Lecture #06

Class (1)

SE271 Object-Oriented Programming (2020)

Yeseong Kim

Original slides from Prof. Shin at DGIST

Short Notice

- We don't have the lecture on Wednesday
- Change the way to get bonus points!
  - We will use **Doodle** instead of LMS
  - I will ask a few of simple short quiz during the class
  - For every answer, you can get one point for HW
- Some important news about HW probably would be out today
  - Stay tuned in LMS

Today's Topic

- Structure type
  - Struct (in c)

- Class
  - Definition
  - Declaration
  - Accessor

What we have learned so far…

- Data Type
  - Built-in data types
  - [], *, & types
- Their Basic Operation
- Flow control
- Function

Structure Type (in C)

- **Syntax**
  - Definition for the new structure type (cf. array type)

```
struct
{
    data_type variable1;
    data_type variable2;
    …
};
```

```
struct struct_tag
{
    data_type variable1;
    data_type variable2;
    …
};

int sID;
struct struct_tag student1;
```

  - Declaration for variables

```
struct {
    int sID;
    char sName[20];
} student1;     // student2;
```

Structure Type (in C)

- **Syntax**
  - Definition for the new structure type (cf. array type)

```
struct Student {
    int id;
    char name[20];
};
struct Student student1;
```

```
typedef struct {
    int id;
    char name[20];
} Student;
Student student1;
```

  - Initialization for structure variables

```
Student student1 = {201911999, “John”};
Student student2 = student1;
```

Structure Type (in C)

- **Syntax**
  - Usage

```
struct Student{
    int id;
    char name[20];
};

struct Student student1;
student1.id = 201911999;
strcpy_s(student1.name, "John");

printf("id: %d, name : %s\n",
student1.id, student1.name);
```

C++ style

```
struct  Student {
    int id;
    string name;
};

Student students[10];
Student* student1 = new Student{ 201911999,
"John" };

cout << students[0].id << endl;
cout << (*student1).id << endl;
cout << student1->id << endl;

delete student1;
```

[Recap] Very Short Introduction to OOP

- One of popular programming paradigms
  - Supporting following concepts based on Objects
    - Abstraction
    - Encapsulation
    - Inheritance
    - Polymorphism
  - Cf. Procedural programming (C style)

Class: User-defined Type

- Object: Everything can be **Object**.
  - 2 major components
    - A list of relevant properties (attributes, variables, …)
    - A list of behaviors (methods, functions, …)

- Class: a New user-defined type to represent an object
- Object: an individual **Instance** of a class

Class: User-defined Type

- **Syntax**
  - Definition for the new type (cf. struct type)

```
struct struct_tag
{
    data_type variable1;
    data_type variable2;
    …



};

struct_tag variable;     // c++
```

```
class ClassName
{
    data_type member_variable1;
    data_type member_variable2;
    …
    return_type member_function1(params);
    return_type member_function2(params);
};

ClassName instance;
```

Class: User-defined Type - Accessor

- **Syntax**
  - **Accessor**

```
class ClassName
{
private:
    data_type member_variable1;
public:
    data_type member_variable2;

    return_type member_function1(params);
    return_type member_function2(params);
};

ClassName instance;
```

1. private
2. public
3. protected

Class: User-defined Type - Usage

- **Syntax**
  - Usage (cf. struct)

```
struct  Student {
    int id;
    string name;
};
Student students[10];
Student * student1;
student1 = new Student{201911999, “Jo

cout << students[0].id << endl;
cout << (*student1).id << endl;
cout << student1->id << endl;

delete student1;
```

Problematic Code

```
class  Student {
private:
    int id;
    string name;
public:
    int GetID() { return id; }
};
Student students[10];
Student* student1;
student1 = new Student{ 201911999, "John" };

cout << students[0].id << endl;
cout << (*student1).id << endl;
cout << student1->id << endl;
delete student1;
```

Class: User-defined Type – method definition

- **Syntax**

student.h

```
class  Student {
private:
    int id;
    string name;
public:
    int GetID() { return id; }
    void SetID(int);
};
```

student.cpp

```
void Student::SetID(int ID)
{
    id = ID;
}
```

References

- Learn c++
  - https://www.learncpp.com/
  - Chapter 8

Example: class

```
#include <iostream>
class Student {
    int * m_id{ nullptr };
public:
    Student(int id) {
        m_id = new int(id);
    }
    ~Student() {
        delete m_id;
    }
    int GetStudent() {
        return *m_id;
    }
};
```

```
int main() {
    Student s1(201911999);
    Student s2{ s1 };
    std::cout << s1.GetStudent() << std::endl;
    std::cout << s2.GetStudent() << std::endl;
    return 0;
}
```

```
201911999
201911999
// error
```

ANY QUESTIONS?