ROYAL INSTITUTE
OF TECHNOLOGY

Introduction to C++ Programming

Vahid Kazemi

ROYAL INSTITUTE OF TECHNOLOGY

Overview

- An overview of C/C++
  - Basic types, Pointers, Arrays, Program control, Functions, Arguments, Structures, Operator overloading, Namespaces, Classes, and Templates
- Standard Template Library (STL)
  - string, vector, list

ROYAL INSTITUTE OF TECHNOLOGY

Hello World

```c
#include <stdio.h>

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

Compilation

- Compilation of a C++ program is consisted of three main steps:

  - **Preprocessing**: The preprocessor translates all the directives in your code, and replace them with C++ code.

  - **Compiling**: The output of last step will be compiled to machine code. The result will be stored in a set of object files.

  - **Linking**: All the dependencies will be resolved. Individual object files will be combined with the imported libraries to create an executable program.

ROYAL INSTITUTE OF TECHNOLOGY

Variables

- C++ is a statically typed language. It means that you have to specify the type of variable when you define it, and you can't change the type of a variable in C++

```
int age = 22;
```

- However you can cast a variable temporarily to a compatible type:

```
float time = 10.5;
int hour = (int)time ;
```

ROYAL INSTITUTE OF TECHNOLOGY

Scope

- Scope determines the lifetime of variables in C++
- Every function has its own scope
- You can create a local scope with curly brackets {}

```
int main()
{
    {
        int id = 0;
    }

    int id = 1;

    int id = 2;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Basic Data Types

- Integers: char, short, int, long (signed/unsigned)

```
unsigned int maximum_grade = 5;
char first_letter = 'A';
```

- Floating points: float, double

```
float pi = 3.14159f;
```

- Boolean: bool

```
bool success = true;
```

- You can determine the size of a type using the "sizeof" operator:

```
sizeof(int)
sizeof(float)
sizeof(bool)
```

ROYAL INSTITUTE OF TECHNOLOGY

Pointers / Dynamic Memory Allocation

- Pointer is a basic type which allows you to access arbitrary locations of memory

```
int *data = nullptr;
```

- You can allocate memory using new operator

```
data = new int;
```

- You can access the value of the location in the memory that the pointer points using the * operator

```
*data = 10;
```

- Memory that is manually allocated by new operator should be freed by delete operator

```
delete data;
```

ROYAL INSTITUTE OF TECHNOLOGY

Arrays

- Arrays in C++ are a set of elements that have contiguous locations in memory
- You can define an array using the new[] operator

```
float *buffer = new float[1024];
```

- Elements of an array can be accessed with the [] operator

```
buffer[0] = 1.5f;
float val = buffer[1];
```

- Allocated memory for an array should be eventually freed using the delete[] operator

```
delete [] data;
```

ROYAL INSTITUTE OF TECHNOLOGY

References

- References are a basic type in C++ which have similar functionality as pointers with some limitations.

```
int val = 10;
int &ref = val;
```

- References have to be assigned to another variable when they are defined

```
int &ref;
```

- References can’t point to nullptr

```
int &ref = nullptr;
```

- A reference can only be assigned to one variable, and you can never change that

ROYAL INSTITUTE OF TECHNOLOGY

Program Control

- You can condition your code to be executed only if certain expression is true using if/else statement

```
if(x == 1){
...
}
else if(x < 1){
...
}
else{
...
}
```

- You can use the switch statement to condition your code on the value of an integer

```
switch(val){
case 0:
   ...
   break;
case 1:
   ...
   break;
default:
   ...
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Program Control

- For loops can be used to do repetitive tasks

```
for(int i=0; i<10; i++){
   printf("Current number: %d\n", i);
}
```

- Another way of defining a loop is by using the while keyword

```
bool found = false;
while(!found){
  ...
}
```

- **break** keyword is used to exit a loop immediately
- **continue** keyword is used to skip an iteration in the loop

ROYAL INSTITUTE OF TECHNOLOGY

Functions

- Functions in C++ are used as a building block for modular programming
- Functions act as a black box, take a set of arguments, manipulate them, and return the result
- You can only access the functions that are declared before the current point in your program
- If you need to access functions that are defined later you need to provide a declaration

```
// function declaration
float add(float x, float y);


int main()
{
     float result = add(5, 10);
     printf("result: %f\n", result);
     return 0;
}


// function definition
float add(float x, float y)
{
    return x + y;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Arguments

```
// This function swaps values of x, and y
void swap(float& x, float& y)
{
    float temp = x;
    x = y;
    y = temp;
}

// This function swaps values of x, and y
void add(const float& x, const float& y)
{
     return x + y;
}

// Fill the content of the arr with random integers
void randomize_array(int *arr, int size)
{
    for(int i=0; i<size; ++i){
         arr[i] = rand();
    }
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Structures

```
struct Vector3
{
     float x;
     float y;
     float z;
};

Vector3 Vector3Add(const Vector3& v1, const Vector3& v2)
{
     Vector3 ret;
     ret.x = v1.x + v2.x;
     ret.y = v1.y + v2.y;
     ret.z = v1.z + v2.z;
     return ret;
}

int main()
{
     Vector3 start = {0, 0, 0};
     Vector3 movement = {1, 0, 0};
     Vector3 end = Vector3Add(start, movement);
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Operator Overloading

```
struct Vector3
{
    float x, y, z;
};

Vector3 operator+(const Vector3& v1, const Vector3& v2)
{
    Vector3 ret;
    ret.x = v1.x + v2.x;
    ret.y = v1.y + v2.y;
    ret.z = v1.z + v2.z;
    return ret;
}

int main()
{
    Vector3 start = {0, 0, 0};
    Vector3 movement = {1, 0, 0};
    Vector3 end = start + movement;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Namespaces

```
namespace math
{
    struct Vector3
    {
        float x, y, z;
    };

    Vector3 operator+(const Vector3& v1,
        const Vector3& v2)
    {
        Vector3 ret;
        ret.x = v1.x + v2.x;
        ret.y = v1.y + v2.y;
        ret.z = v1.z + v2.z;
        return ret;
    }
}
```

```
using namespace math;

int main()
{
    Vector3 start = {0, 0, 0};
    Vector3 movement = {1, 0, 0};
    Vector3 end = start + movement;
}
```

```
int main()
{
    math::Vector3 start = {0, 0, 0};
    math::Vector3 movement = {1, 0, 0};
    math::Vector3 end = start + movement;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Classes

```
class Vector3
{
public:
     Vector3() : x(0), y(0), z(0){}
     Vector3(float _x, float _y, float _z)
      : x(_x), y(_y), z(_z){}

     Vector3 operator+(const Vector3& v)
     {
          Vector3 ret;
          ret.x = x + v.x;
          ret.y = y + v.y;
          ret.z = z + v.z;
          return ret;
     }

     float getX(){ return x; }
     float getY(){ return y; }
     float getZ(){ return z; }

private:
     float x, y, z;
};
```

```
int main()
{
     Vector3 start(0, 0, 0);
     Vector3 movement(1, 0, 0);
     Vector3 end = start + end;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Source File vs Header File

- Rule of thumb:
  - All the function definitions should go into the source file.
  - All the class and function declarations should go into the header files. Note: the contents of header files can be shared between different source files

```
#pragma once

class Vector3{
public:
     Vector3();
     Vector3(float _x, float _y, float _z);
     Vector3 operator+(const Vector3& v);

private:
     float x, y, z;
};
```

```
#include "Vector3.h"

Vector3::Vector3() : x(0), y(0), z(0){}

Vector3::Vector3(float _x, float _y, float _z) :
     x(_x), y(_y), z(_z){}

Vector3 Vector3::operator+(const Vector3& v){
     Vector3 ret;
     ret.x = x + v.x;
     ret.y = y + v.y;
     ret.z = z + v.z;
     return ret;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

Templates

```
template<class T>
class Vector3
{
public:
     Vector3() : x(0), y(0), z(0){}
     Vector3(T _x, T _y, T _z)
      : x(_x), y(_y), z(_z){}

     Vector3 operator+(const Vector3& v)
     {
          Vector3 ret;
          ret.x = x + v.x;
          ret.y = y + v.y;
          ret.z = z + v.z;
          return ret;
     }

     T getX(){ return x; }
     T getY(){ return y; }
     T getZ(){ return z; }
private:
     T x, y, z;
};
```

```
int main()
{
     Vector3<int> start(0, 0, 0);
     Vector3<int> movement(1, 0, 0);
     Vector3<int> end = start + end;
}
```

```
int main()
{
     Vector3<float> start(0, 0, 0);
     Vector3<float> movement(1, 0, 0);
     Vector3<float> end = start + end;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

STL

- Standard Template Library (STL) is a library consisted of a set of commonly used classes, and functions
- STL heavily uses templates to provide generic building blocks that can be used for a variety of applications
- Examples:
  - Containers: vector, list, string
  - Algorithms: sort, search, unique
  - I/O: stream, fstream
  - Others: shared_ptr, weak_ptr, …

stl::string

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
     string str = string("Hello ") + string("World!");

     cout << str << endl;

     return 0;
}
```

ROYAL INSTITUTE OF TECHNOLOGY

stl::vector

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
	vector<int> values;
	values.push_back(150);
	values.push_back(250);
	values.push_back(100);

	cout << values[0] << " " << values[1] << " " << values[2] << endl;

	sort(values.begin(), values.end());

	cout << values[0] << " " << values[1] << " " << values[2] << endl;

	return 0;
}
```

```
150 250 100
100 150 250
```

ROYAL INSTITUTE OF TECHNOLOGY

Want more?

- Online resources:
  - http://cplusplus.com/doc/tutorial/
  - http://msdn.microsoft.com/en-us/library/3bstk3k5.aspx
- Tools:
  - Windows: Microsoft Visual Studio
  - Mac: Xcode
  - Linux: CodeLite, old school: vim, make, g++
- Questions:
  - Send email to: vahidk@kth.se

ROYAL INSTITUTE
OF TECHNOLOGY

- Happy Debugging!