ENGR101: Lecture 18

C++ classes, high-level software design

May 2024

What we cover today?

- Group variables and function together - C++ classes
- How to design big programs?

Classes in C++. Organizing your code.

We discussed **struct** before - customized data type. **struct** is similar to **Object**s.
In COMP102 you went through **Object-Oriented programming**.
Is it applicable to C++?
Sure. There are very little differences between C++ and Java in that.
OO programming is a good way to organize your data when it is possible to separate data into objects (Camera and Motor objects make Robot, as an example).
We can group related functions and variable together.

As an example...

Class to describe image-processing part of the project:

Listing 1: Camera

```
class Camera{
  // function declarations
  double measure_line();
  double measure_intersection();
};
//function implementation
double Camera::measure_line(){
  return 0.0;
}
double Camera::measure_intersection(){
  return 0.1;
}

int main(){
  Camera camera;
  double err = camera.measure_line();
}
```

- We grouped all image-processing functions into one class
- Note semicolon at the end of class declaration
- Function implementations do not have to be inside class scope {..}. Scope resolution operator **::**.
- Declare variable of the type **Camera**.
- Does not compile (private)

Accessing class members(public, private)

Listing 2: Camera

```
class Camera{
 public:
   double measure_line();
   double measure_intersection();
};
double Camera::measure_line(){
  return 0.0;
}
double Camera::measure_intersection(){
  return 0.1;
}

int main(){
  Camera camera;
  double err = camera.measure_line();
}
```

class1.cpp

- To get access to class members (variables or functions) from outside the class - these members should be declared as **public**.
- Private variables can be by modified only by functions which are members of the class.
- Private functions can be called only by functions which are members of the class.

Simple and not so good solution:

Listing 3: All public

```
class Motor{
 public:
 // data
   double v_left;
   double v_right;
   // functions
   void forward(double dist);
   void back(double dist);
};

void Motor::forward(double dist){
   //code here
}

int main(){
   Motor m1;
   m1.v_left = 99.0;
   m1.forward(34);
}
```

class2.cpp

- Make everything **public**.
- Variables and functions are accessible now.
- Class went to another extreme - there is no data protecton
- It became equivalent to C++ **struct**

Getter and setter functions

Listing 4: Setter getter

```
#include <iostream>
class Motor{
 private:
   // data
   double v_left, v_right;
  // functions
  public:
    void set_v_left(double x){v_left=x;}
    double get_v_left(){ return v_leftx;}
    void forward(double dist);
};
void Motor::forward(double dist){
  //code here
}
int main(){
  Motor m1;
  m1.set_v_left(99.0);
  std::cout<<m1.get_v_left();
}
```

- both variables are **private** now
- **set_v_left** function - set value of **v_left**
- **get_v_left** function - returns value of **v_left**
- Both these functions are **public** and can be called from outside of the class
- If function is short (one-liner), declaration and implementation can be combined inside the class body

class2_5.cpp

Constructor

Listing 5: Constructor

```
class Motor{
  private:
    double v_left, v_right;
  public:
   Motor();
   void set_v_left(double x){v_left=x;}
   double get_v_left(){return v_left;}
 };
Motor::Motor(){ //constructor
        v_left = 78.0;
}

int main(){
  Motor m1; //constructor call
  std::cout<<m1.get_v_left();
}
```

class3.cpp

- It is not a good idea to leave declared variables not initialized
- Values are random: likely 0 but can be anything
- **Constructor** is special function in the class
- It is called when instance of class is created

Class instance as member of other class

Listing 6: Class member

```
class AVC{
 private:
    Camera camera;
    Motor motor;
   // data
  public:
    void go();
};

void AVC::go(){
  double err = camera.measure_line();
  motor.forward(45);
}

int main(){
  AVC avc;
  while (true){
   avc.go();
  }
}
```

If variable **motor** is of type **Motor** (and is member of **AVC**) you can call **Motor** functions using **.**: **motor.forward();**.
Recommended way is run one function from top class forever.
That is what
while (true) { **...**} does.
class4.cpp

Can we draw software structure?

Flow chart shows the algorithm.
It is not intended to represent code structure.

If you are using classes, there is a software to do high-level design.
Most widely used is **Universal Modelling Language**, UML for short. Its purpose is to provide visual representation of the software.
Umbrello is the software you can use. It is reasonably easy to use.

Draw the class

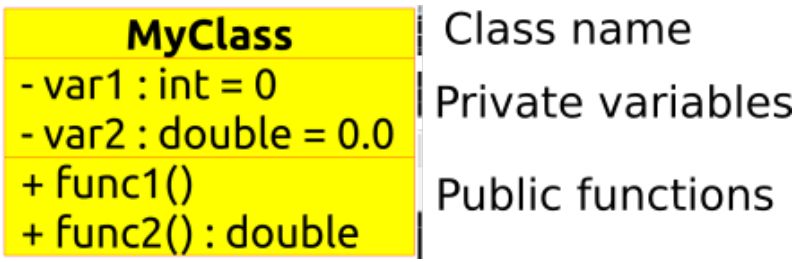

Figure: One class

- Each class in the program (can be many) is drawn as rectangle
- Variables are on the top, functions - in bottom part
- Private members are labeled with -, public - with +

Several classes

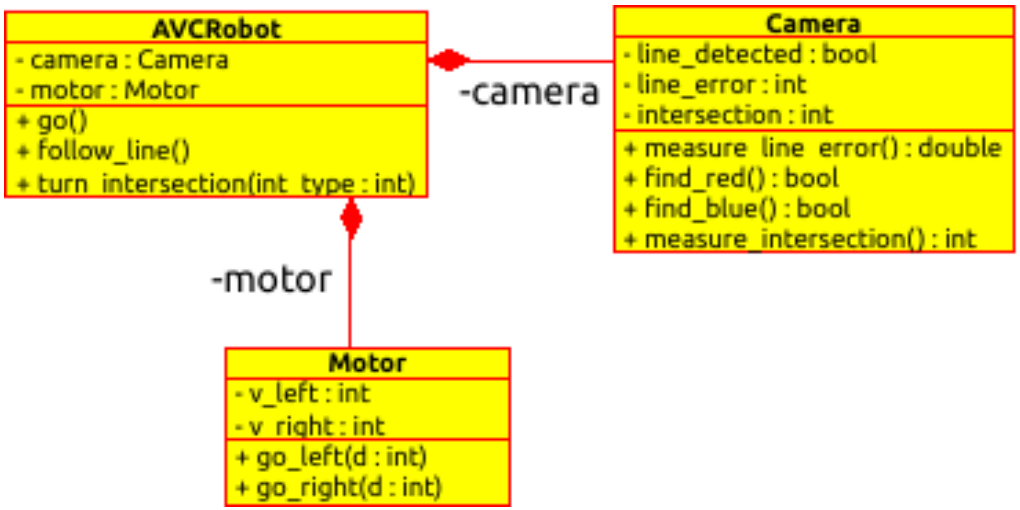

Figure: Class diagram

You can generate code from this diagram but I would not recommend it for now.

Note: Please do not use **Robot** as class name - it is already used in simulation library. Shown is just an example - your design can be totally different.

Another type of chart - sequence diagram

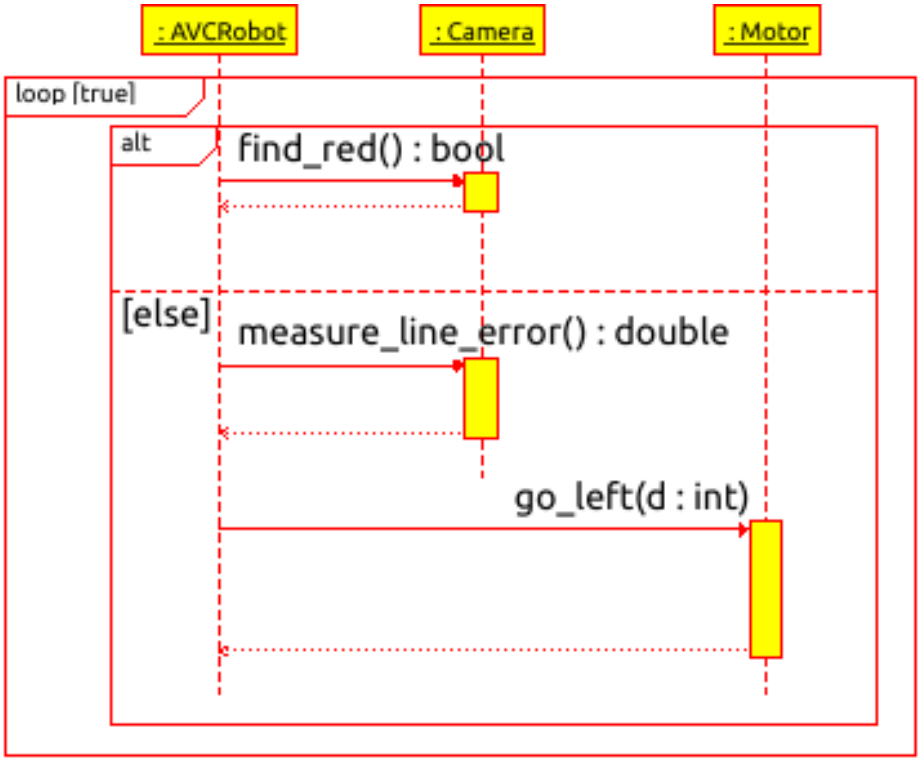

Figure: Sequence diagram

- Class diagram shows how your code is divided into pieces
- **Sequence** diagram shows how class functions are calling each other

Is it compulsory to use?

Not at all.
Is it useful - yes. Start with flow chart - to visualize your algorithm. It is not a requirement to use classes but it helps.