EINSTEIN CENTER Digital Future

Technische Universität Berlin

Tutorial 2: C++ Basics

C++ Basics

Prof. Sangyoung Park

Module ”Vehicle-2-X: Communication and Control“

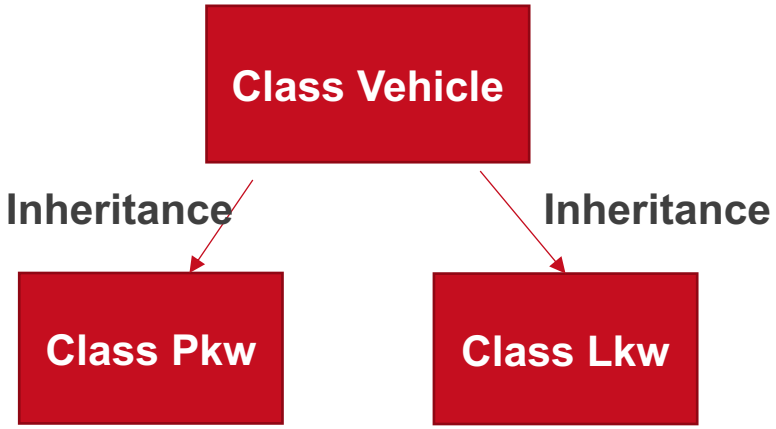
For those of you not familiar with C++

- C++ is an objective-oriented programming language
  - Deals with classes (not in C)
  - Classes have member functions and member variables
  - Public: any member can access the functions and variables
  - Protected: derived classes can access the functions and variables
  - Private: only the object itself can access the functions and variables
- Inheritance (derived?)
  - Classes can inherit other classes
- This is NOT a good guide. Please refer to other credible sources as well

Classes, Objects and Inheritance

- Class is like a user-defined blueprint

```
class Vehicle{
public:
    Vehicle(double, double);
    double getVelocity();
    void setVelocity(double velocity);
    double m_velocity;
    void proceedTime(double time);
    double getLocation();
protected:
    double m_x;
};
```



- Inheritance
  - You can inherit other classes to more easily make classes

```
class Pkw: public Vehicle{
public:
    Pkw(int);
    void setPassenger(int passenger);
    int getPassenger();
protected:
    int m_passenger;
};
```

Classes, Objects and Inheritance

- Objects
  - A class is only a template
  - When you declare a variable using a class, we say an object is instantiated
  - An object actually occupies some memory space during program execution
    - i.e., if you make 100 objects, it will occupy 100 times memory space than 1 object

```
int main(void)
{
    Pkw pkw = Pkw(4);
    printf("Num of passengers: %d\n", pkw.getPassenger());
    Pkw* pkw_p = new Pkw(5);
    printf("Num of passengers: %d\n", pkw_p->getPassenger()); // -> operator is used if pointer is used
    return 0;
}
```

How to Test C++ Features Yourself?

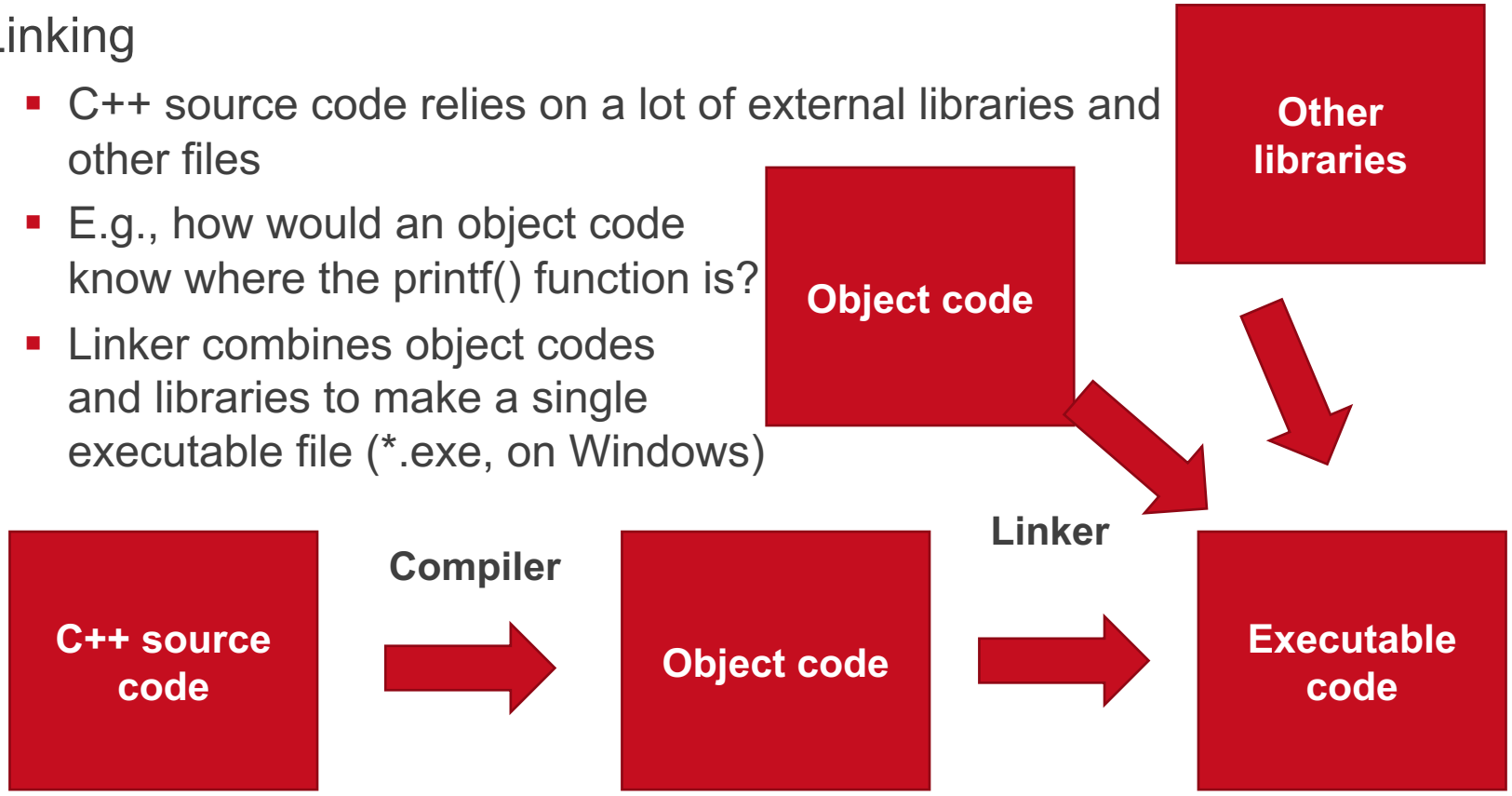
- I can't teach you everything in a single tutorial
- The best way to deal with it is to do it yourself
- If you are not familiar with C++, I can provide a very simple example code
  - Test.cc is about basics of classes
  - Car.cc is about basics of class inheritance
- Quickest way to test your code
  - On Msys (MinGW) terminal, type
    - >> g++ test.cc
    - >> ./a.exe
- You will be able to find out how the code runs
- Besides this, the programming is basically Googling
  - For example, if you are wondering what „printf" is,
  - Please google it for the function description
  - C++ standard library (STL) documentation is available on the web

Basic Flow

- Compilation vs interpretation
- C++ source code is compiled, then machine code is generated, which can then be executed
- Interpretation
  - Source code is execute on the fly
  - MATLAB, Python, etc.

Compilation & Linking

- Compilation
  - Compiles source code to generate object code (very close to machine code)
- Linking
  - C++ source code relies on a lot of external libraries and other files
  - E.g., how would an object code know where the printf() function is?
  - Linker combines object codes and libraries to make a single executable file (*.exe, on Windows)



How to Test C++ Features Yourself?

- Sometimes you are not sure about the C++ feature you are coding will work or not
- It’s better to test it on a simpler code and then integrate it in your code

Scope of Variables

- In scope.cc,
- You can find the function add5() which adds 5 to two inputs
- However, if you print the results after calling the function, the results will remain the same because the scope of variable a and b is limited inside add5()

Scope

```cpp
#include <iostream>

void add5(int, int);

int main(void){
    int a = 3;
    int b = 4;
    add5(a,b);
    std::cout << "a is :" << a << " b is : " << b;
    return 0;
}

void add5(int a, int b){
    a = a + 5;
    b = b + 5;
}
```

- Result

```
/c/Users/user/src/test$ g++ scope.cc

/c/Users/user/src/test$ ./a
a is :3 b is : 4
```

Pointers

- What are pointers?
  - A pointer variable is a variable which stores the address of an another variable
  - int* a: a variable named „a“ which points to an integer variable
  - Pkw* a: a variable named „a“ which points to an Pkw object
- Why use pointers?
  - It is sometimes convenient to have access to a memory location directly
- Example
  - add5() gets pointer variables as inputs
  - As add5() accesses the memory locations directly, it able to edit the values of the memory locations
- Result

```
/c/Users/user/src/test$ g++ pointer.cc

/c/Users/user/src/test$ ./a
a is :8 b is : 9
```

```cpp
#include <iostream>

void add5(int*, int*);

int main(void){
    int a = 3, b = 4;

    add5(&a, &b);

    std::cout << "a is :" << a << " b is : " << b;

}

void add5(int* a, int* b){
    *a = *a + 5;
    *b = *b + 5;
}
```

New Operator

- *New* operator creates an object somewhere in memory (called heap) and returns the pointer to the memory location
- The location is determined at „runtime“ by the operating system
- What happens if the pointer variable is redirected to somewhere else?
- Memory leak happens, you should use delete „delete a, b;“
- In new.cc,

Stack

a	0x08
b	0x0c

Heap

0x00	
0x04	
0x08	3
0x0c	4
0x10	
0x14	
..	

```
#include <iostream>

void add5(int*, int*);

int main(void){
    int* a = new int(3);
    int* b = new int(4);

    add5(a, b);

    std::cout << "a is :" << *a << " b is : " << *b;

    return 0;
}

void add5(int* a, int* b){
    *a = *a + 5;
    *b = *b + 5;
}
```

Accessing Member Variables and Functions

- In car.cc,

```
int main(void)
{
    Pkw pkw = Pkw(4);
    printf("Num of passengers: %d\n", pkw.getPassenger());
    Pkw* pkw_p = new Pkw(5);
    printf("Num of passengers: %d\n", pkw_p->getPassenger()); // -> operator is used if pointer is used
    return 0;
}
```

- You use . to access member variables and functions of an object
- You use -> to access member varialbes and functions of an object if a pointer is used