게임 자료구조와 알고리즘

-CHAPTER13-

SOULSEEK

목차

생성자와 소멸자, STATIC

1.생성자와 소멸자, STATIC

생성자

- **Class** 객체가 만들어질 때 자동으로 호출되는 함수

- 특징
  - **Class의 이름과 동일한 이름**으로 함수를 만든다.
  - 함수의 **반환값(return value)**이 없다.
  - 객체의 맴버변수 **초기화**하는 목적으로 사용된다.

```
#include <iostream>
using namespace std;
class Con
{
public:
	Con()
	{
		cout << "생성자 호출" << endl;
	}
};

void main()
{
	Con c;
}
```

1.생성자와 소멸자, STATIC

```
#include <iostream>
 using namespace std;
class Point
{
    private :
      int m_ipx,m_ipy;
    public :
      Point()
      {
            m_ipx = 5;
            m_ipy = 5;
      }
      int getPx() { return m_ipx; }
      int getPy() { return m_ipy; }
};
void main()
{
    Point pt;
    cout << "pt -> x : " << pt.getPx()  << ", y : " << pt.getPy() << endl;
}
```

1.생성자와 소멸자, STATIC

```
#include <iostream>
 using namespace std;
class Point
{
    private :
      int m_ipx,m_ipy;
    public :
      Point(int x,int y)
      {
            m_ipx = x;
            m_ipy = y;
      }
      int getPx() { return m_ipx; }
      int getPy() { return m_ipy; }
};
void main()
{
    Point pt(10,20);
    cout << "pt -> x : " << pt.getPx()  << ", y : " << pt.getPy() << endl;
}
```

1.생성자와 소멸자, STATIC

```
#include <iostream>
 using namespace std;
class Point
{
    private :
      int m_ipx,m_ipy;
    public :
      Point()
      {
            m_ipx = 5;
            m_ipy = 5;
      }
      Point(int x,int y)
      {
            m_ipx = x;
            m_ipy = y;
      }
      int getPx() { return m_ipx; }
      int getPy() { return m_ipy; }
};
void main()
{
    Point pt1,pt2(10,20);
    cout << "pt1 -> x : " << pt1.getPx()  << ", y : " << pt1.getPy() << endl;
    cout << "pt2 -> x : " << pt2.getPx()  << ", y : " << pt2.getPy() << endl;
}
```

1.생성자와 소멸자, STATIC

```
#include <iostream>
using namespace std;
class Point
{
private:
	int m_ipx, m_ipy;
public:
	Point(int x, int y)
	{
		m_ipx = x;
		m_ipy = y;
	}
	int getPx() { return m_ipx; }
	int getPy() { return m_ipy; }
};

void main()
{
	Point pt[3] = { Point(3,5),Point(20,40),Point(50,80) };
	for (int i = 0; i < 3; i++)
		cout << "pt[" << i << "]->x : " << pt[i].getPx() << ", y : " << pt[i].getPy() << endl;
}
```

1.생성자와 소멸자, STATIC

복사 생성자

- 동일한 **class** 형의 다른 객체의 정보를 복사하여 새로운 객체를 만든다.
- 일반 변수를 초기화 할 때 다른 변수의 값을 이용하는 경우와 같다.

```
void main()
{
	int a = 10;
	int b = 10;
}
```

```
void main()
{
	int a = 10;
	int b = a;
}
```

1.생성자와 소멸자, STATIC

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

class person
{
private:
	int m_iAge;
	string m_strName;
public:
	person(int _age, string tmp)
	{
		m_iAge = _age;
		m_strName = tmp;
	}
	person(person* tmp)
	{
		m_iAge = tmp->m_iAge;
		m_strName = tmp->m_strName;
	}
	void print()
	{
		cout << "나이 : " << m_iAge << endl;
		cout << "이름 : " << m_strName << endl;
	}
};
void main()
{
	person p1(99, “SoulSeek");
	person p2(&p1);
	p1.print();
	p2.print();
}
```

1.생성자와 소멸자, STATIC

콜론 초기화

- 해당 생성자 함수 영역 생성보다 먼저 매개변수를 받아 초기화 하는 방법.
- **class** 맴버 변수 중에서 변수 선언과 동시에 초기화 해야 하는 변수에 주로 사용

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

class A
{
private:
	const int m_iNum1;
	int m_iNum2;
public:
	A(int Num1,int Num2)
	{
		m_iNum1 = Num1;
		m_iNum2 = Num2;
	}
	void print()
	{
		cout << "const int m_iNum1 = " << m_iNum1 << endl;
		cout << "int m_iNum2 = " << m_iNum2 << endl;
	}
};

void  main()
{
	A a(10, 20);
	a.print();
}
```

1.생성자와 소멸자, STATIC

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

class A
{
private:
	const int m_iNum1;
	int m_iNum2;
public:
	A(int Num1, int Num2) : m_iNum1(Num1)
	{
		m_iNum2 = Num2;
	}
	void print()
	{
		cout << "const int m_iNum1 = " << m_iNum1 << endl;
		cout << "int m_iNum2 = " << m_iNum2 << endl;
	}
};

void  main()
{
	A a(10, 20);
	a.print();
}
```

1.생성자와 소멸자, STATIC

소멸자

- **Class** 객체가 소멸될 때 자동으로 호출되는 함수

- 특징
  -객체가 소멸되는 시점에 자동으로 호출된다.
  -함수의 **반환값(return value)**이 없다.
  -함수의 매개변수가 없다.
  -소멸자는 사용자가 직접 호출할 수 있다.
  -객체의 동적 할당된 맴버 변수를 할당해제 하는 목적으로 사용된다.

1.생성자와 소멸자, STATIC

```
#include <iostream>
using namespace std;
class Con
{
public:
	Con()
	{
		cout << "생성자 호출" << endl;
	}
	~Con()
	{
		cout << "소멸자 호출" << endl;
	}
};
void main()
{
	Con c;
	cout << "Hello" << endl;
}
```

1.생성자와 소멸자, STATIC

```
#include <iostream>
#include<string>
using namespace std;
class str
{
private:
	char* name;
public:
	str()
	{
		string str;
		cout << "이름 입력 : ";
		cin >> str;
		name = new char[str.length()+1];
		strcpy(name, str.c_str());
	}
	~str()
	{
		cout << name << " 의 동적 할당을 해제합니다.\n";
		delete[] name;
	}
	void Disp()
	{
		cout << "name = " << name << endl;
	}

};
```

```
void main()
{
	str st1, st2;
	st1.Disp();
	st2.Disp();
}
```

1.생성자와 소멸자, STATIC

	생성자	소멸자
사용용도	맴버변수 초기값 부여	맴버변수의 할당된 공간 해제
형식	**Class**이름(매개변수)	**~Class**이름()
호출시점	객체 생성시	객체 소멸시
매개변수	있음/없음	없음
오버로딩	가능	불가능

1.생성자와 소멸자, STATIC

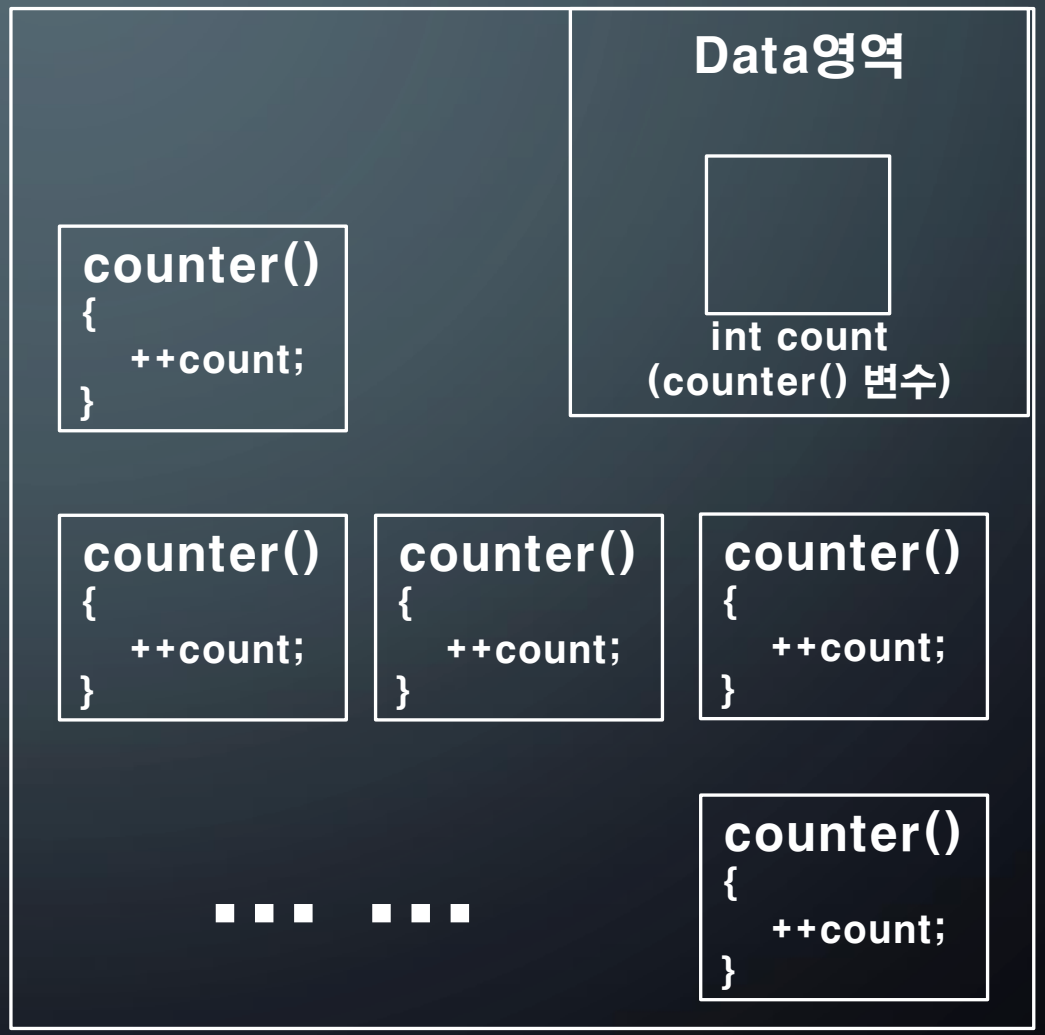
Static

- 해당 변수를 **특정 영역에 제한**된 전역변수로 변환한다.
- **Static**선언 위치에 따라 사용법이 조금 달라진다.

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

void counter()
{
    static int count = 0;
    cout << ++count << endl;
}

void main()
{
    for(int i = 0 ; i < 10; i++)
    counter();
    //count = 20;
}
```

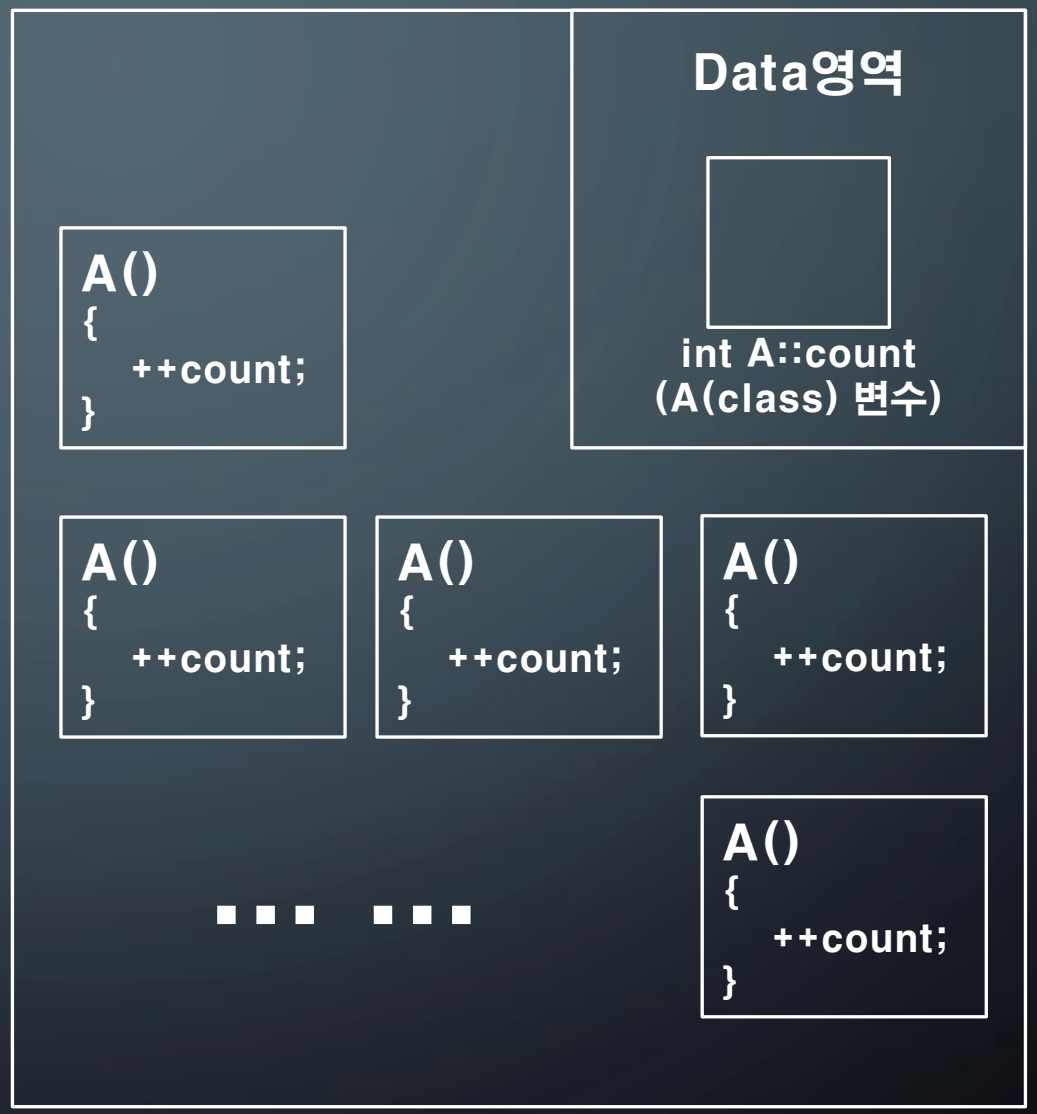


1.생성자와 소멸자, STATIC

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

class A
{
    public :
    static int count;
    A()
    {
        cout << ++count << endl;
    }
};

int A::count = 0;
void main()
{
    A a1;
    A a2;
    A a3;
    A::count = 10;
    A a4;
    A a5;
}
```



1.생성자와 소멸자, STATIC

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

class A
{
	private:
		int Num;
	public:
		static int count;
		A()
		{
			Num = 10;
			cout << ++count << endl;
		}
		static void Print()
		{
			cout << "count : " << count << endl;
			//cout << "Num : " << Num << endl;
		}
};
int A::count = 0;
void main()
{
	A a1;
	A a2;
	A::count = 10;
	A a3;
	A a4;
	A a5;
	a5.Print();
}
```

THIS포인터

2.THIS포인터

- This포인터는 **자신을 가리키는 포인터변수**다.
- 보통 맴버함수에서 맴버변수에 접근하기 위해 사용한다.
- 객체의 0번째 맴버변수 라고도 한다.

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

class calc
{
  private :
    int num1;
    int num2;
  public :
    void Setdata(int num1,int num2)
    {
       num1 = num1;
       num2 = num2;
    }
    int GetSum()
    {  return num1 + num2; }
};

 void main()
{
    int su1,su2;
    calc ca;
    cin >> su1 >> su2;
    ca.Setdata(su1,su2);
    cout << su1 <<" + "<< su2 << "= “ << ca.GetSum();
}
```

2.THIS포인터

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

class calc
{
   private :
     int num1;
     int num2;
   public :
     void Setdata(int num1,int num2)
     {
         this->num1 = num1;
         this->num2 = num2;
     }
     int GetSum()
     {  return num1 + num2; }
};

 void main()
{
     int su1,su2;
     calc ca;
     cin >> su1 >> su2;
     ca.Setdata(su1,su2);
     cout << su1 <<" + "<< su2 << "=“ << ca.GetSum();
}
```

NAMESPACE

2.NAMESPACE

- 특정 코드영역에 이름을 부여하여 구분하는 작업
- 지역을 구분하여 가독성을 올리고 중복성을 제어한다.

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

namespace A
{
	void printf(string a)
	{
		cout << "저는 A팀의 printf 입니다.";
		cout << endl << a;
	}
}
namespace B
{
	void printf(string a)
	{
		cout << "저는 B팀의 printf 입니다.";
		cout << endl << a;
	}
}
void main()
{
	A::printf("Hello");
	cout << endl << endl;
	B::printf(“Hello”);
}
```

2.NAMESPACE

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

namespace A
{
	int score = 100;
	string name = "A팀";
}
namespace B
{
	int score = 200;
	string name = "B팀";
}

void main()
{
	cout << A::name << "Score = " << A::score << endl;
	cout << B::name << "Score = " << B::score << endl;
}
```

2.NAMESPACE

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

namespace A
{
	int score = 100;
	string name = "A팀";
}
namespace B
{
	int score = 200;
	string name = "B팀";
}
using namespace A;


void main()
{
	cout << name << "Score = " << score << endl;
	cout << B::name << "Score = " << B::score << endl;
}
```

2.NAMESPACE

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

string str = "Global Variable\n";

namespace Nsp
{
	string str = "Namespace Variable\n";
}

void main()
{
	string str = "Local Variable\n";
	cout << str;
	cout << Nsp::str;
	cout << ::str;
}
```

학습과제

- 생성자 인자값 없이 자동으로 1~10까지의 총합을 구하는 Class를 만들어 보자
- 생성자의 인자값을 하나만 받고 1~ 받은 인자값 만큼의 총합을 구하는 Class를 만들어 보자.
- **Computer Class 를 만들 것**
  (main에서는 하나의 맴버함수만 사용할 것)
  (컴퓨터 정보는 생성자에서 초기화)
  계산기 호출 명령어 = system("calc");
  메모장 호출 명령어 = system("notepad");
  그림판 호출 명령어 = system("mspaint");