0.1 Let's get started

0.1.1 Purpose of the class

Welcome to the data structure class. During this class, you will learn the most used structures is computer science, as well as C++. This class will prove to be very useful to you for the following reasons:

1. Data structures are a must-know for every computer scientist and programmer.
2. 90% of interview questions for software engineering position are about data structures.
3. Will significantly improve your programming skill, and take your employability to the next level.
4. Give you insight on how things happen when you execute a program.

By the end of this class, you will have a thorough understanding of the most useful data structures. For every data structure that I will cover in this class, you will learn how to use them, how to implement them, and when to use them.

0.1.2 Structure

For most of the classes, you will be given a programming assignment, which I will refer to as **Machine Problem(MP)**. MPs are a wonderful way for you to prove your understanding of data structures. You will be asked to implement the structures, and use them in some significant way. I expect you to have some programming experience prior to this class. We will be moving fast, so it is important that you have some prior experience.
The MPs will be in C++ for the following reasons:

- One of the most influential programming language of all time. A must know for every computer scientist.
- Allows memory management. Very important when dealing with data structures.
- Lower level than dynamic languages. You will be able to understand many more things about how a computer program actually works.

You are not required to know C++ before this class, although it will help. During the first two lectures, I will be covering some of the most important aspects of C++. So, lets dive straight into C++, and lets take a look at some code.

0.1.3 Intro to C++

C++ was developed on top of C, in order to add Object Oriented features to C. C++ also comes with a useful collections of data structures and functions, which is called **Standard Template Library (STD)**. Lets dive our first C++ program:

In order to compile this code, type in your terminal:

```
g++ -o hello helloWorld.cpp
```

and then run the executable **hello** by

```
./hello
```

Congratulations! You made your first C++ program. Lets now move to a more interesting example. Here is an implementation of a Fibonacci function:

```
1  int fibonacci(int n) {
2    if ( n == 0 || n == 1 )
3        return n;
4
5    int fib1 = 0;
6    int fib2 = 1;
7    int fib;
8
9    for ( int i = 2; i < n; i++ )
10   {
11       fib = fib1 + fib2;
12       fib1 = fib2;
13       fib2 = fib;
14   }
15   return fib;
16 }
```

Now, lets analyze this code:

1. Every **expression** in C++ needs to end with a semicolon.
2. Blocks of code need to be wrapped with curly braces.
3. On line *2*, the || operator means **logic or**.
4. C++ is a static typed language. That is, you have to declare a variable type before you use it. Notice, that the **fibonacci** function returns a value of type int.

Likewise, all the variables defined have a associated type. C++ has the following primitive types:

Type name	Translation	Values	Dimension in bit(s)
bool	Boolean	true, false	1
char	Character	$[-2^4 \dots 2^4]$	8
short	Integer	$[-2^8 \dots 2^8]$	16
int	Integer	$[-2^{16} \dots 2^{16}]$	32
float	Floating Point		16
double	Floating Point		32
void	Null		

Converting from one type to another is called **casting**.

5. The loop on line *9* can be decomposed as:

```
1    int i = 2;
2    while ( i < n )
3    {
4      /* do stuff */
5      i++;
6    }
```

where $i++$ is equivalent to $i = i + 1$ or $i+ = 1$.

You can also have multiple variable loops, such as

```
1 for ( int i =0, int j=3; i < n, j++, i ++ )
```

For every iteration, both j and i will increase. Here is a table of operators for C++:

Operator	Meaning	Example
++	Increase by one	$i++;$
−−	Decrease by one	$i--;$
&&	Logic and	i && $false$ will always return $false$
||	Logic or	i || $true$ will always return $true$
!	Logic negation	!0 == 1 will always return $true$

Also, we have the basic math operators, such as $+, -, *, /$.
In C++ there are two different types of arrays. The first type, are arrays that cannot change their length, and are called static arrays. The second type can change their length, and are called dynamic.
Here is an example on how to use a static array:

```cpp
int array [5]; // creates array of 5. not initialized
int five  [5] = { 0, 1, 2, 3, 4 }; // you can initialize arrays this way
//int five [] = { 0, 1, 2, 3, 4 }; this is also valid

for( int i=0; i < 5; i++ )
{
  cout << five[i] << "  "; // will print 0 1 2 3 4
  array[i] = i; // initialize array
}
return 0;
```

0.1.4 First C++ assignment

Lets get some practice. Open with your favorite editor the file 'snippets.cpp'. You will find several functions that need to be completed. You may notice that I have already created test cases, so that you can verify correctness of your code. Use the command:

```
g++ -o snippets snippets.cpp
```

and then run the executable **snippets** by

```
./snippets
```

Good luck!