7강. 구조체와 클래스

C++ 프로그래밍

jhhwang@kumoh.ac.kr

컴퓨터공학과 황준하

목차

- 구조체
  - 구조체의 개념
  - 구조체 사용 방법
- 클래스
  - 클래스의 개념
  - 클래스 기본 사용 방법

구조체의 개념

- 구조체란?
  - 여러 개의 데이터(변수)들을 하나의 단위로 묶는 방법
  - int와 같은 새로운 타입을 만듦

예1: 2차원 평면 상의 점

```
struct Point {
    int x;
    int y;
};
```

예2: 학생에 대한 정보

```
struct Student {
    char name[10];
    int age;
    int grade;
    double credit;
};
```

구조체의 개념

- 구조체의 필요성
  - 2차원 평면상의 점 2개를 더하는 함수를 작성해 보라.
    - (1, 2) + (3, 4) → (4, 6)

구조체를 사용하지 않는다면?

```
void main(void)
{
    int x1 = 1, y1 = 2;
    int x2 = 3, y2 = 4;
    int x3, y3;

    x3 = Sum(x1, y1, x2, y2);   // 어떡해?
}
```

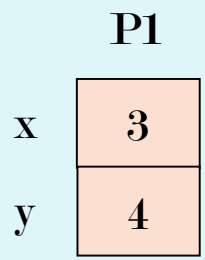
구조체를 사용한다면!

```
void main(void)
{
    Point P1 = { 1, 2 };
    Point P2 = { 3, 4 };
    Point P3;

    P3 = Sum(P1, P2);   // Wow!
}
```

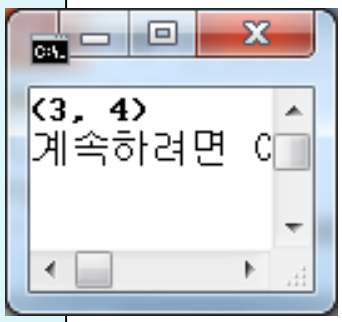
구조체 사용 방법

- 구조체 사용
  - 기본적으로 int와 동일
  - 변수 선언, 대입, 배열, 포인터, 참조, 매개변수 전달, 반환, ...

```
struct Point {
        int x;
        int y;
};

void main(void)
{
        Point P1;
        P1.x = 3;
        P1.y = 4;
        cout << "(" << P1.x << ", " << P1.y << ")" << endl;
}
```



구조체변수.멤버변수



구조체 사용 방법

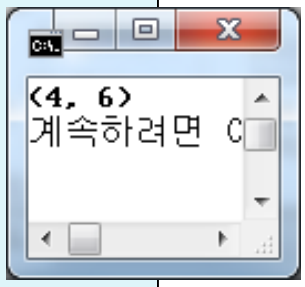
- 구조체 변수 선언 및 초기화, 매개변수 전달 및 반환

```
struct Point {
      int x;
      int y;
};

Point Sum(Point Po1, Point Po2)
{
      Point Po3;
      Po3.x = Po1.x + Po2.x;
      Po3.y = Po1.y + Po2.y;
      return Po3;
}
```

매개변수 전달, 반환

```
void Print(Point Po)
{
      cout << "(" << Po.x << ", "
           << Po.y << ")" << endl;
}

void main(void)
{
      Point P1 = { 1, 2 };
      Point P2 = { 3, 4 };
      Point P3;

      P3 = Sum(P1, P2);
      Print(P3);
}
```

변수 선언 및 초기화

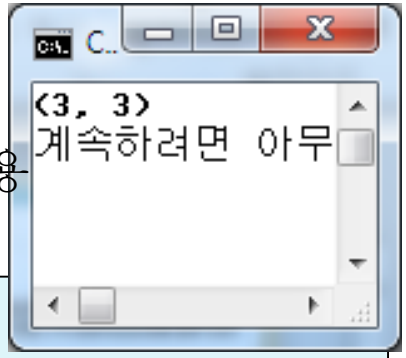


구조체 사용 방법

- 구조체 배열
  - Point 구조체 변수를 원소로 갖는 1차원 배열을 매개변수로 받아 모든 원소(점)들의 평균 점을 구해 반환하는 함수를 만

```
Point Avg(Point *Ary, int Count)
{
    Point temp = { 0, 0 };
    for (int i = 0; i < Count; i++) {
        temp.x += Ary[i].x;
        temp.y += Ary[i].y;
    }
    temp.x = temp.x / Count;
    temp.y = temp.y / Count;
    return temp;
}
```

기본적으로 int와 동일
포인터는 배열처럼 사용

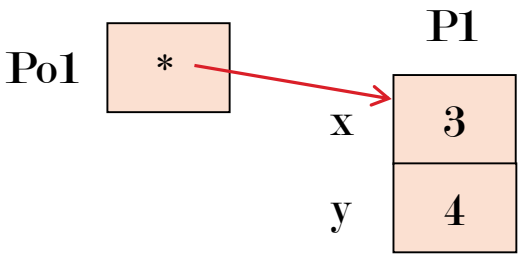



```
void main(void)
{
    Point Ary[5] = { { 1, 1 }, { 2, 2 },
        { 3, 3 }, { 4, 4 }, { 5, 5} };
    Point Po = Avg(Ary, 5);
    Print(Po);
}
```

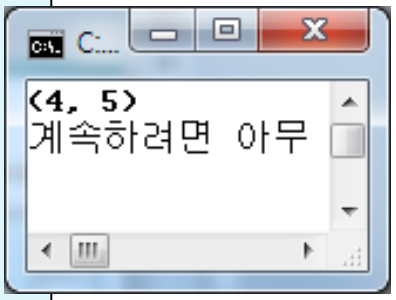
구조체 사용 방법

- 구조체 포인터
  - Point 변수 2개를 매개변수로 전달받아 첫 번째 Point 변수의 값을 두 번째 Point 변수의 값만큼 이동하는 Move 함수를 작성하라.

구조체 포인터 변수로부터 멤버 변수 접근 방법
(*Po1).x
Po1->x




```
void Move(Point *Po1, Point Po2)
{
    Po1->x = Po1->x + Po2.x;
    Po1->y = Po1->y + Po2.y;
}

void main(void)
{
    Point P1 = { 1, 1 };
    Point P2 = { 3, 4 };
    Move(&P1, P2);
    Print(P1);
}
```



구조체 사용 방법

- 구조체 참조
  - Point 변수 2개를 매개변수로 전달받아 첫 번째 Point 변수의 값을 두 번째 Point 변수의 값만큼 이동하는 Move 함수를 작성하라

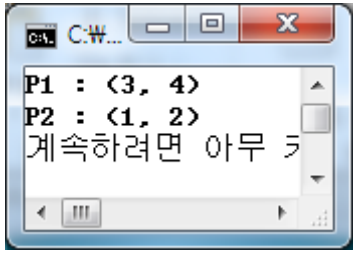
```
void Move(Point &Po1, Point Po2)
{
	Po1.x = Po1.x + Po2.x;
	Po1.y = Po1.y + Po2.y;
}

void main(void)
{
	Point P1 = { 1, 1 };
	Point P2 = { 3, 4 };
	Move(P1, P2);
	Print(P1);
}
```

P1과 Po1은 동일한 변수

연습 문제

- Point 구조체 변수 2개를 매개변수로 받아 두 변수의 값을 교환하는 Swap 함수를 만들어 보라.
  - 포인터를 사용할 수도 있고, 참조를 사용할 수도 있다.
  - 참조를 사용하여 구현해 보라.

```
void main(void)
{
	Point P1 = { 1, 2 };
	Point P2 = { 3, 4 };
	Swap(P1, P2);

	cout << "P1 : ";
	Print(P1);
	cout << "P2 : ";
	Print(P2);
}
```



연습 문제

- 프로그램 확인

```
void Swap(Point &Po1, Point &Po2)
{
	Point temp;

	temp.x = Po1.x;
	temp.y = Po1.y;

	Po1.x = Po2.x;
	Po1.y = Po2.y;

	Po2.x = temp.x;
	Po2.y = temp.y;
}
```

```
void Swap(Point &Po1, Point &Po2)
{
	Point temp;

	temp = Po1;
	Po1 = Po2;
	Po2 = temp;
}
```

구조체 변수 대입 가능

클래스의 개념

- 클래스란?
  - 여러 개의 데이터(변수)들과 그 데이터들을 조작할 수 있는 함수들을 하나의 단위로 묶는 방법
  - int와 같은 새로운 타입을 만듦
  - C++에서 구조체 == 클래스

```
class CPoint {
    int x;
    int y;

    void Print() {
        cout << "(" << x << ", "
            << y << ")" << endl;
    }
    void Move(CPoint Po) {
        x = x + Po.x;
        y = y + Po.y;
    }
};
```

클래스의 개념

- 클래스의 필요성
  - 객체지향 프로그래밍
    - 데이터 캡슐화 == 정보 은닉
    - 상속
    - 다형성
  - 정보 은닉
    - 예: Point의 x, y 좌표값이 1~10만 가능하다면...

구조체를 사용하는 경우 (단순 데이터의 묶음)

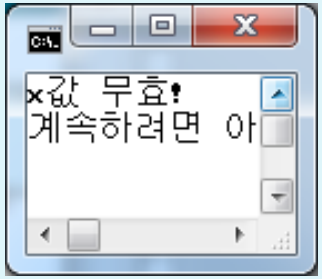
```
struct Point {
	int x;
	int y;
};

void main(void)
{
	Point P1;
	P1.x = 100; // 컴파일러는 오류 모름!
}
```

클래스의 개념

- 클래스의 필요성 (계속
  - 정보 은닉
- 정보 은닉 방법
  - private 영역
    - 내부접근 가능
    - 외부접근 불가
  - public 영역
    - 내부접근 가능
    - 외부접근 가능
- 클래스 변수 → 객체

클래스를 사용하는 경우

```
class CPoint {
private :
        int x;
        int y;

public :
        void SetX(int a) {
                if (a < 1 || a > 10)
                        cout << "x값 무효!" << endl;
                else
                        x = a;
        }
};

void main(void)
{
        CPoint P1;
        P1.SetX(100);
}
```



클래스 기본 사용 방법

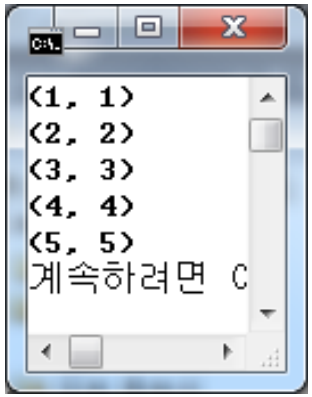
- 클래스 사용 방법
  - 구조체와 동일
    - 멤버(변수, 함수) 접근 방법 : 객체.SetX(3) 객체포인터->SetX(3)
    - 객체 선언, 대입, 배열, 포인터, 참조
  - 추가 학습 내용
    - 객체 선언(생성) 시 초기화 방법 : 생성자와 소멸자
    - 복사 생성자
    - 연산자 오버로딩
    - 상속
    - 상속과 다형성

클래스 기본 사용 방법

- 객체 배열

```
class CPoint {
private :
	int x;
	int y;

public :
	void SetXY(int a, int b) {
		x = a;
		y = b;
	}
	void Print() {
		cout << "(" << x << ", " << y << ")" << endl;
	}
};
```

```
void main(void)
{
	CPoint Po[5];

	for (int i = 0; i < 5; i++)
		Po[i].SetXY(i + 1, i + 1);

	for (int i = 0; i < 5; i++)
		Po[i].Print();
}
```



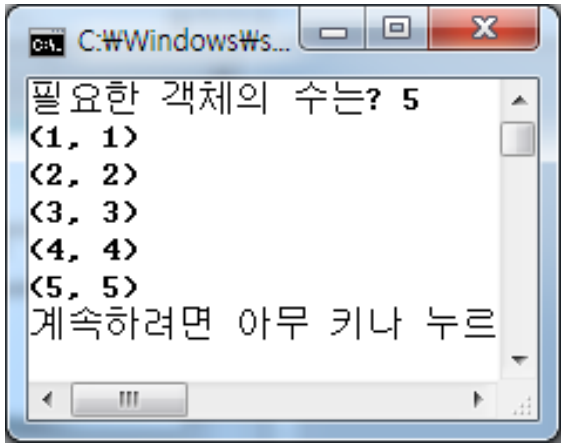
클래스 기본 사용

- 객체 포인터

int와 같이 메모리 동적 할당 가능

pTemp로 첫 번째 원소 가리킴

포인터로 접근 시 -› 연산자 사용
포인터 연산 int와 개념 동일

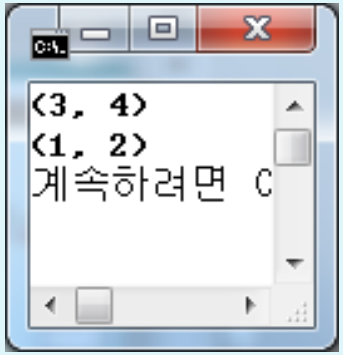



```
void main(void)
{
	int Count;
	CPoint *pPo;

	cout << "필요한 객체의 수는? ";
	cin >> Count;
	pPo = new CPoint[Count];

	CPoint *pTemp = pPo;
	for (int i = 0; i < Count; i++) {
		pTemp->SetXY(i + 1, i + 1);
		pTemp++;
	}

	pTemp = pPo;
	for (int i = 0; i < Count; i++) {
		pTemp->Print();
		pTemp++;
	}

	delete [] pPo;
}
```

클래스 기본 사용 방법

- 참조
  - CPoint 클래스 객체 2개를 매개변수로 전달받아 값을 교환하는 Swap 함수를 작성해 보라.
  - 기본적인 방법은 int나 구조체와 동일

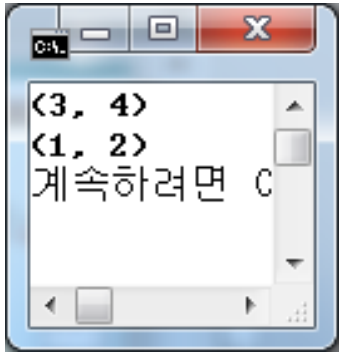
```
void Swap(CPoint &Po1, CPoint &Po2)
{
    CPoint temp = Po1;
    Po1 = Po2;
    Po2 = temp;
}

void main(void)
{
    CPoint P1, P2;
    P1.SetXY(1, 2);
    P2.SetXY(3, 4);

    Swap(P1, P2);

    P1.Print();
    P2.Print();
}
```



연습 문제

- 다음과 같이 실행할 수 있도록 Swap 함수를 CPoint의 멤버 함수로 구현해 보라.

```
void main(void)
{
	CPoint P1, P2;
	P1.SetXY(1, 2);
	P2.SetXY(3, 4);

	P1.Swap(P2);

	P1.Print();
	P2.Print();
}
```



연습 문제

P1.Swap(P2);

- 프로그램 확인

```
class CPoint {
private :
    int x;
    int y;

public :
    void SetXY(int a, int b) {
        x = a;
        y = b;
    }
    void Print() {
        cout << "(" << x << ", " << y << ")" << endl;
    }
```

내부 접근이므로 private에 접근 가능

```
    void Swap(CPoint &Po) {
        CPoint temp;

        temp.x = x;
        temp.y = y;

        x = Po.x;
        y = Po.y;

        Po = temp;
    }
};
```

멤버 변수 별로 접근?
→ 9강에서 this 포인터 소개