5. 상속 (inheritance)

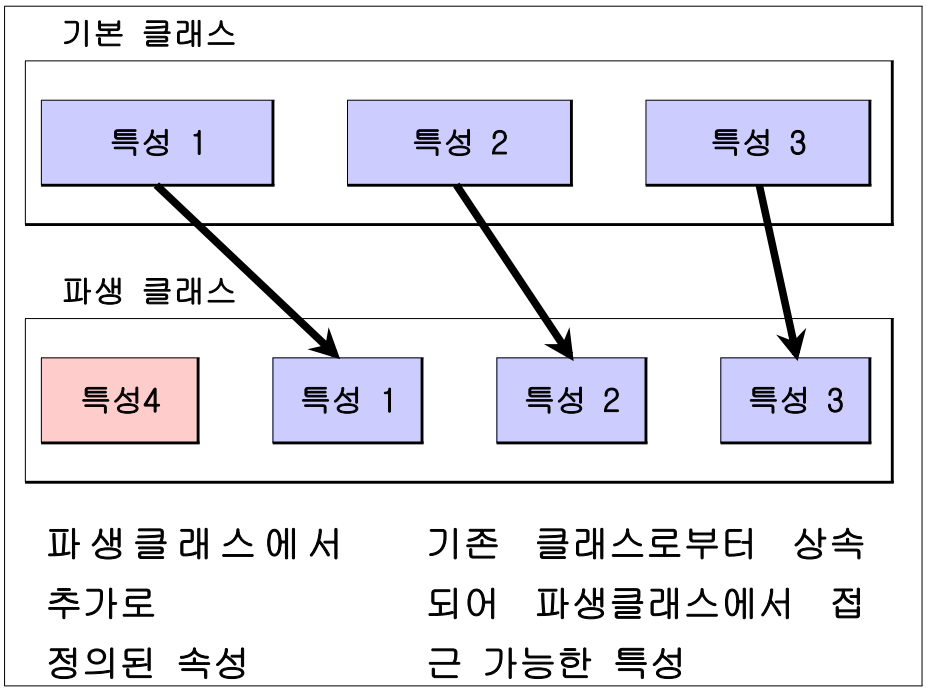
♲ 상속이란?

한 클래스가 다른 클래스에서 정의된 속성 (데이터, 함수)을 그대로 물려받을 수 있는 객체지향의 프로그래밍의 중요한 개념이다. 즉, 하나의 객체에 다른 객체의 성질들을 이어 받을 수 있도록 하는 것이다. 상속은 클래스화의 개념을 제공하여 대부분의 지식을 계층적인 클래스로 관리할 수 있게 한다. ex) 감귤이라는 품종은 귤이라는 클래스의 일부이며 귤은 과일이라는 클래스에 속한다.

상속을 사용하는 시기

부모클래스와 비슷한 일을 하지만 추가적으로 더 해주고 싶은 일이 있을때 자식 클래스를 생성하여 사용

클래스간의 상속 관련성



기본클래스의 멤버들에 대한 접근

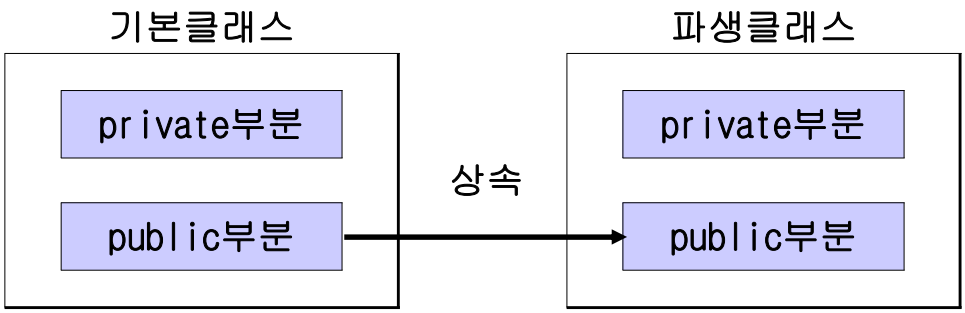
파생클래스의 기본클래스 멤버들에 대한 접근 여부는 엑세스 지정자의 의해서 결정된다. 만약 명확한 엑세스 지정자가 없는 경우에 파생 클래스의 엑세스 지정자는 디폴트 값으로 private이 지정된다. 그리고 파생클래스가 구조체(struct)일 때 명확한 엑세스 지정자가 없다면 디폴트로 public이 지정된다.

기본 엑세스 지정자

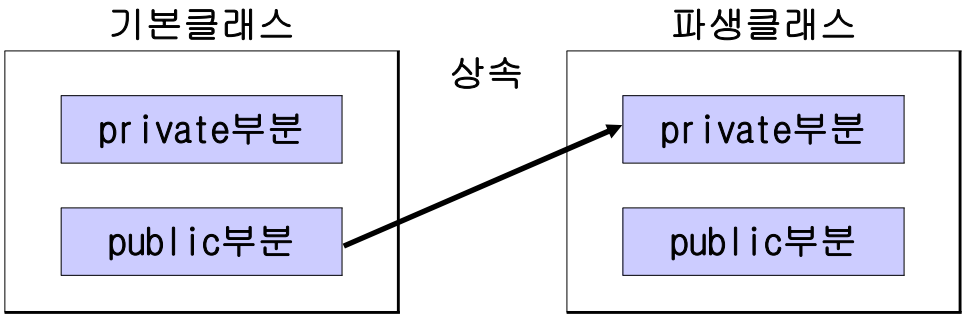
public	기본클래스의 공용멤버(public)가 파생클래스의 공용멤버(public)처럼 사용되며 기본클래스의 모든 보호 멤버들은 파생클래스에서도 보호 멤버가 된다.	※ 두 경우 모두 기본클래스의 전용멤버(private)들은 파생 클래스에서 상속받지 못한다.
private	기본클래스의 공용멤버(public)가 파생클래스의 전용멤버(private)처럼 사용된다	

엑세스 지정자에 따른 상속 관계

Public



Private



상속의 특징

① 특정한 클래스를 바탕으로 이 클래스에 또 다른 기능을 넣어서 새로운 클래스를 만들 때 사용
② 멤버들을 넘겨 주는 클래스는 부모 클래스가 됨
③ 멤버들을 물려받는 클래스는 자식 클래스가 됨
④ 자식 클래스는 부모 클래스의 멤버변수와 멤버 함수를 사용할 수 있게됨
⑤ 코드의 재활용성을 지원하므로 중요한 개념
⑥ 시간과 비용을 절약하며 코드의 신뢰성을 높인다. 또한 클래스 라이브러리들을 쉽게 분산할 수 있게 한다.
⑦ 기존의 클래스는 부모 클래스, 기반 클래스, 기본 클래스 등으로 불리우며 새롭게 파생된 클래스는 파생 클래스, 유도된 클래스, 자식 클래스 등으로 불린다.

파생클래스의 선언 형식

```
class 파생클래스 이름 : 액세스 지정자 (private, public) 기본클래스 이름 {
      // 클래스 본체
};
```

※ 파생클래스 뒤에 : 을 명시한 후 엑세스 지장자와 기본클래스의 이름을 기술

♲ 상속의 사용

Ex1) 간단한 상속 프로그램

```
#include <iostream.h>

class base {
  int i, j;
public:
  void set(int a, int b) { i = a; j = b;}
  void show_ij() {
              cout << "₩n i = " << i << ",  j = " << j << endl;
       }
};

class derived : public base {
  int k;
public:
  derived(int x) { k = x; }
  void show_k() { cout << " k = " << k << endl;}
};

int main()
{
  derived ob(3);

  ob.set(1, 2);  // 기본 클래스 멤버 접근
  ob.show_ij();  // 기본 클래스 멤버 접근
  ob.show_k();  // 파생 클래스 멤버 접근

  return 0;
}
```

실 행 결 과	결 과 분 석
i = 1, j = 2 k = 3 Perss any key to continue...	기본클래스인 base 클래스를 public으로 상속받은 derived 클래스는 기본클래스의 set()함수를 상속받았음으로 set()함수에 접근 할 수 있다.

Ex2) 에러가 발생하는 상속 프로그램

```
#include <iostream.h>

class base {
  int i, j;
public:
  void set(int a, int b) { i = a; j = b; }
  void show_ij() {
              cout << "₩n i = " << i << ",  j = " << j << endl;
       }
};

class derived : private base {
  int k;
public:
  derived(int x) { k = x;}
  void show_k() { cout << " k = " << k << endl; }
};

int main()
{
  derived ob(3);

  ob.set(1, 2);  // 오류, set()에 접근 불가
  ob.show_ij();  // 오류, show_ij()에 접근 불가

  return 0;
}
```

실 행 결 과	결 과 분 석
Error	기본 클래스에 대한 접근 지시자가 private일 때 기본클래스의 public과 protected 멤버들은 파생 클래스 멤버에 대해서도 비공개 멤버들이 되기 때문에 외부에서의 접근이 불가능하다

Ex3) 원의 중심점을 상속받아 원의 면적을 구하는 프로그램 #1

```
#include <iostream.h>

class Point {
	int xp, yp;
public :
	void set_xy(int x, int y) { xp = x, yp = y; }
	int get_x()  { return xp; }
	int get_y()  { return yp; }
};

class Circle : public Point {
	float r;
public:
	void set_rd(float d) { r = d; }
	float area() { return 3.14*r*r; }
};

int main() {
	Circle c;
	c.set_xy(10, 20);  // 기본 클래스의 멤버 호출
	c.set_rd(15.2);

	cout << "Wn원의 중심점 : x = " << c.get_x();
	cout << ", y = " << c.get_y() << endl;
	cout << "원의 면적 : " << c.area() << endl;
	return 0;
}
```

실 행 결 과	결 과 분 석
원의 중심점 : x=10, y=20 원의면적 : 725.466 Perss any key to continue...	기본클래스 Point의 public멤버 set_xy(), get_x(), get_y() 함수를 Point의 파생 클래스인 Circle이 public멤버로 상속 받아서 객체 c로 호출하였다.

Ex4) 원의 중심점을 상속 받아 원의 면적을 구하는 프로그램 #2

```
#include <iostream.h>

class Point {
	int xp, yp;
public :
	void set_xy(int x, int y) { xp = x, yp = y; }
	int get_x()  { return xp; }
	int get_y()  { return yp; }
};

class Circle : private Point {
	float r;
public:
	void set_rd(float d) { r = d; }
	float area() { return 3.14*r*r; }
	void center(int x, int y) { set_xy(x, y); } // 기본클래스의 멤버 set_xy()호출
	void prn_xy() {  // 기본클래스의 멤버 get_x(), get_y()호출
		cout << "Wn원의 중심점 : x = " << get_x();
		cout << ", y = " << get_y() << endl;
	}
};

int main() {
	Circle c;
	c.center(10, 20);  // 파생 클래스의 멤버 호출
	c.set_rd(15.2);
	c.prn_xy();

	cout << "원의 면적 : " << c.area() << endl;
	return 0;
}
```

실 행 결 과	결 과 분 석
원의 중심점 : x=10, y=20 원의면적 : 725.466 Perss any key to continue...	기본클래스 Point의 public멤버 set_xy(), get_x(), get_y() 함수를 Point의 파생 클래스인 Circle이 전용멤버로 상속 받았지 때문에 클래스 멤버함수인 center(), pm_xy()함수를 통하여 호출하였다.

protected의 상속

protected 는 상속성 체계에 좀더 많은 융통성을 부여하기 위해 만들어진 연산자이다. 파생클래스의 멤버 함수는 기본 클래스의 private 멤버에 대해서는 접근이 불가능하나 protected를 사용하여 정의된 기본 클래스의 멤버들은 파생클래스의 멤버 함수에서 직접 접근이 가능해진다.

키워드에 따른 멤버변수의 엑세스 가능성

키 워 드	자신 클래스에서의 엑세스 가능성	파생된 클래스 에서의 엑세스 가능성	클래스 외부 객체 에서의 엑세스 가능성
private	○	×	×
protected	○	○	×
public	○	○	○

Ex5) 기본클래스의 멤버 변수가 protected로 지정되어 상속된 프로그램-1

```
#include <iostream.h>
class Counter {
      protected:
            unsigned int count;
      public :
            Counter(  ) { count = 0; }
            Counter(int c) { count = c; }
            int get(  ) { return count; }
            Counter operator++(  ) {
                  count++;
                  return Counter(count);
            }
};
class CountDn : public Counter {
      public :
            Counter operator--(  ) {
                  count--;
                  return Counter(count);
            }
};
void main(  ) {
      CountDn cnt;
      cout << "Wn cnt = " << cnt.get(  );
      ++cnt; ++cnt; ++cnt;
      cout << "Wn cnt = " << cnt.get(  );
      --cnt; --cnt;
      cout << "Wn cnt = " << cnt.get(  );
}
```

실 행 결 과

```
cnt = 0
cnt = 3
cnt = 1

Perss any key to continue...
```

Ex6) 기본클래스의 멤버 변수가 protected로 지정되어 상속된 프로그램-2

```
#include <iostream.h>

class Point {  // 기본 클래스
protected :
	int xp, yp;
public :
	void set_xy(int x, int y) { xp = x, yp = y; }
	int get_x()  { return xp; }
	int get_y()  { return yp; }
};

class Circle : public Point {
	float r;
public:
	void set_rd(float d) { r = d; }
	float area() { return 3.14*r*r; }
};

int main() {
	Circle c;
	c.set_xy(10, 20);  // 기본 클래스의 멤버 호출
	c.set_rd(15.2);

	cout << "₩n원의 중심점 : x = " << c.get_x();
	cout << ", y = " << c.get_y() << endl;
	cout << "원의 면적 : " << c.area() << endl;
	return 0;
}
```

실 행 결 과

```
원의 중심점 : x = 10, y = 20
원의 면적 : 725.466

Press any key to continue...
```

Ex7) base기본 클래스로부터 두 개의 파생 클래스가 각각 덧셈과 곱셈을 수행하는 프로그램

```
#include<iostream.h>
class base{
protected:
	int a, b, c;
public:
	void initial(int, int);   // 초기값 지정 함수
	void prn();                    // 출력 함수
 };
 void base::initial(int x, int y)  {
	a = x;    b = y;
 }
 void base::prn() {
	cout << "a = " << a << "  b = " << b << "  c = " << c << endl;
 }
 class add:public base{
 public :
	void sum();
 };
 void add::sum() {
	c = a + b;
 }
 class mul:public base{
 public:
	void gob();
 };
 void mul::gob() {
	c = a * b;
 }

 void main() {
	add x;
	x.initial(3, 5);
	mul y;
	y.initial(2, 7);
	x.sum();                x.prn();
	y.gob();                y.prn();
 }
```

실 행 결 과

```
a = 3  b = 5   c = 8
a = 2  b = 7   c = 14

Press any key to continue...
```

Ex8) 기본클래스로부터 책의 정보를 상속받아 이에 관한 정보를 출력하는 프로그램

```
#include <iostream.h>
#include <string.h>
class Base_class {
protected:
    char *book;
    int year;
public:
     void set()  {
         book = new char[35];
         strcpy (book, "C++ 프로그래밍");
         year = 2005;
     }
 };

 class Der_class : public Base_class {
     char author[15];
 public:
     Der_class()   {
         strcpy (author, "Kim Lee Park" );
     }
     void print()   {
         cout << "Book Title : " << book << endl;        // 상속된 멤버변수
         cout << "Year : " << year << endl;          // 상속된 멤버변수
         cout << "Author : " << author << endl;    // 파생클래스에서 추가된 멤버변수
     }
 };

 int main()  {
     Der_class ob;     // 파생 클래스형의 객체
     ob.set();     // 기본 클래스로부터 상속받은 멤버함수 호출
     ob.print();
     return 0;
 }
```

실 행 결 과

```
Book Title : C++ 프로그래밍
Year : 2005
Author : Kim Lee Park

Press any key to continue...
```

Ex9) 상속관계에서 생성자와 소멸자에 관한 프로그램-1

```
#include <iostream.h>
class Point {
	int xp, yp;
public :
	Point() {
		cout << "기본 클래스의 생성자₩n";
		xp = yp = 0;
	}
	~Point(){
		cout << "기본 클래스의 소멸자₩n";
	}
	void set_xy(int x, int y) { xp = x, yp = y; }
	int get_x()  { return xp; }
	int get_y()  { return yp; }
};

class Circle : public Point {
	float r;
public:
	Circle(float d) {
		cout << "파생 클래스의 생성자₩n";
		r = d;
  }
	 ~Circle() {
		cout << "파생 클래스의 소멸자₩n";
	}
	float area() { return 3.14*r*r; }
};

void main() {
	Circle c(7.6);
	c.set_xy(10, 20);  // 기본 클래스의 멤버 호출

	cout << "₩n원의 중심점 : x = " << c.get_x();
	cout << ", y = " << c.get_y() << endl;
	cout << "원의 면적 : " << c.area() << endl;
}
```

실 행 결 과

```
기본 클래스의 생성자
파생 클래스의 생성자
원의 중심점 : x = 10, y = 20
원의 면적 : 181.366
파생 클래스의 소멸자
기본 클래스의 소멸자
Press any key to continue...
```

Ex10) 상속관계에서 생성자와 소멸자에 관한 프로그램-2

```
#include<iostream.h>

class base{       // 기반 클래스
public:
       base();
       ~base();
};
base::base() {
       cout << "₩n Base Constructor ₩n";
}
base::~base() {
       cout << "₩n Base Destructor ₩n";
}

class derived:public base{  // 파생 클래스
public :
       derived();
       ~derived();
};
derived::derived() {
       cout << "₩n Derived Constructor ₩n";
}
derived::~derived() {
       cout << "₩n Derived Destructor ₩n";
}
int main() {
       derived ob;
       return 0;
}
```

실 행 결 과

```
 Base Conestructor

 Derived Constructor

 Derived Destructor

 Base Destructor

Perss any key to continue...
```

Ex11) 파생클래스의 생성자 함수에서 기본클래스의 생성자 함수 호출-1

```
#include<iostream.h>
class base{                          // 기본 클래스
protected:                                  // 상속을 위한 액세스 지정자
       int a, b, c;
public:
       base();
       base(int, int);                      // 생성자 함수 선언
       void print();
};
base::base( ) {
       a = b = c = 0;
}
base::base(int x, int y) {                  // 생성자 함수 정의
       a = x;
       b = y;
       c = 0;
}
void base::print() {
  cout << "Wn base::print() 함수안에서...Wn";
       cout << "a = " << a << "  b = " << b << "  c = " << c << endl;
}

class add:public base{
public :
       void sum();
       add(int x, int y);
};
add::add(int x, int y) : base() {  // 기본 클래스의 생성자 base() 호출
  cout << "Wn add(int, int) 생성자안에서...Wn";
       cout << "a=" << a <<  "  b = " << b << "  c = " << c << endl;
       a = x;   b = y;
}
void add::sum() {
       c = a + b;
}

class mul:public base{
public:
       void gob();
       mul(int x, int y);
};
mul::mul(int x, int y) : base(x, y) { // 기본 클래스의 생성자 base(int, int) 호출
  cout << "Wn mul(int, int) 생성자안에서...Wn";
       cout << "a=" << a <<  "  b = " << b << "  c = " << c << endl;
       a = x;          b = y;
}
void mul::gob() {
       c = a * b;
}

int main() {
       add x(3, 5);
       mul y(2, 7);
       x.sum();        x.print();
       y.gob();        y.print();
       return 0;
}
```

실 행 결 과

```
add (int, int) 생성자 안에서...
a = 0 b = 0 c = 0
mul (int, int) 생성자 안에서...
a = 2 b = 7 c = 0
base::print() 함수 안에서...
a = 3 b = 5 c = 8
base::print() 함수 안에서...
a = 2 b = 7 c = 14
Press any key to continue...
```

Ex12) 파생클래스의 생성자 함수에서 기본클래스의 생성자 함수 호출-2

```
#include <iostream.h>
class Date {
	protected:
		int year, month, day;
public:
	Date(int yy, int mm, int dd) {
		year = yy;	month = mm;		day = dd;
  }
	void show() {
		cout << year << "년 " << month << "월 " << day << "일" << endl;
	}
};

class Derv_Date : public Date {
	public:
		Derv_Date(int yy, int mm, int dd) : Date(yy, mm, dd)    {
		}  // yy, mm, dd는 인수 전달만을 해줌
		void show();
};
void Derv_Date::show() {
	static char *mon[] = {
		"Jan.", "Feb.", "Mar.", "Apr.", "May.", "June.",
		"July.", "Aug.", "Sep.", "Oct.", "Nov.", "Dec."
	};
	cout << mon[month - 1] << ' ' << day << ' ' << year << endl;
}
void main() {
	Date date1(1995, 5, 5);
	Derv_Date date2(2005, 5, 5);
	date1.show();
	date2.show();
}
```

실 행 결 과

```
1995년 5월 5일
May. 5 2005

Press any key to continue...
```

Ex13) 파생클래스의 생성자 함수에서 기본클래스의 생성자 함수 호출-3

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

class Base_class {
    char book[35];
    int year;
public:
    Base_class(char *title, int p_year)   {
        strcpy(book, title);
        year = p_year;
    }
    void print()   {
        cout << "Title : " << book << endl;
        cout << "Year : " << year << endl;
    }
};

class Der_class : public Base_class {
    char Author[15];
public:
    Der_class(char *title, int p_year, char *author):Base_class(title, p_year) {
        strcpy(Author, author);
    }
    void print();
};
void Der_class::print()  {
    Base_class::print();            // 기본 클래스의 멤버함수 호출
    cout << "Author : " << Author << endl;
}

void main() {
    Der_class book("C++ 프로그래밍", 2005, "Kim Lee Park");
    book.print();
}
```

실 행 결 과

```
Title : C++ 프로그래밍
Year : 2005
Author : Kim Lee Park

Press any key to continue...
```

Ex14) 콜론 초기화 방법을 이용하여 생성자를 초기화 하는 프로그램

```
#include<iostream.h>
#include<iomanip.h>
#include<string.h>
class Goods {                // 기본 클래스
protected:                          // 상속을 위한 지정자
   char that[10];                 // 품명
   int qty;                     // 수량
   int cost;                  // 단가
public:
   Goods(char THAT[], int QTY, int COST) : qty(QTY), cost(COST)  {
      strcpy(that, THAT);
   }
};

class D_Goods : Goods {         // 파생 클래스
    char client[30];     // 거래처
    int year;                   // 판매 일자
    int month;
    int day;
public:
    D_Goods(char THAT[], int Q, int C, char cl[], int Y, int M, int D) :
      Goods(THAT, Q, C), year(Y), month(M), day(D)  {
        strcpy(client, cl);
    }
    void print();
};

void line_prn() {  // 일반 함수
       for(int i=0; i<3; i++)
                cout << "--------------------";
       cout << endl;
}

void D_Goods::print()  {
    cout << setiosflags(ios::left);  // 왼쪽정렬
    cout << setw(10) << "Wn품목" << setw(6) << "수량" << setw(6) << "단가"  << setw(7) << "금액"
         << setw(15) << "거래처" << "판매일자" << endl;
    line_prn();
    cout << setiosflags(ios::left);
    cout << setw(10) << that ;
    cout << setiosflags(ios::right);   // 오른쪽정렬
    cout << setw(6) << qty << setw(6) << cost  << setw(7) << (qty * cost);
    cout << setiosflags(ios::left) << setw(15) << client;
    cout << setiosflags(ios::right) << setw(4) << year << "." << month << "." << day << endl;
}

void main()  {
    D_Goods Item("손목시계", 50, 1000, "(주)청설", 2005, 12, 25);
    Item.print();
}
```

실 행 결 과

```
품목            수량     단가     금액     거래처          판매일자
-------------------------------------------------------
손목시계        50       1000     50000    (주)청설              2005.12.25

Press any key to continue...
```

Ex15) 상속 관계에서 포인터 객체를 이용한 프로그램

```
#include <iostream.h>

class Parents{
public:
   void print()   {
          cout << "₩n 부모 클래스 ₩n";
   }
};

class Child : public Parents  {
public:
   void print()   {
          cout << "₩n 자식 클래스 ₩n";
   }
};

int main()  {
   Parents *mom;    // 기본 클래스형의 포인터 객체
   Child son;

   mom = &son;
   mom -> print( );  // 기본 클래스의 멤버함수 호출
        return 0;
}
```

실 행 결 과

```
 부모 클래스

Press any key to continue...
```

Ex16) 스택 프로그램을 상속을 이용하여 구현한 예제

```
#include <iostream.h>
#include <process.h>
const int MAX = 5;
class STACK {
	protected:
		int st[MAX], top;
	public:
		STACK() { top = 0; }
		void push(int data) { st[++top] = data; }
		int pop() { return st[top--]; }
};

class STACK2:public STACK {
	public:
		void push(int data) {
		    if(top < MAX)
			STACK::push(data);
		    else {
			cout << "₩n Error : Stack is full!₩n";
			exit(1);
			}
		}
		int pop() {
			if(top > 0)
				return STACK::pop();
			else {
			cout << "₩nError : Stack is empty!₩n";
			exit(1);
			}
		}
};
void main()  {
	STACK2 s1;
	s1.push(100);
	s1.push(200);
	s1.push(300);
	cout << endl << s1.pop();
	cout << endl << s1.pop();
	cout << endl << s1.pop();
	cout << endl << s1.pop();
}
```

실 행 결 과

```
300
200
100
Error : stack is empty!
Press any key to continue...
```