8강. 생성자와 소멸자

C++ 프로그래밍

jhhwang@kumoh.ac.kr

컴퓨터공학과 황준하

목차

클래스 객체 생성과 초기화의 문제점

- 기존 변수의 선언과 동시에 초기화 방법
  - 일반 변수 : int a = 3;
  - 배열 : int ary[3] = { 1, 2, 3 };
  - 구조체 : Point P1 = { 3, 4 };
- 클래스 객체 생성 시 초기화의 문제점
  - CPoint P1 = { 3, 4 };
    - 가능한가?
    - 불가능한가?

멤버변수가 public인 경우

```
class CPoint {
public :
      int x;
      int y;
};
```

가능: 외부접근 Ok

멤버변수가 private인 경우

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

불가능: 외부접근 No

생성자

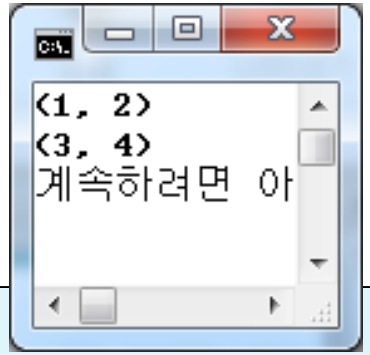
- 클래스 객체의 선언과 동시에 초기화하는 방법 → 생성자
  - 일반적으로 public 영역에 존재하는 멤버함수의 일종
  - 객체가 생성되면 반드시 하나의 생성자가 호출됨
- 생성자 만드는 방법
  - 생성자 이름은 클래스 이름과 동일
  - 반환형이 없음
  - 그 외에는 일반 함수와 동일
    - 디폴트 매개변수 가능
    - 생성자 오버로딩 가능

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

public :
	CPoint(int a, int b) { x = a; y = b; }
};
```

생성자

- 생성자 사용 방법 (= 생성자 호출 방법)

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

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




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

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

주의 : 매개변수가 없는 생성자 호출 방법

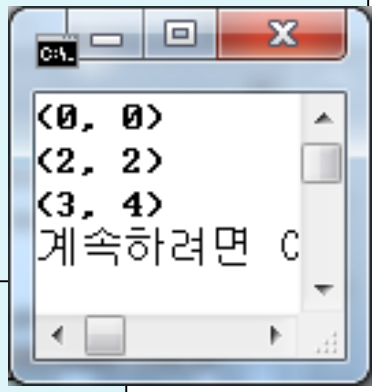
```
CPoint() { x = 0; y = 0; }
CPoint P1;      (O)
CPoint P1();    (X) → 함수 프로토타입
```

생성자 : 연습 문제 (1)

- 다음 프로그램 실행화면과 같이 실행될 수 있도록 CPoint 클래스의 생성자를 추가하라.

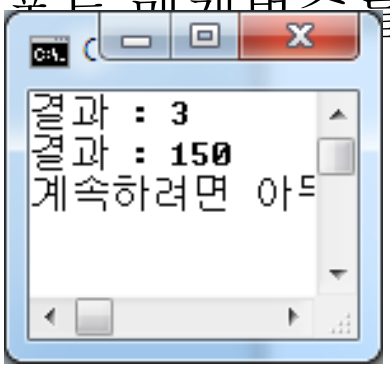
```
class CPoint {
private :
    int x;
    int y;

public :
    CPoint() { x = 0; y = 0; }
    CPoint(int a) { x = a; y = a; }
    CPoint(int a, int b) { x = a; y = b; }
    void Print() { cout << "(" << x << ", " << y << ")" << endl; }
};
```

```
void main(void)
{
    CPoint P1;          // (0, 0)
    CPoint P2(2);       // (2, 2)
    CPoint P3(3, 4);    // (3, 4)

    P1.Print();
    P2.Print();
    P3.Print();
}
```



생성자 : 연습 문제 (2)

- main 함수를 참고하여 덧셈, 뺄셈 결과를 저장하는 CCalc 클래스를 만들어 보라.
  - 멤버 변수로는 최종 결과를 저장하는 int Result 변수가 있다.
  - 객체 생성 시 초기값이 넘어오면 그 값을 최종 결과로 사용하고 넘어오지 않으면 0으로 초기화한다.
    - 생성자를 위해 디폴트 매개변수를 사용




```
void main(void)
{
	CCalc Calc1;
	CCalc Calc2(100);

	Calc1.Plus(5);
	Calc1.Minus(2);
	Calc1.Print();

	Calc2.Plus(100);
	Calc2.Minus(50);
	Calc2.Print();
}
```

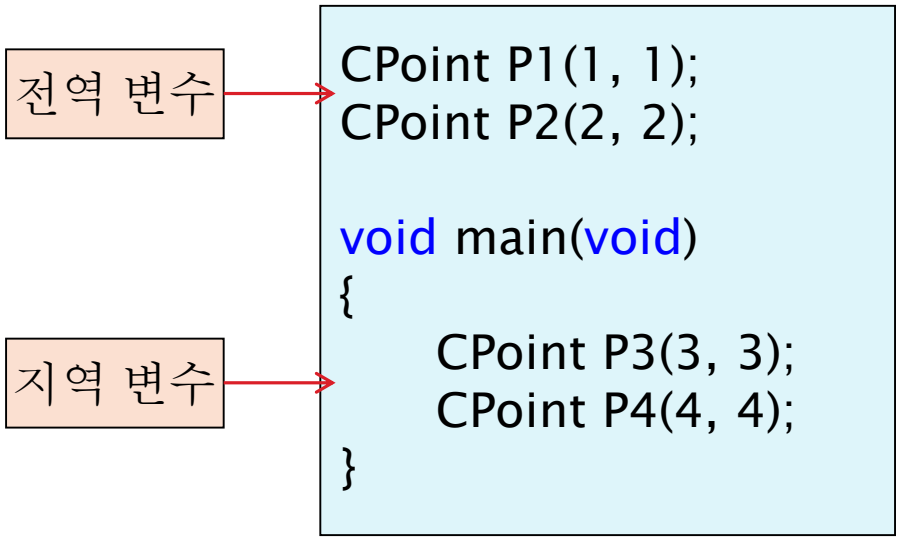
생성자 : 연습 문제 (2)

- 프로그램 확인

```
class CCalc {
private :
	int Result;

public :
	CCalc(int Init = 0) { Result = Init; }
	void Plus(int Num) { Result += Num; }
	void Minus(int Num) { Result -= Num; }
	void Print() { cout << "결과 : " << Result << endl; }
};
```

소멸자

- 객체 생성 및 소멸 시에 호출되는 함수
  - 객체 생성 → 생성자
  - 객체 소멸 → 소멸자
- 객체 생성 및 소멸 시점 (= 변수 생성 및 소멸 시점과 동일)
  - 지역 변수(객체)
    - 생성 : 해당 지역 수행 시
    - 소멸 : 해당 지역 탈출 시
  - 전역 변수(객체)
    - 생성 : 프로그램 시작 시
    - 소멸 : 프로그램 종료 시



소멸자

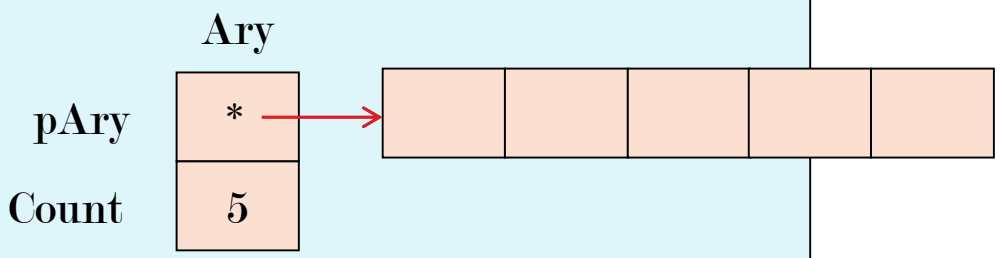
- 소멸자 만드는 방법
  - 소멸자 이름은 클래스 이름과 동일
    - 단, 소멸자 이름 앞에 '~' 문자 붙임
  - 반환형이 없음
  - 매개 변수가 없음
    - 소멸자는 단 하나만 존재

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

public :
	CPoint(int a, int b) { x = a; y = b; }
	~CPoint() { cout << "소멸자" << endl; }
};
```

소멸자

- 소멸자의 용도 (필요성)
  - 객체 소멸 시 처리해야 될 일이 있을 때
  - 예: 객체 생성과 동시에 메모리 동적 생성 → 객체 소멸시 해

```
class CArray {
private :
        int *pAry;
        int Count;

public :
        CArray(int Co) { Count = Co; pAry = new int[Count]; };
        ~CArray() { delete [] pAry; }
};

void main(void)
{
        CArray Ary(5);
}
```

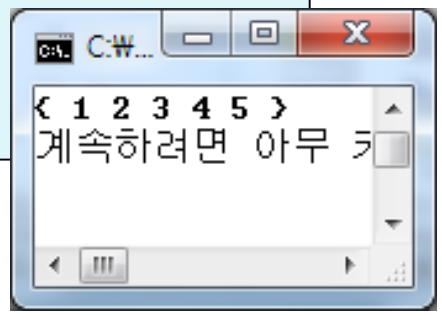


객체가 사라질 때 자동으로 호출

소멸자 : 연습 문제

- CArray 클래스를 main 함수와 같이 사용할 수 있도록 필요한 멤버 함수를 추가하라.
  - GetCount
  - SetValue
  - Print

```
void main(void)
{
	CArray Ary(5);

	for (int i = 0; i < Ary.GetCount(); i++)
		Ary.SetValue(i, i + 1);
	Ary.Print();
}
```



소멸자 : 연습 문제

- 프로그램 확인

```
class CArray {
private :
	int *pAry;
	int Count;

public :
	CArray(int Co) { Count = Co; pAry = new int[Count]; };
	~CArray() { delete [] pAry; }
	int GetCount() { return Count; }
	void SetValue(int index, int value) { pAry[index] = value; }
	void Print() {
		cout << "{ ";
		for (int i = 0; i < Count; i++)
			cout << pAry[i] << " ";
		cout << "}" << endl;
	}
};
```

생성자와 소멸자의 호출 순서

- 생성자의 호출 순서 : 객체 생성 순
- 소멸자의 호출 순서 : 객체 소멸 순 = 객체 생성의 역순
- 다음 프로그램의 실

```
CPoint P1(1, 1);
CPoint P2(2, 2);

void main(void)
{
        CPoint P3(3, 3);
        CPoint P4(4, 4);
}
```


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

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

디폴트 생성자와 디폴트 소멸자

- 다음 프로그램은 정상적으로 실행됨
  - 객체 생성 시 생성자, 소멸 시 소멸자가 반드시 한 번씩 수행됨!

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

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

void main(void)
{
	CPoint P1;
	P1.SetXY(3, 4);
	P1.Print();
}
```

디폴트 생성자와 디폴트 소멸자가 존재함!

디폴트 생성자와 디폴트 소멸자

- CPoint 클래스의 디폴트 생성자와 디폴트 소멸자
  - 생성자 : CPoint() { }
  - 소멸자 : ~CPoint() { }
- 디폴트 생성자와 디폴트 소멸자
  - 매개변수가 없고 특별히 하는 일도 없음
  - 생성자를 1개 이상 명시적으로 추가하면 디폴트 생성자는 사라짐
  -

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

public :
      CPoint(int a, int b) { x = a; y = b; }
};
```

```
void main(void)
{
      CPoint P1(3, 4);
      CPoint P2;
}
```

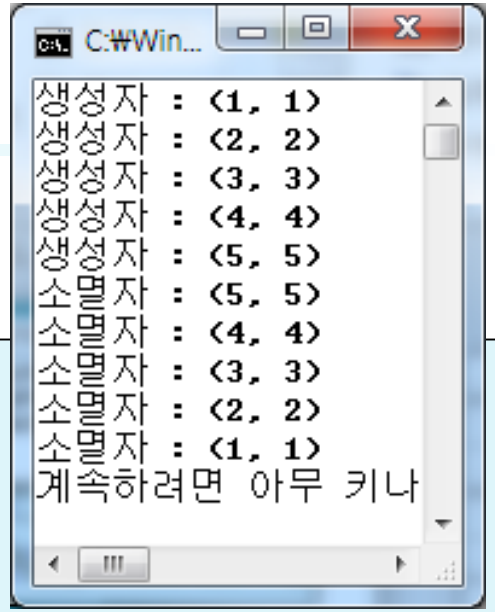
맞는 생성자 없음 → 에러

객체 배열과 생성자, 소멸자

- 배열 선언 시 초기화 방법
  - int ary[3] = { 1, 2, 3 };
  - Point ary[3] = { { 1, 1 }, { 2, 2 }, { 3, 3 } }; // 구조체
- 객체 배열 선언 시 초기화 방법
  - CPoint Ary[3];
    - 3개의 객체 모두 CPoint() 생성자 사용
  - CPoint Ary[3] = { CPoint(), CPoint(1), CPoint(3, 4) };
    - Ary[0] : CPoint(), Ary[1] : CPoint(int), Ary[2] : CPoint(int, int)
  - CPoint Ary[3] = { CPoint(1, 2) };
    - Ary[0] : CPoint(int, int), 나머지는 CPoint() 사용
- 객체 배열 소멸 순서
  - 마지막 원소 → 첫번째 원소 순

객체 배열과 생성자, 소멸자

- 다음 프로그램의 실행 결과는?



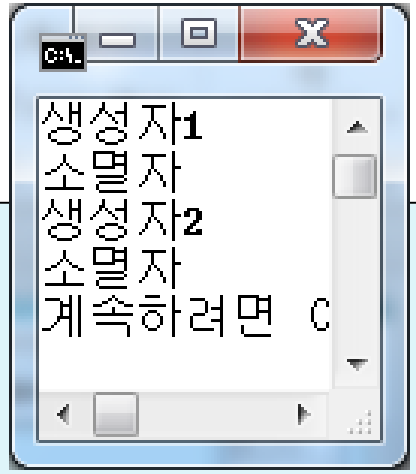

```
class CPoint {
private :
	int x, y;

public :
	CPoint(int a, int b) { x = a; y = b; cout << "생성자 : "; Print(); }
	~CPoint() { cout << "소멸자 : "; Print(); }
	void Print() { cout << "(" << x << ", " << y << ")" << endl; }
};

void main(void)
{
	CPoint Ary[5] = { CPoint(1, 1), CPoint(2, 2), CPoint(3, 3), CPoint(4, 4), CPoint(5, 5) };
}
```

메모리 동적 할당과 생성자, 소멸자

- 메모리 동적 할당(new)을 통한 객체 생성 시
  → 생성자 호출
- 메모리 해제 시
  → 소멸자 호출

```
void main(void)
{
        CPoint *Po;

        Po = new CPoint();
        delete Po;

        Po = new CPoint(3, 4);
        delete Po;
}
```

Po = new CPoint; // 동일




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

public :
        CPoint() { x = 0; y = 0; cout << "생성자1" << endl; }
        CPoint(int a, int b) { x = a; y = b; cout << "생성자2" << endl; }
        ~CPoint() { cout << "소멸자" << endl; }
};
```

멤버 초기화 구문

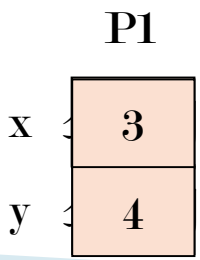
- 다음 두 변수(객체) 선언 및 초기화의 차이는?

```
void main(void)
{
      int a = 3;
}
```

변수 a가 메모리에 생기면서 바로 3으로 초기화

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

public :
      CPoint(int a, int b) { x = a; y = b; }
};

void main(void)
{
      CPoint P1(3, 4);
}
```

P1 객체가 메모리에 생기면서 쓰레기값으로 초기화
→ 생성자가 실행되면서 3, 4로 초기화



멤버 초기화 구문

- 멤버 초기화 구문
  - 객체 생성 시 멤버 변수가 메모리에 생기면서 바로 특정값으로 초기화

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

public :
      CPoint(int a, int b) : x(a), y(b) { }
};

void main(void)
{
      CPoint P1(3, 4);
}
```

멤버 초기화 구문