C++ 스터디

6. 클래스(1)

D.Com

포인터와 어레이 복습

p가 포인터일 때,

p[i] == *(p+i)

p[j] [i] == *(*(p+j)+i)

p는 시작지점의 주소!!!

Software Component

- 이미 우리는 다양한 Software Component, 소프트웨어를 구성하는 구성요소,들을 사용해 왔다.
- cin, cout / string / fstream, ifstream, ofstream 등이 모두 Software Component이다.
- Software Component = Data + Function 이다.
- 즉 Data를 객체의 상태(State), Function을 객체의 행위(Behave)로 볼 수 있다.

Software Component (Class & Object)

- Class와 객체의 예시로 fstream 클래스(라이브러리)와 fstream myFile;을 들 수 있다.
- Class
  - 추상적 개념체 (건물의 설계도?)
  - fstream은 그 자체로는 아무런 실제 정보를 가지지 않으므로 class이다.
- 객체
  - Class의 실체화 (위치라는 정보를 가지는 실제 건물)
  - fstream myFile;은 이제 정보(file)를 가지고 존재한다. 그러므로 객체이다.

- 즉 객체는 Class의 한 사례라고 할 수 있다.
  (Class = 프로그래머가 설정한 타입 / 객체 = 클래스의 한 instance)

D.Com

Software Component (Class 만들기)

1. 객체의 상태를 구성하는 데이터 정하기! (멤버 변수 정의하기)
2. 객체를 이용하는 클라이언트에게 제공해지는 서비스들에 대한 객체의 실행 코드 작성하기(멤버 함수 정의하기)
3. 새로 만들어진 객체를 초기 상태로 만들 코드 작성하기 (생성자 정의하기)
4. private, public, protected의 접근 제어 지시자를 이용해 어떤 것을 클라이언트에게 보이고 감출 것인지 정하기 (이들에 대해서는 아래 표와 나중 ppt 참조, 클래스 멤버 = 멤버 변수 + 멤버 함수)

키워드	의미
private	선언된 클래스 멤버는 외부에 공개되지 않으며, 외부에서 직접 접근할 수도 없다.
public	선언된 클래스 멤버는 외부로 공개되며, 해당 객체를 사용하는 프로그램 어디에서나 직접 접근할 수 있다.
protected	protected 멤버는 파생 클래스에 대해서는 public 멤버처럼 취급되며, 외부에서는 private 멤버처럼 취급된다. 상속과 관련된 접근제어지시자로 나중에 다시 배운다!

D.Com

Class (선언하기)

- 클래스의 선언 방법은 다음과 같다.

```
class 클래스명 {
접근제어지시자:
        멤버변수1타입 멤버변수1이름
        멤버변수2타입 멤버변수2이름
        …
        멤버함수1
        멤버함수2
        …
};
```

벡터 컨테이너를 이용해 클래스를 담는 것도 가능하다.

```
1   #include <iostream>
2   #include <vector>
3   using namespace std;
4
5   class Point {
6   public:
7       double x;
8       double y;
9   };
10
11  int main() {
12      vector<Point> points(100);
13  }
```

D.Com

Class (예시)

- 아래는 멤버 데이터만을 가지는 좌표 클래스의 예시이다.

```
1   #include <iostream>
2   using namespace std;
3
4   class Point {
5   public:
6       double x;
7       double y;
8   };
```

```
1   #include <iostream>
2   using namespace std;
3
4   class Point {
5   public:
6       double x;
7       double y;
8   };
9
10  int main() {
11      Point pt1;
12      pt1.x = 1.0; // public 변수이기에 클래스 외부 접근 가능
13      pt1.y = 1.3;
14      cout << "X = " << pt1.x << " Y = " << pt1.y << endl;
15  }
16
```

```
선택 Microsoft Visual Studio 디버그 콘솔
X = 1 Y = 1.3

C:₩Users₩bluej₩Desktop₩방학프로그래밍₩c++스터
```

D.Com

Class (멤버 함수)

- 멤버함수는 클래스 내부의 함수로 멤버 데이터들을 이용하는데 사용된다.
- 같은 객체 내에서의 멤버 데이터에 멤버 함수가 접근하기 위해서는 '.'이 필요 없다. (pt1.x -> x)
- 다른 객체에서의 멤버 데이터에 접근 하기 위해서는 '.'을 필요로 한다. (pt1.x)
- 멤버 함수의 정의는 클래스 선언 안이나 밖에서 할 수 있다.
- 멤버 함수의 호출은 기존 standard Class에서 하던 것처럼 하면 된다. (fstream, string, 등)

  acct1.deposit(1000);

```
1   #include <iostream>
2   #include <vector>
3   using namespace std;
4
5   class Account {
6   private:
7       string name;
8       int id;
9       double balance;
10  public:
11      void print() {
12          cout << name << ", " << id << ", " << balance << endl;
13          //클래스 내부 변수이기에 '.'을 사용하지 않음
14      }
15      void deposit(double amt); // 클래스 밖에서 정의
16  };
17  // 클래스명:: 을 붙여 특정 클래스의 함수 정의임을 알려줌
18  void Account::deposit(double amt) {
19      balance += amt;
20  }
```

D.Com

Class (생성자)

- class의 생성자는 객체가 생성 될 때 무조건 호출되는 함수로 객체의 초기 상태를 구성하고 정의하는 데에 사용된다.
- 몇몇 생성자들은 초기 구성을 위한 정보(변수들)을 인자로 받기도 한다.
- 생성자는 class와 같은 이름을 가진다.
- 생성자는 리턴 타입이 없다. (void도 적지 않는다!)
- 디폴트 인자를 이용하여 생성자를 정의할 수도 있다. (아래)

```
Point(double _x = 0, double _y = 0) :
    x(_x), y(_y) {};
```

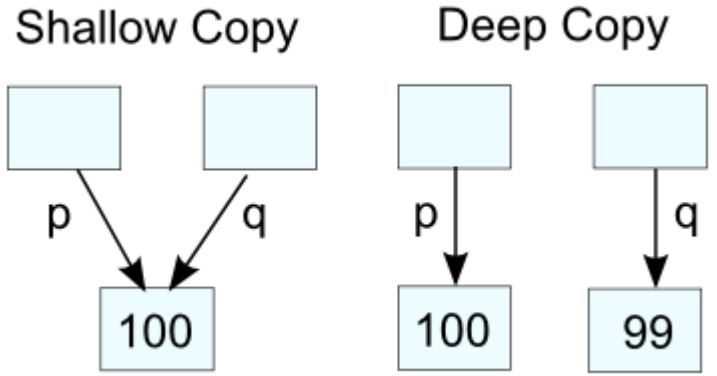
```
#include <iostream>
using namespace std;

class Account {
private:
    string name;
    int id;
    double balance;
public:
    Account(string _name, int _id, double _balance) {
        name = _name, id = _id, balance = _balance;
        // _는 그냥 임의로 구분하려고 붙인것 , 생성자임
        if (_balance < 0) {
            balance = 0;
            cout << "잔액은 최소 0원입니다." << endl;
        }
    }
    void print() {
        cout << name << ", " << id << ", " << balance << endl;}
    void deposit(double amt) {
        balance += amt;}
};

int main() {
    Account a("심규진", 123, 100);
    a.print();
}
```

```
Microsoft Visual Studio 디버그 콘솔
심규진, 123, 100
```

D.Com

Class (생성자)

- 그렇다면 생성자가 없는데도 어떻게 앞에서는 객체(클래스의 표현형)를 정의한 것일까?
- 생성자를 정의하지 않은 경우에는 아무런 파라미터를 가지지 않고 아무런 기능을 하지 않는 기본 생성자를 기본으로 실행한다. 그러므로 Account a; 는 제대로 동작할 것이다.
- 그러나 어떤 생성자라도 정의한 순간 컴파일러가 default 생성자를 제공하지 않으므로 Account a; 와 같이 적은 경우 에러가 날 것이다.
- 기본 생성자의 호출 방법은 3가지 이다.
  1. 클래스명 변수명;
  2. 클래스명 변수명 = 클래스명();
  3. 클래스명* 변수명= new 클래스명;



얕은 복사(Shallow Copy)
Point pt1 (1,2);
Point pt2 = pt1;
// 주소 값을 전달해 둘이 같은 객체를 나타내게 됨.

깊은 복사(Deep Copy)
// 메모리 공간을 확보해
// 각각 다른 객체가 됨
참조 : https://sosal.kr/243

D.Com

Class (생성자)

- 앞의 생성자를 initialization list를 사용하여 이렇게 적을 수도 있다. const로 선언된 멤버 변수는 생성자 내에서 조작이 불가능 하지만, 초기화 리스트로 설정하는 것은 가능하다.

```
10      Account(string _name, int _id, double _balance)
11          : name(_name), id(_id), balance(_balance)
12      {if (_balance < 0) {
13          balance = 0;
14          cout << "잔액은 최소 0원입니다." << endl;}}
```

```
1   #include <iostream>
2   using namespace std;
3
4   class MyClass {
5   private:
6       const int num;
7   public:
8       MyClass() : num(1) { // num = 1; 이렇게 적으면 오류남
9       };
10  };
```

D.Com

Class (생성자)

- 생성자를 여러 개 정의하는 것도 물론 가능하다. 다만 각 생성자의 파라미터는 반드시 달라야 한다.

```
1   #include <iostream>
2   using namespace std;
3
4   class S {
5     int n; //아무런 접근제어지시자를 적지않으면
6     //기본 private로 됨
7
8   public:
9     S(int _n) {
10      n = 1;
11    }
12    S() : n(0) {}
13  };
```

Class (예시)

- vector에 class를 담는 것도 가능하다.
  마찬가지로 어레이에도 담을 수 있다.

```cpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Human {
private:
    string name;
    int age;
    double length;
    double weight;

public:
    Human(string _name, double _length, double _weight) {
        name = _name, length = _length, weight = _weight;
        age = 1;
    }
    Human() {
        age = 1;
        cout << "이름/키/몸무게를 입력하세요 : ";
        cin >> name >> length >> weight;
    }

    void print() {
        cout << "이름 : " << name << " 나이 : " << age
             << " 키 : " << length << " 몸무게 : " << weight << endl;
    }

    void pass_one_year() {
        age++;
        length *= 1.1;
        weight *= 1.1;
    }
    // getter = get 함수
    string get_name() { return name; }
    int get_age() { return age; }
    double get_length() { return length; }
    double get_weight() { return weight; }
    // setter = set 함수
    void set_name(string _name) { name = _name; }
    void set_age(int _age) { name = _age; }
    void set_length(double _length) { name = _length; }
    void set_weight(double _weight) { name = _weight; }

};

void all_pass_one_year(vector<Human> v) {
    for (auto elem : v) {
        elem.pass_one_year();
        elem.print();
    }
}

int main() {
    vector<Human> god;
    while (true) {
        Human temp;
        int is_end;
        god.push_back(temp);
        cout << "입력을 마치려면 0을 입력하세요 : ";
        cin >> is_end;
        if (is_end == 0) break;
    }
    all_pass_one_year(god);
    all_pass_one_year(god);
    return 0;
}
```

D.Com

이해를 돕기 위한 페이지

- 즉 여태까지 써왔던 〈string〉 〈iostream〉 〈vector〉 등의 표준 라이브러리는 전부 클래스이다!
- 〈string〉을 예시로 들자면 string a; 이렇게 선언한 순간 string 클래스의 a라는 객체가 생성된 것(기본 생성자를 이용해 초기데이터 x)
- a가 가진 문자열이 멤버 데이터 (= 멤버 변수)이고 at, length, size, find 등의 함수가 메소드 (= 멤버 함수)이다.

D.Com

캡슐화

- 접근제어지시자(private, public, protected)의 키워드를 사용해서 사용자가 굳이 알 필요 없는 정보를 숨기고 노출의 범위를 정하는 이유는 보안과 코드의 안정성을 위해서이다.
- private:
  - private로 선언된 변수들은 그 클래스의 다른 멤버들에 의해서만 사용될 수 있다. 주로 getter와 setter를 이용해 변수들을 컨트롤한다.
  - 이것이 캡슐화를 이루는 한 방식으로 프로그램의 안정적 구현과 함부로 건드려서는 안되는 코드를 보호하는 데에 사용된다.
  - C++은 별다른 접근 제어 지시자가 없다면 기본으로 private로 선언한다.
- public:
  - public으로 선언된 함수와 변수는 프로그램의 어디서든 접근 가능하다.
  - 멤버 함수들은 public으로 선언해준다.

캡슐화

- 캡슐화란 객체 지향 프로그래밍을 위한 방법으로 전체 소프트웨어를 부분으로 나눠 기계를 조립하는 부품처럼 하나의 소프트웨어를 조립하는 것이다.
- 캡슐화를 하는 이유는 코드의 재활용성을 높이고 에러 발생을 최소화하며 동적인 구현을 강화하기 위해서 이다.
- 캡슐화를 위한 규칙으로 데이터는 private로 만들고 클라이언트에게 서비스를 제공하는 메소드는 public으로 만들어야 한다.

캡슐화 (예시)

- private한 변수 들에는 그냥 접근할 수 없으므로 get함수와 set함수를 만들어 컨트롤 해준다.
- get함수와 set함수를 각각 getter와 setter라고 하며 private 변수의 값을 리턴 하거나 대입하는 액세스 함수라고 한다.

```cpp
#include <iostream>
#include <string>
using namespace std;

class Account {
private:
    string name;
    int id;
    double money;

public:
    Account(string _name, int _id) :
        name(_name), id(_id), money(0.0) {}
    // 초기화 리스트를 이용 멤버 변수 초기화

    void set_name(string _name) { name = _name; }
    void set_id(int _id) { id = _id; }
    void set_money(double _money) { money = _money; }
    string get_name() { return name; }
    int get_id() { return id; }
    double get_money() { return money; }
};

int main() {
    Account a("심규진", 1111);
    cout << a.id << endl; //private 변수에 그냥 접근하면 오류
    cout << a.get_id() << endl; //getter와 setter 사용해 접근
}
```

D.Com

과제 6

- 랩 6 푸세요~