                }
        }
        return x_index;
}
```

What if we want a generic sorting function?

A function that can work with different types? use c++ templates

```
template<typename Type>
int right_place(Type* sorted_array, int & n, Type x){
        int x_index = n;
        for (int i = 0; i<n; i++){
                if (sorted_array[i] >= x){
                        x_index = i;
                        break;
                }
        }
        return x_index;
}
```

Sorting program - generic update

```
template<typename Type>
void update(Type* sorted_array, int & n, Type x )
{
	// finding the rightful place of x
	int x_index = right_place(sorted_array, n, x);

	n++;

	// shift everything starting from x_index to the right to open a space for x
	for (int i= n-2; i>= x_index; i--)
		sorted_array[i+1] = sorted_array[i];

	sorted_array[x_index] = x;
}
```

Sorting program - generic - works with strings

```
void sort_string(){
        string sorted[max_num_items];
        int n=0;

        string x = "<start>";
        while ((n < max_num_items) && (x != "<stop>")){
                cin >> x;
                update(sorted, n, x);
        }
}
```

Sorting program - generic - works with integers

```
void sort_positive_int(){
        int sorted[max_num_items];
        int n=0;
        int x=-100;
        while ((n < max_num_items) && (x != -1)){
                cin >> x;
                if (x > 0 ){
                        update(sorted, n, x);
                }
                cin.clear();
                cin.ignore(1000, '\n');
        }
}
```