3-1. 생성자와 소멸자

♲ 생성자란?

생성자란 객체를 즉시 사용할 수 있는 상태로 초기화 시켜주는 클래스의 멤버함수이다. 따라서 생성자를 사용하면 생성된 객체의 상태를 일일이 신경을 써 초기화 하지 않고도 사용 할 수 있다. 단, 생성자가 하는 일이 너무 많으면 오히려 코드에 대한 이해가 복잡해 질수 있음으로 꼭 필요한 작업만 생성자 함수를 통해 수행하는 것이 좋다.

1. 클래스로 만들어지는 객체를 초기화 해주는 멤버함수이다.
   (객체의 초기화란 멤버변수의 초기화를 의미한다.)
2. 생성자 함수는 클래스로 객체를 선언할 때 자동적으로 호출된다.
3. 생성자 함수는 사용자가 직접 호출해 사용할 수 없다.
   (함수의 호출은 명시적이지 않다.)
4. 생성자 함수의 이름은 클래스의 이름과 동일하게 작성해야 한다.
5. 생성자 함수는 리턴 값이 void인 함수이어야 하며 void를 생략하여 함수를 선언하고 정의 한다.
   (생성자 함수는 리턴 값을 가질 수 없다.)
6. 생성자 함수는 객체가 생성될 때 마다 호출되어 생성된 객체의 멤버변수를 초기화 시킨다. 따라서 전역객체에 대한 생성자 함수는 한번만 실행되며 지역객체에 대한 생성자 함수는 지역객체가 선언될때 마다 실행된다.
7. 생성자는 가상함수로 선언 될 수 없다

♲ 생성자 함수의 사용방법

1. 생성자 함수를 사용하지 않는 객체의 초기화 방법

설명 : 객체의 멤버 변수는 private(전용) 멤버이기 때문에 직접적인 초기화가 불가능 하다. 이 문제를 해결하려면 멤버변수를 public으로 선언하거나 초기화를 위한 멤버함수를 통해 초기화를 해야 한다. 하지만 멤버변수를 public으로 선언하는 방법은 객체지향 언어의 데이터 은닉성에 위배됨으로 사용할 수 없고 별도의 멤버함수를 통한 초기화는 번거롭다.

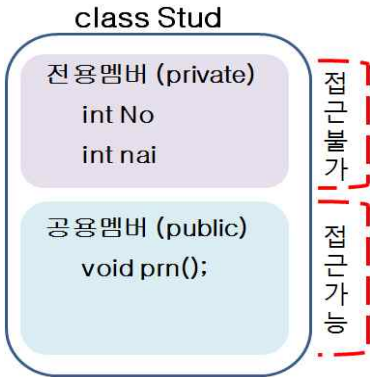
예제 1-1 : 클래스로 객체를 만들면서 private인 멤버변수를 직접 초기화 한 예제 (에러발생)

```
#include <iostream.h>
class Stud  {
private :
    int  No;
    int  nai
public :
    void prn()  {
        cout << "번호 : " << No  << "Wt" << "나이 : " << nai << endl ;
    }
};

void main( ) {
    Stud ob1 = { 1, 25 };     // 에러 발생 : 멤버변수가 private임으로 값을 직접 넣을 수 없다.
    ob1.prn();
}
```

실 행 결 과

```
error C2552: 'ob1' : non-aggregates cannot be initialized with initializer list
```

예제 1-1 해설 : 선언한 class Stud로 객체 ob1을 만들면서 멤버 변수의 초기화를 위해 struct과 같은 방법으로 초기값을 부여 하였지만 에러가 발생하게 된다. 에러가 발생하는 이유는 멤버변수인 No와 nai가 private(전용) 멤버 변수임으로 외부에서의 접근이 불가능하기 때문이다.



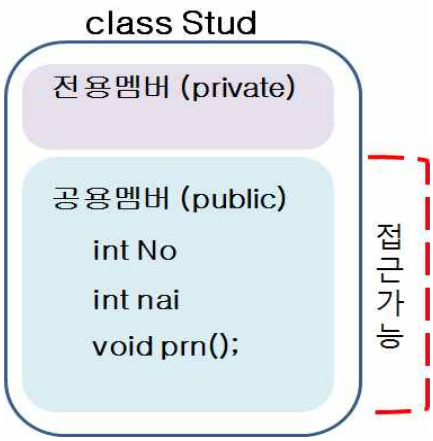
예제 1-2 : 클래스로 객체를 만들면서 public인 멤버변수를 직접 초기화 한 예제 (에러는 발생하지 않지만 이렇게 하면 객체지향 프로그래밍이 아니다.)

```
#include <iostream.h>
class Stud  {
public :
    int  No;
    int  nai;
    void prn()  {
        cout << "번호 : " << No  << "Wt" << "나이 : " << nai << endl ;
    }
};

void main( ) {
    Stud ob1 = { 1, 25 };     // 멤버변수가 public임으로 값을 직접 넣을 수 있다.
    ob1.prn();                // 초기화된 값이 출력된다.
}
```

실 행 결 과

```
번호 : 1        나이 : 25
Press any key to continue
```

예제 1-2 해설 : class Stud를 선언 할 때 멤버변수인 No와 nai를 public(공용) 멤버로 정의했기 때문에 외부에서 해당 멤버변수에 대한 직접적인 접근이 가능해 진다. 따라서 struct과 같은 방법으로 객체를 만들면서 초기값을 부여 한다면 부여한 초기값으로 멤버 변수가 초기화 된 객체가 생성되는 것을 확인 할 수 있다. 하지만 이 방법은 객체지향언어의 데이터 은닉성에 위배됨으로 사용하면 안 된다.



예제 1-3 : 클래스로 객체를 만들면서 private인 멤버변수를 초기화하기 위해 별도의 멤버함수를 만든 예제 (올바른 객체의 초기화 방법)

```
#include <iostream.h>
class Stud  {
public :
    int  No;
    int  nai;
    void set()  {
        No = 1;
        nai = 25;
    }
    void prn()  {
        cout << "번호 : " << No  << "Wt" << "나이 : " << nai << endl ;
    }
};

void main( ) {
    Stud ob1;
    ob1.set();              // 객체를 만든 뒤 멤버변수를 초기화 할 함수를 별도로 호출함
    ob1.prn();              // 초기화된 값이 출력된다.
}
```

실 행 결 과

```
번호 : 1        나이 : 25
Press any key to continue
```

예제 1-3 해설 : 예제 1-1 과 1-2와는 달리 문제없이 멤버변수를 초기화 시키려면 이와 같인 별도의 초기화 함수를 만들어 멤버 변수를 초기화 하면 된다. 단, 이 방법은 객체를 만들 때 마다 초기화 함수를 호출해 주어야 함으로 class를 사용하는 사용자가 번거롭게 된다.

2. 생성자 함수의 선언 및 정의 방법

설명 : 생성자 함수도 멤버 함수임으로 클래스 내에 정의 하면 된다. 또한 다른 멤버함수와 같이 클래스 내부에서는 함수의 선언만 하고 클래스 외부에서 정의 하는 것도 가능하다.

```
// 클래스 내부에 생성자 함수를 정의 하는 방법
class 클레스명{
      :
public:
      클래스명()  {          // 생성자 함수의 정의. return type을 쓰지 않는다.
          :                 // 이곳에 생성자 함수의 내용을 기재 한다.
      }
      :                     // 클래스의 다른 내용
};
```

```
// 클래스 내부에는 생성자 함수에 대한 선언을 하고 클래스 외부에 생성자 함수를 정의하는 방법
class 클레스명{
          :
      public:
             클래스명();            // 생성자 함수의 선언. return type을 쓰지 않는다.
          :
};

클래스명::클래스명() {
          :                         // 이곳에 생성자 함수의 내용을 기재 한다.
}
```

예제 2-1 : 생성자 함수를 클래스 내부에 정의하는 방법

```
#include <iostream.h>
class Stud  {
private :
    int  No;
    int  nai;
public :
    Stud()  {
        No = 1;
        nai = 25;
    }
    void prn()  {
        cout << "번호 : " << No  << "Wt" << "나이 : " << nai << endl ;
    }
};

void main( ) {
    Stud ob1;                       // 객체를 만들면 자동으로 생성자 함수가 실행된다.
    ob1.prn();                      // 초기화된 값이 출력된다.
}
```

실 행 결 과

```
번호 : 1        나이 : 25
Press any key to continue
```

예제 2-1 해설 : class Stud안에 생성자 함수를 만들어 두었음으로 객체를 만들면 생성자 함수가 자동으로 실행되어 멤버변수가 초기화된다. 위의 예를 보면 main 함수에서 객체를 만든 후 즉시 출력 했지만 멤버변수인 No와 nai가 1과 25로 출력 되는 것을 확인 할 수 있다. 즉, 객체를 만들면 자동으로 생성자 함수가 실행되어 객체를 초기화 되었음을 알 수 있다.

예제 2-2 : 생성자 함수를 클래스 내부에 선언하고 클래스 외부에서 정의하는 방법

```
#include <iostream.h>
class Stud  {
private :
    int  No;
    int  nai;
public :
    Stud();                      // 클래스 내부에는 생성자 함수에 대한 선언만 한다.
    void prn()  {
        cout << "번호 : " << No  << "Wt" << "나이 : " << nai << endl ;
    }
};

Stud :: Stud()  {                // 클래스 외부에서 생성자 함수 정의하기
    No = 1;
    nai = 25;
}

void main( ) {
    Stud ob1;                   // 객체를 만들면 자동으로 생성자 함수가 실행된다.
    ob1.prn();                  // 초기화된 값이 출력된다.
}
```

실 행 결 과

```
번호 : 1        나이 : 25
Press any key to continue
```

예제 2-2 해설 : 예제 2-1과 동일하지만 생성자 함수를 class 외부에 정의하였다. class 외부에 생성자 함수를 정의 하는 방법은 일반 멤버 함수와 동일하지만 생성자 함수는 return type이 없음으로 함수가 정의된 모양이 독특해 보인다.

3. 생성자 함수의 사용 예

설명 : 생성자 함수는 객체를 선언과 동시에 사용하기 위해 사용한다. 생성자 함수를 이용하면 별도의 함수 호출 없이 객체를 즉시 원하는 형태로 초기화 할 수 있으며 생성자 함수가 매개변수를 받아가게 하면 원하는 값으로 객체의 멤버 변수를 초기화 할 수 있다.

```
// 매개변수를 받아 갈 수 있는 생성자 함수를 클래스 내부에 정의 하는 방법
class 클레스명 {
        :
public:
        클래스명(매개변수의자료형 매개변수명, ...)  {
            :
        }
        :                       // 클래스의 다른 내용
};
```

```
// 매개변수를 받아 갈 수 있는 생성자 함수를 클래스 내부에 선언하고 클래스 외부에서 정의 하는 방법
class 클레스명{
          :
      public:
             클래스명((매개변수의자료형 매개변수명, ...);
          :
};

클래스명::클래스명((매개변수의자료형 매개변수명, ...) {
          :                          // 이곳에 생성자 함수의 내용을 기재 한다.
}
```

예제 3-1 : 생성자 함수를 이용한 객체의 자동초기화 - 1

```
#include <iostream.h>
#include <string.h>

class Stud {
private :
       int  No;
       char Name[10];
public :
       Stud();
       void prn();
};

Stud::Stud() {
       No = 0;
       strcpy(Name, "noboby");
}

void Stud::prn() {
       cout << "번호 : " << No << "₩t" << "이름 : " << Name << endl ;
}

void main( )  {
       Stud ob1;
       ob1.prn();
}
```

실 행 결 과

```
번호 : 0        이름 : nobody
Press any key to continue
```

예제 3-2 : 생성자 함수를 이용한 객체의 자동초기화 - 2

```
#include <iostream.h>
#include <string.h>

class Stud {
private :
       int  No;
       char *name;
public :
       Stud( );
       void prn( );
       void input( );
       void del( );
};

Stud::Stud( ) {
       No = 0;
       name = new char[7];
       strcpy(name, "noboby");
}

void Stud::prn( ) {
       cout << "번호 : " << No << "₩t" << "이름 : " << name << endl ;
}

void Stud::del ( ) {
       delete name;
}

void Stud::input ( ) {
       cout << "번호를 입력해 주세요 : " ;
       cin >> No;
       cout << "이름을 입력해 주세요 : " ;
       cin >> name;
       cout << "입력 결과" << endl;
       prn( );
}

void main( )  {
       Stud ob1;
       ob1.prn( );
       ob1.input( );
       ob1.del( );
}
```

실 행 결 과

```
번호 : 0        이름 : noboby
번호를 입력해 주세요 : 1
이름을 입력해 주세요 : 홍길동
입력 결과
번호 : 1        이름 : 홍길동
Press any key to continue
```

예제 3-3 : 매개변수를 받아오는 생성자 함수 - 1

```
#include <iostream.h>
class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud(int x, int y)  {
        no = x;
        nai = y;
    }
    void prn()  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};
void main( ) {
    Stud ob1(1,25);                 // 객체를 만들면서 매개변수를 쓰면 생성자 함수에 전달된다.
    ob1.prn();                      // 초기화된 값이 출력된다.
    Stud ob2(2,23);                 // 객체를 만들면서 매개변수를 쓰면 생성자 함수에 전달된다.
    ob2.prn();                      // 초기화된 값이 출력된다.
}
```

실 행 결 과

```
번호 : 1        나이 : 25
번호 : 2        나이 : 23
Press any key to continue
```

예제 3-4 : 매개변수를 받아오는 생성자 함수 - 2

```
#include <iostream.h>
#include <string.h>

class Stud {
private :
       int  no;
       char Name[10];
public :
       Stud(int x, char *y);
       void prn();
};

Stud::Stud(int x, char *y) {
       no = x;
       strcpy(Name, y);
}

void Stud::prn() {
       cout << "번호 : " << no << "₩t" << "이름 : " << Name << endl ;
}

void main( )  {
       Stud ob1(1, "홍길동");
       ob1.prn();
}
```

실 행 결 과

```
번호 : 1        이름 : 홍길동
Press any key to continue
```

예제 3-5 : 매개변수를 받아오는 생성자 함수 - 3

```
#include <iostream.h>
#include <string.h>

class Stud {
private :
        int  No;
        char *name;
public :
        Stud( int x, char *y );
        void prn( );
        void input( );
        void del( );
};

Stud::Stud( int x, char *y  ) {
        No = x;
        name = new char[strlen(y)+1];
        strcpy(name, y);
}

void Stud::prn( ) {
        cout << "번호 : " << No << "Wt" << "이름 : " << name << endl ;
}

void Stud::del ( ) {
        delete name;
}

void Stud::input ( ) {
        cout << "번호를 입력해 주세요 : " ;
        cin >> No;
        cout << "이름을 입력해 주세요 : " ;
        cin >> name;
        cout << "입력 결과" << endl;
        prn( );
}

void main( )  {
        Stud ob1(0, "독수리오형제");
        ob1.prn( );
        ob1.input( );
        ob1.del( );
}
```

실 행 결 과

```
번호 : 0        이름 : 독수리오형제
번호를 입력해 주세요 : 1
이름을 입력해 주세요 : 멋쟁이삼형제
입력 결과
번호 : 1        이름 : 멋쟁이삼형제
Press any key to continue
```

예제 3-6 : 객체를 배열로 만들 경우 생성자 함수에 매개변수를 건네주는 방법

```
#include <iostream.h>

class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud( int x, int y )  {
        no = x;
        nai = y;
    }
    void prn( )  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};
void main( ) {
    Stud arr[3] = { Stud(1, 15 ), Stud( 2, 25 ), Stud( 3, 35 ) };
    arr[0].prn( );
    arr[1].prn( );
    arr[2].prn( );
}
```

실 행 결 과

```
번호 : 1        나이 : 15
번호 : 2        나이 : 25
번호 : 3        나이 : 35
Press any key to continue
```

예제 3-6 : 객체를 배열로 만들 경우 생성자 함수에 매개변수를 건네주는 방법 (오류 발생)

```
#include <iostream.h>

class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud( int x, int y )  {
        no = x;
        nai = y;
    }
    void prn( )  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};
void main( ) {
    Stud arr[3] = { Stud(1, 15 ), Stud( 2, 25 ) };
    arr[0].prn( );
    arr[1].prn( );
    arr[2].prn( );
}
```

실 행 결 과

```
error C2073: 'arr' : partially initialized array requires a default constructor
```

예제 3-6 : 매개변수를 필요로 하는 생성자 함수만 존재하는 클래스로 만든 객체를 일반 함수에 매개변수로 주는 방법 **(해당 내용은 복사생성자 챕터를 공부하여야 이해 할 수 있다.)**

```
#include <iostream.h>

class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud( int x, int y )  {
        no = x;
        nai = y;
    }
    void prn( )  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};

void func( Stud temp ) {
    cout << " -- func 함수에서 출력합니다. -- " << endl;
    temp.prn( );
}

void main( ) {
    Stud ob1( 1, 15 );
    func(ob1);
}
```

실 행 결 과

```
 - func 함수에서 출력합니다. --
번호 : 1        나이 : 15
Press any key to continue
```

4. 생성자 함수의 오버로딩

설명 : 생성자 함수도 class안의 멤버 함수임으로 함수의 오버로딩과 같이 오버로딩 하는 것이 가능하다. 생성자 함수를 오버로딩 한다면 객체를 선언 할 때 건네준 매개변수 값에 따라 어떠한 생성자 함수가 호출 될지 결정된다. 오버로딩 함수에는 수적 제한이 없으며 매개변수가 없는 생성자 함수를 default 생성자 함수라 한다.

예제 4-1 : 매개변수를 받아가지 않는 default 생성자 함수와 매개변수를 받아가는 생성자 함수를 오버로딩한 예제 - 1

```
#include <iostream.h>

class Point {
	int xp, yp;
public :
	Point( )  { xp = yp = 0; }			// 디폴트 생성자 함수
	Point(int x, int y) {				// 매개변수를 받아오는 생성자 함수
		xp = x;
		yp = y;
	}
	int get_x()  { return xp; }
	int get_y()  { return yp; }
};

void main()
{
	Point p1;					// default 생성자 함수가 호출된다.
	Point p2(10, 20);				// 매개변수를 받아가는 생성자 함수가 호출된다.
	cout << "== 객체 p1 ==" << endl;
	cout << "x 좌표 : " << p1.get_x() << ", y 좌표 : " << p1.get_y() << endl;
	cout << endl;
	cout << "== 객체 p2 ==" << endl;
	cout << "x 좌표 : " << p2.get_x() << ", y 좌표 : " << p2.get_y() << endl;
}
```

실 행 결 과

```
== 객체 p1 ==
x 좌표 : 0, y 좌표 : 0

== 객체 p2 ==
x 좌표 : 10, y 좌표 : 20
Press any key to continue
```

예제 4-2 : 매개변수를 받아가지 않는 default 생성자 함수와 매개변수를 받아가는 생성자 함수를 오버로딩 한 예제 - 2

```
#include <iostream.h>

class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud( )  {
        cout << "default 생성자가 실행됩니다." << endl;
        no = 0;
        nai = 0;
    }
    Stud( int x, int y )  {
        cout << "매개변수로 " << x << "와 " << y << "를 받아온 생성자가 실행됩니다." << endl;
        no = x;
        nai = y;
    }
    void prn( )  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};

void main( ) {
    Stud arr[3] = { Stud( ), Stud( 2, 25 ), Stud( 3, 35 ) };
    arr[0].prn( );
    arr[1].prn( );
    arr[2].prn( );
}
```

실 행 결 과

```
default 생성자가 실행됩니다.
매개변수로 2와 25를 받아온 생성자가 실행됩니다.
매개변수로 3와 35를 받아온 생성자가 실행됩니다.
번호 : 0        나이 : 0
번호 : 2        나이 : 25
번호 : 3        나이 : 35
Press any key to continue
```

예제 4-2 : 생성자 함수를 3개 만들어 오버로딩 한 예제

```
#include <iostream.h>

class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud( )  {
        cout << "default 생성자가 실행됩니다." << endl;
        no = 0;
        nai = 0;
    }
    Stud( int x )  {
        cout << "매개변수로 " << x << "를 받아온 생성자가 실행됩니다." << endl;
        no = x;
        nai = 0;
    }
    Stud( int x, int y )  {
        cout << "매개변수로 " << x << "과 " << y << "를 받아온 생성자가 실행됩니다." << endl;
        no = x;
        nai = y;
    }
    void prn( )  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};

void main( ) {
    Stud arr[3] = { Stud( ), Stud( 2 ), Stud( 3, 35 ) };
    arr[0].prn( );
    arr[1].prn( );
    arr[2].prn( );
}
```

실 행 결 과

```
default 생성자가 실행됩니다.
매개변수로 2를 받아온 생성자가 실행됩니다.
매개변수로 3과 35를 받아온 생성자가 실행됩니다.
번호 : 0        나이 : 0
번호 : 2        나이 : 0
번호 : 3        나이 : 35
Press any key to continue
```

♲ 소멸자란?

소멸자란 객체의 사용이 끝난 후 객체에 대한 뒤처리 작업 등을 자동으로 수행 할 수 있도록 해주는 class의 멤버함수 이다. 소멸자를 사용하면 생성된 객체에 대한 뒤처리를 신경쓰지 않고 편하게 프로그래밍을 할 수 있다. 단, 생성자와 마찬가지로 소멸자 함수가 하는 작업이 너무 많아지면 프로그램의 이해가 어려워지고 오류 발생 위험이 높아지는 등의 문제가 있음으로 꼭 필요한 내용만 소멸자 함수에 포함시키는 것이 좋다.

1. 클래스로 만들어진 객체에 대한 뒤처리를 해주는 멤버함수이다.
   (뒤처리 작업은 주로 동적메모리할당 해제를 의미한다.)
2. 소멸자 함수는 사용중인 객체가 소멸되는 시점에 자동으로 호출된다.
3. 소멸자 함수는 사용자가 직접 호출해 사용할 수 없다.
   (함수의 호출은 명시적이지 않다.)
4. 소멸자 함수의 이름은 클래스의 이름과 동일하게 작성해야 한다. 단, 생성자 함수와 구분하기 위해 함수명 앞쪽에 ~ 연산자를 붙인다. (~는 틸드(tilde) 또는 bit not 으로 읽으면 된다.)
5. 소멸자 함수는 리턴 값이 void인 함수이어야 하며 void를 생략하여 함수를 선언하고 정의 한다.
   (소멸자 함수는 리턴 값을 가질 수 없다.)
6. 소멸자 함수는 매개변수를 받아올 수 없다. 따라서 소멸자에 대한 오버로딩이 불가함으로 class 하나에 오직 한 개의 소멸자 함수만 존재해야 한다.
7. 소멸자 함수는 객체가 소멸될 때 마다 호출되어 소멸된 객체에 대한 뒤처리 작업을 한다. 따라서 전역객체에 대한 소멸자 함수는 프로그램 종료 시 한번만 실행되며 지역객체에 대한 소멸자 함수는 지역객체가 소멸될 때 마다 실행된다.

♲ 소멸자 함수의 사용방법

5. 소멸자 함수를 사용하지 않는 경우 객체의 뒤처리 방법

설명 : 소멸자 함수를 사용하지 않는 경우 생성한 객체가 벌여 놓은 일에 대한 뒤처리 등의 작업을 하려면 직접 해당 작업을 하는 멤버함수를 만들어 사용자가 호출해 주어야 하지만 사용자가 일일이 객체가 사라지는 곳을 찾아가 해당 멤버함수를 호출하는 것은 매우 불편하다.

예제 5-1 : 소멸자를 사용하지 않고 직접 객체의 뒤처리 작업을 하는 예제

```
#include <iostream.h>

class exam  {
private :
    int size;
    int *ip;
public :
    exam( )  {
        cout << "만들려는 배열의 크기를 입력해 주세요 : ";
        cin >> size ;
        ip = new int[size];
    }
    void input( ) {
        int i;
        cout << "연달아 " << size << "개의 정수를 입력해 주세요 " << endl;
        for (i = 0; i < size ; i ++) {
            cin >> ip[i];
        }
    }
```

```
    void print( )  {
        int i;
        cout << "ip = " ;
        for (i = 0; i < size ; i ++) {
            cout << "[" << ip[i] << "]";
        }
        cout << endl;
    }
    void del( ) {
        cout << "메모리 할당을 해제 합니다." << endl;
        delete ip;
    }
};

void func( ) {
    exam ob2;
    ob2.input( );
    cout << "ob2의값 ==> ";
    ob2.print( );
    ob2.del( );
}

void main( ) {
    exam ob1;
    ob1.input( );
    cout << "ob1의값 ==> ";
    ob1.print( );
    func( );
    {                                   // 내부블럭 시적
        exam ob3;
        ob3.input( );
        cout << "ob3의값 ==> ";
        ob3.print( );
        ob3.del( );
    }
    ob1.del( );
}
```

실 행 결 과

```
만들려는 배열의 크기를 입력해 주세요 : 1
연달아 1개의 정수를 입력해 주세요
10
ob1의값 ==> ip = [10]
만들려는 배열의 크기를 입력해 주세요 : 2
연달아 2개의 정수를 입력해 주세요
20 30
ob2의값 ==> ip = [20][30]
메모리 할당을 해제 합니다.
만들려는 배열의 크기를 입력해 주세요 : 3
연달아 3개의 정수를 입력해 주세요
40 50 60
ob3의값 ==> ip = [40][50][60]
메모리 할당을 해제 합니다.
메모리 할당을 해제 합니다.
Press any key to continue
```

6. 소멸자 함수의 선언 및 정의 방법

설명 : 소멸자 함수도 멤버 함수임으로 클래스 내에 정의 하면 된다. 또한 다른 멤버함수와 같이 클래스 내부에서는 함수의 선언만 하고 클래스 외부에서 정의 하는 것도 가능하다.

```
// 클래스 내부에 소멸자 함수를 정의 하는 방법
class 클레스명{
        :
public:
        ~클래스명()  {       // 소멸자 함수의 정의. return type을 쓰지 않고 함수명 앞에 ~를 붙인다.
             :               // 이곳에 소멸자 함수의 내용을 기재 한다.
        }
        :                    // 클래스의 다른 내용
};
```

```
// 클래스 내부에는 소멸자 함수에 대한 선언을 하고 클래스 외부에 소멸자 함수를 정의하는 방법
class 클레스명{
        :
public:
        ~클래스명();         // 소멸자 함수의 선언. return type을 쓰지 않고 함수명 앞에 ~를 붙인다.
        :
};

클래스명::~클래스명() {
        :                    // 이곳에 소멸자 함수의 내용을 기재 한다.
}
```

예제 6-1 : 소멸자 함수를 클래스 내부에 정의하는 방법

```
#include <iostream.h>
class Stud  {
private :
    int  No;
    int  nai;
public :
    Stud()  {   No = nai = 0;    }
    ~Stud()  {
        cout << "소멸자 함수가 실행됩니다." << endl;
    }
    void prn()  {
        cout << "번호 : " << No  << "Wt" << "나이 : " << nai << endl ;
    }
};

void main( ) {
    Stud ob1;
    ob1.prn();
}
```

실 행 결 과

```
번호 : 0        나이 : 0
소멸자 함수가 실행됩니다.
Press any key to continue
```

예제 6-1 해설 : class Stud 안에 소멸자 함수를 만들어 두었음으로 객체가 사라지는 main 함수 종료 시점에 소멸자 함수가 자동으로 실행된 것을 확인 할 수 있다. 소멸자 함수는 생성자 함수와 구분하기 위하여 함수명 앞쪽에 ~ 기호가 붙은 것을 확인 할 수 있다.

예제 6-2 : 소멸자 함수를 클래스 내부에 선언하고 클래스 외부에서 정의하는 방법

```
#include <iostream.h>
class Stud  {
private :
    int  No;
    int  nai;
public :
    Stud()  {   No = nai = 0;    }
    ~Stud();
    void prn()  {
        cout << "번호 : " << No  << "Wt" << "나이 : " << nai << endl ;
    }
};

Stud :: ~Stud()  {
    cout << "소멸자 함수가 실행됩니다." << endl;
}

void main( ) {
    Stud ob1;
    ob1.prn();
}
```

실 행 결 과

```
번호 : 0        나이 : 0
소멸자 함수가 실행됩니다.
Press any key to continue
```

예제 6-2 해설 : 예제 6-1과 동일하지만 소멸자 함수를 class 외부에 정의하였다. class 외부에 소멸자 함수를 정의 하는 방법은 일반 멤버 함수와 동일하지만 소멸자 함수는 return type이 없음으로 함수가 정의된 모양이 독특해 보인다.

7. 소멸자 함수의 사용 예

설명 : 소멸자 함수는 객체에 대한 뒤처리를 하기 위해 사용하며 주로 동적 메모리 할당을 자동으로 해제하기 위해 사용한다.

예제 7-1 : 예제 5-1번에 소멸자 함수를 추가하여 동적메모리할당 해제를 한 예제

```
#include <iostream.h>

class exam  {
private :
    int size;
    int *ip;
public :
    exam( )  {
        cout << "만들려는 배열의 크기를 입력해 주세요 : ";
        cin >> size ;
        ip = new int[size];
    }
    void input( ) {
        int i;
        cout << "연달아 " << size << "개의 정수를 입력해 주세요 " << endl;
        for (i = 0; i < size ; i ++) {
            cin >> ip[i];
        }
    }
    void print( )  {
        int i;
        cout << "ip = " ;
        for (i = 0; i < size ; i ++) {
            cout << "[" << ip[i] << "]";
        }
        cout << endl;
    }
    ~exam( ) {
        cout << "메모리 할당을 해제 합니다." << endl;
        delete ip;
    }
};

void func( ) {
    exam ob2;
    ob2.input( );
    cout << "ob2의값 ==> ";
    ob2.print( );
}

void main( ) {
    exam ob1;
    ob1.input( );
    cout << "ob1의값 ==> ";
    ob1.print( );
    func( );
    {                                   // 내부블럭 시적
        exam ob3;
        ob3.input( );
        cout << "ob3의값 ==> ";
        ob3.print( );
    }
}
```

실 행 결 과

```
만들려는 배열의 크기를 입력해 주세요 : 1
연달아 1개의 정수를 입력해 주세요
10
ob1의값 ==> ip = [10]
만들려는 배열의 크기를 입력해 주세요 : 2
연달아 2개의 정수를 입력해 주세요
20 30
ob2의값 ==> ip = [20][30]
메모리 할당을 해제 합니다.
만들려는 배열의 크기를 입력해 주세요 : 3
연달아 3개의 정수를 입력해 주세요
40 50 60
ob3의값 ==> ip = [40][50][60]
메모리 할당을 해제 합니다.
메모리 할당을 해제 합니다.
Press any key to continue
```

예제 7-1 해설 : 예제 5-1과 동일하지만 소멸자 함수 이용해 동적 메모리 할당 해제를 하였음으로 직접 멤버 함수를 호출할 필요 없이 객체 소멸시 자동으로 호출되는 소멸자 함수에 의해 동적메모리할당이 해제 된다.

예제 7-2 : 소멸자 함수가 호출 호출되는 순서

```
#include <iostream.h>
#include <string.h>

class Stud {
private :
               int  No;
               char Name[10];
public :
       Stud(int n, char na[ ]);
       ~Stud( );
       void prn( );
 };

Stud::Stud(int n, char na[]) {
       cout << "Name이 " << na << " 인 객체의 생성자 함수" << endl;
       No = n;
       strcpy(Name, na);
}

Stud::~Stud( ) {
       cout << "Name이 " << Name << " 인 객체의 소멸자 함수" << endl;
}

void Stud::prn( ) {
       cout << "번호 : " << No << "Wt" << "이름 : " << Name << endl ;
}

void main( ) {
       Stud std[3]={ Stud( 1,"김연아"), Stud( 2, "박지성"), Stud( 3, "유재석") };
       for(int i=0; i<3; i++)
               std[i].prn();
}
```

실 행 결 과

```
name이 김연아 인 객체의 생성자 함수
name이 박지성 인 객체의 생성자 함수
name이 유재석 인 객체의 생성자 함수
번호 : 1        이름 : 김연아
번호 : 2        이름 : 박지성
번호 : 3        이름 : 유재석
name이 유재석 인 객체의 소멸자 함수
name이 박지성 인 객체의 소멸자 함수
name이 김연아 인 객체의 소멸자 함수
Press any key to continue
```

예제 7-2 해설 : 소멸자는 생성자가 실행된 것과 역순으로 실행되는 것을 확인 할 수 있다. 즉, 동시에 여러개의 객체가 소멸될 경우 가장 마지막에 생성된 객체의 소멸자가 먼저 실행된다. (stack구조)

예제 7-3 : 동적메모리 할당한 곳에 있는 문자열을 결합해주는 함수가 있는 class 예제

```
#include <iostream.h>
#include <string.h>

class String {
        char *str;
        int len;
public:
        String(char *);                         // 생성자
        ~String();                              // 소멸자
        void cat(String &);                     // 매개변수가 객체인 멤버함수
        void prn();
};

String::String(char *p) {
        len = strlen(p);
        str = new char[len+1];
        strcpy(str, p);
}

String::~String() {
        cout << "Deleting str...Wn";
        delete str;  // 메모리 해제
}

void String::cat(String &s) {
        len = len + s.len;
        char *tmp = new char[len+1];            // 메모리 동적할당
        strcpy(tmp, str);                       //문자열 복사
        strcat(tmp, s.str);                     // 문자열 연결
        delete str;                             // 옛 문자열 해제
        str = tmp;
}
```

```
void String::prn() {
	cout << "문자열 : " << str << "(" << len << ")" << endl;
}

int main() {
	String str1("대한민국"), str2("파이팅");
	str1.prn();
	str2.prn();
	str1.cat(str2);
	cout << "Wn====문자열 연결==========Wn";
	str1.prn();
	return 0;
}
```

실 행 결 과

```
문자열 : korea team (11)
문자열 : fighting(8)

====문자열 연결==========
문자열 : korea team fighting(19)
Deleting str...
Deleting str...
Press any key to continue
```

♲ 복사생성자란?

복사생성자란 객체를 만들면서 생성자 함수에 매개변수로 기존에 생성한 객체를 건네줄 경우 컴파일러가 자동으로 생성해주는 생성자 함수이다. 원래 생성자 함수에 어떠한 매개변수를 건네주려면 해당 매개변수를 받아올 수 있는 생성자 함수를 사용자가 직접 만들어야 하지만 객체가 매개변수로 넘어갈 경우에는 사용자가 생성자 함수를 만들지 않아도 컴파일러가 자동으로 복사생성자를 만들어 객체를 매개변수로 받아올 수 있게 해준다.

1. 컴파일러에 의해 자동으로 생성되는 생성자 함수이다.
2. 복사생성자는 객체를 생성자의 매개변수로 받아갈 수 있게 해준다.
3. 필요할 경우 사용자가 직접 복사 생성자를 만들어 사용 할 수도 있다.

♲ 복사생성자의 사용방법

8. 컴파일러가 자동으로 생성해 주는 복사 생성자

설명 : 복사생성자는 사용자가 직접 정의하지 않아도 생성자 함수에 매개변수로 객체를 건네주면 컴파일러가 자동으로 생성한다.

예제 8-1 : 컴파일러가 자동으로 생성한 복사 생성자 - 1

```
#include <iostream.h>

class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud( )  {
        no = nai = 0;
    }
    void print( )  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};
void main( ) {
    Stud ob;
    ob.print( ) ;

    Stud pb(ob);
    pb.print( ) ;
}
```

실 행 결 과

```
번호 : 0        나이 : 0
번호 : 0        나이 : 0
Press any key to continue
```

예제 8-2 : 컴파일러가 자동으로 생성한 복사 생성자 - 2

```
#include <iostream.h>

class Stud  {
private :
    int  no;
    int  nai;
public :
    Stud( int x, int y )  {
        no = x, nai = y;
    }
    void print( )  {
        cout << "번호 : " << no  << "₩t" << "나이 : " << nai << endl ;
    }
};
void main( ) {
    Stud arr[3] = { Stud( 1, 15) , Stud( 2, 25), Stud( arr[1] ) };
    arr[0].print( );
    arr[1].print( );
    arr[2].print( );
}
```

실 행 결 과

```
번호 : 1        나이 : 15
번호 : 2        나이 : 25
번호 : 2        나이 : 25
Press any key to continue
```

예제 8-3 : 컴파일러가 자동으로 생성한 복사 생성자 - 3

```
#include<iostream.h>
#include<string.h>

class Stud {
private :
                int No;
                char Name[10];
public :
        Stud();
        Stud(int n, char na[]);
        ~Stud();
        void prn();
};

Stud::Stud() {
        cout << "디폴트 생성자 호출₩n";
        No = 0;
        strcpy(Name, "없어요");
}

Stud::Stud(int n, char na[]) {
        cout << "매개변수를 받아온 생성자 호출₩n";
        No = n;
        strcpy(Name, na);
}

Stud::~Stud() {
        cout << "소멸자 호출₩n";
}

void Stud::prn() {
        cout << "번호 : " << No << "₩t" << "이름 : " << Name << endl ;
}

void main( ) {
        Stud std[3]={Stud(), Stud( 1, "김연아"), Stud( std[1]) };
        for(int i=0; i<3; i++)
                std[i].prn();
}
```

실 행 결 과

```
디폴트 생성자 호출
매개변수를 받아온 생성자 호출
소멸자 호출
번호 : 0        이름 : 없어요
번호 : 1        이름 : 김연아
번호 : 1        이름 : 김연아
소멸자 호출
소멸자 호출
소멸자 호출
Press any key to continue
```

9. 사용자의 재정의가 필요한 복사 생성자

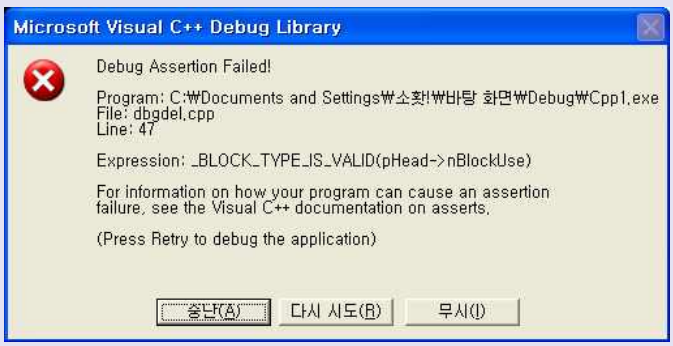
설명 : 복사 생성자는 컴파일러가 자동으로 생성하기 때문에 동적메모리 할당을 하는 객체의 경우 복사 생성자에 의해 객체를 만들게 되면 동적메모리 할당 해제 시 문제가 발생할 수 있다. 이러한 문제가 발생 될 것이 예상된다면 사용자는 복사 생성자를 직접 만들어 사용할 수 있다.

예제 9-1 : 소멸자의 동적 메모리 할당 때문에 오류가 발생하는 예제- 1

```
#include<iostream.h>

class Stud {
private :
        int *ip;
public :
        Stud();
        ~Stud();
        void print();
};

Stud::Stud() {
        ip = new int;
        *ip = 5;
}

Stud::~Stud() {
        delete ip;
}

void Stud::print() {
        cout << "동적할당한 공간에는 " << *ip << "가 들어있습니다." << endl;
}

void main( ) {
        Stud ob;
        ob.print( );
        Stud pb(ob);
        pb.print( );
}
```

실 행 결 과

```
동적할당한 공간에는 5가 들어있습니다.
동적할당한 공간에는 5가 들어있습니다.
```



예제 9-2 : 복사 생성자를 직접 정의하여 예제 9-1의 문제를 해결한 예제

```
#include<iostream.h>

class Stud {
private :
	int *ip;
public :
	Stud();
	Stud(Stud &temp);
	~Stud();
	void print();
};

Stud::Stud() {
	ip = new int;
	*ip = 5;
}

Stud::Stud(Stud &temp) {
	ip = new int;
	*ip = *(temp.ip)
}

Stud::~Stud() {
	delete ip;
}

void Stud::print() {
	cout << "동적할당한 공간에는 " << *ip << "가 들어있습니다." << endl;
}

void main( ) {
	Stud ob;
	ob.print( );
	Stud pb(ob);
	pb.print( );
}
```

실 행 결 과

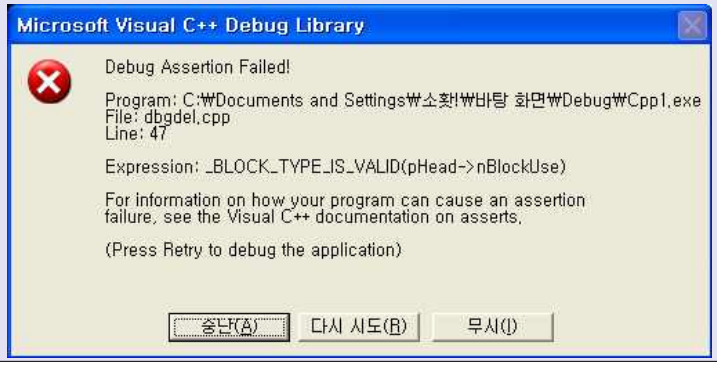
```
동적할당한 공간에는 5가 들어있습니다.
동적할당한 공간에는 5가 들어있습니다.
Press any key to continue
```

예제 9-3 : 소멸자의 동적 메모리 할당 때문에 오류가 발생하는 예제- 2

```
#include <iostream.h>
#include <string.h>

class String {
	char *str;
	int len;
public:
	String(char *);   // 생성자
	~String();        // 소멸자
	void cat(String &);  // 전달인자가 객체인 멤버함수
	void prn();
};

String::String(char *p) {
	len = strlen(p);
	str = new char[len+1];
	strcpy(str, p);
}

String::~String() {
	cout << "Deleting str...Wn";
	delete str;  // 메모리 해제
}

void String::cat(String &s) {
	len = len + s.len;
	char *tmp = new char[len+1];   // 메모리 동적할당
	strcpy(tmp, str);       //문자열 복사
	strcat(tmp, s.str);     // 문자열 연결
	delete str;    // 옛 문자열 해제
	str = tmp;
}

void String::prn() {
	cout << "문자열 : " << str << "(" << len << ")" << endl;
}

int main()
{	String str1("Object-oriented "), str2(str1); // 에러
	str1.prn();
	str2.prn();
	str1.cat(str2);
	cout << "Wn====문자열 연결=========Wn";
	str1.prn();
	return 0;
}
```

실 행 결 과

```
문자열 : Object-oriented (16)
문자열 : Object-oriented (16)

====문자열 연결==========
문자열 : Object-oriented Object-oriented (32)
```



예제 9-4 : 복사 생성자를 직접 정의하여 예제 9-3의 문제를 해결한 예제

```
#include <iostream.h>
#include <string.h>

class String {
        char *str;
        int len;
public:
        String(char *);                     // 생성자
        ~String();                          // 소멸자
        void cat(String &);                 // 전달인자가 객체인 멤버함수
        String(const String &tmp);          // 복사 생성자 재정의
        void prn();
};

String::String(char *p) {
        len = strlen(p);
        str = new char[len+1];
        strcpy(str, p);
}

String::~String() {
        cout << "Deleting str...₩n";
        delete str;  // 메모리 해제
}

void String::cat(String &s) {
        len = len + s.len;
        char *tmp = new char[len+1];        // 메모리 동적할당
        strcpy(tmp, str);                   //문자열 복사
        strcat(tmp, s.str);                 // 문자열 연결
        delete str;    // 옛 문자열 해제
        str = tmp;
}
```

```
// 복사 생성자 재정의
// 복사 생성자는 이미 만들어진 객체를 레퍼런스변수로 받음
String::String(const String &tmp) {
	len = tmp.len;			// 초기값으로 객체의 멤버 변수 복사
	str = new char[len+1];		// 동적 메모리 할당
	strcpy(str, tmp.str);
}

void String::prn() {
	cout << "문자열 : " << str << "(" << len << ")" << endl;
}

int main()
{	String str1("Object-oriented "), str2(str1);
	str1.prn();
	str2.prn();
	str1.cat(str2);
	cout << "Wn====문자열 연결==========Wn";
	str1.prn();
	return 0;
}
```

실 행 결 과

```
문자열 : Object-oriented (16)
문자열 : Object-oriented (16)

====문자열 연결==========
문자열 : Object-oriented Object-oriented (32)
Deleting str...
Deleting str...
Press any key to continue
```