

C++ 프로그래밍

□ C++ 기초

○ 프로그래밍 언어

- * 기계어 : 0, 1 로 이루어진 언어
- * 어셈블리어 : 기계어의 명령을 ADD, SUB, MOVE 등과 같이 상징적인 니모닉 기호로 일대일 대응시킨 언어
- * 고급언어 : 사람이 이해하기 쉬운 언어

○ C++에 추가된 기능

- * 함수 중복 (function overloading)
 - 매개 변수의 개수나 타입이 다른 동일한 이름의 함수들 선언
- * 디폴트 매개 변수 (default parameter)
 - 매개 변수에 디폴트 값이 전달되도록 함수 선언
- * 참조와 참조 변수 (reference)
 - 하나의 변수에 별명을 사용하는 참조 변수 도입
- * 참조에 의한 호출 (call-by-reference)
 - 함수 호출 시 참조 전달
- * new / delete 연산자
 - 동적 메모리 할당/해제를 위해 new와 delete 연산자 도입
- * 인라인 함수
 - 함수 호출 대신 함수 코드의 확장 삽입
 - C언어가 자바보다 빠르다.
- * 연산자 재정의 (overriding)
 - 기존 C++ 연산자에 새로운 연산 정의
- * 제너릭 함수와 클래스 (일반화 프로그래밍)
 - 데이터 타입에 의존하지 않고 일반화시킨 함수나 클래스 작성 가능
 - 어떤 데이터 타입이든지 적용하면 사용할 수 있음

○ C++ 장단점

- * 장점 : 기존에 개발된 C프로그램 코드 활용
- * 단점 : 캡슐화의 원칙이 무너지짐
 - C++에서 전역 변수와 전역 함수를 사용할 수밖에 없음

○ 기본 구조

소스 코드

```
#include<iostream> // .h가 없음

using namespace std; // 작성 안할 경우 std::cout 식으로 일일이 붙여줘야함

int main()
{
    cout << "야 시인난다 ~~\n";
    return 0;
}
```

- #include<iostream> // .h가 없음
- #include<string> // string 타입을 사용하기 위함
- #include<iomanip> // 입출력 조작자를 사용하기 위함

○ 입출력

- * `cout << "글자" << "글자" << 5.5 << c << true << endl;` // 인자타입이 자동으로 설정된다. `endl`은 줄바꿈
- * `cin >> 변수1 >> 변수2;` // 변수 2개를 입력받음
- * `>>` 은 입력받은 값을 스트림(임시 저장)에 넘겨주는 것

소스 코드

```
#include <iostream>
using namespace std;
void main(){
    int width;
    int height;

    cout << "너비 입력 : ";
    cin >> width;
    cout << "높이 입력 : ";
    cin >> height;

    cout << "너비 : " << width << "높이 : " << height << "넓이 : " << width*height << endl;
}
```

* 입출력 조작자

- * `cout.int width(int i);` : 최소 필드 너비를 조정 // 한 번 사용하면 사라진다, 마지막 숫자는 반올림된다.
- 디폴트 값 : 6
- * `cout.char fill(char c);` : 필드 내의 공백 자리에 채워질 문자 설정 // 한 번 설정하면 계속 남아있다.
- * `cout.setf(ios::left);` : 어느 방향으로 정렬할 것인지 설정 // 한 번 설정하면 계속 남아있다.
- * `cout.int precision(int p);` : 실수 출력 시 출력되는 총 자릿수, 출력 형식이 fixed 또는 scientific이라면 소수점 이하 자릿수 // 한 번 설정하면 계속 남아있다.

소스 코드

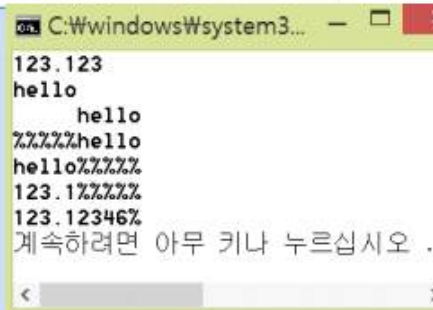
```
#include <iostream>
using namespace std;
int main(void)
{
    cout << 123.1234567 << endl;
    cout << "hello" << endl ;
    cout.width(10);cout << "hello" << endl;
    cout.fill('%');cout.width(10); cout << "hello" << endl ;
    cout.setf(ios::left); cout.width(10);cout << "hello" << endl ;
    cout.width(10);cout.precision(4);cout << 123.1234567 << endl;
    cout.width(10);cout.precision(8);cout << 123.1234567 << endl;
    return 0;
}
```

* 입출력 조작자 (전역 함수) // #include<iomanip>를 포함하고 cout 내부에 작성해야함

- * `setw(int)` : 필드 너비 조정, 이후 한 번의 출력 후 디폴트로 환원됨
- * `setfill(char)` : 공백 자리 채움 문자 지정
- * `setprecision(int)` : 실수 출력 자릿수 설정
- * `flush` : 스트림을 비움

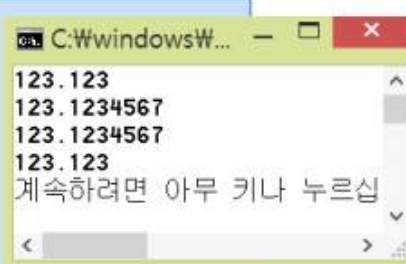
소스 코드

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
    cout << 123.1234567 << endl;
    cout << "hello" << endl;
    cout << setw(10) << "hello" << endl ;
    cout << setfill('%') << setw(10) << "hello" << endl ;
    cout << setw(10) << left << "hello" << endl ;
    cout << setw(10) << setprecision(4) << 123.1234567 << endl;
    cout << setw(10) << setprecision(8) << 123.1234567 << endl;
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
123.123
hello
hello
%%%hello
hello%%%
123.1%%
123.12346%
계속하려면 아무 키나 누르십시오 . . .
```

```
int main(void) {
    double dvalue = 123.1234567;
    int oldpre = cout.precision();
    cout << dvalue << endl;
    cout << setprecision( 10 ) << dvalue << endl ;
    cout << dvalue << endl;
    cout << setprecision(oldpre) << dvalue << endl;
    return 0;
}
```

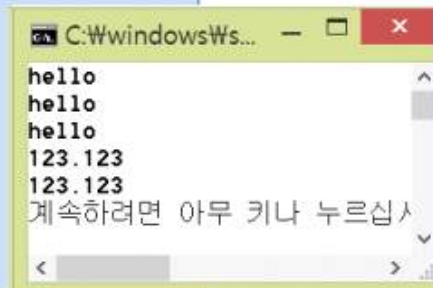


```
C:\Windows\system32\cmd.exe
123.123
123.1234567
123.1234567
123.123
계속하려면 아무 키나 누르십시오 . . .
```

```
int main(void)
{
    setw(10);cout << "hello" << endl;
    setfill('%');setw(10); cout << "hello" << endl ;

    cout.setf(ios::left); setw(10);cout << "hello" << endl ;
    setw(10);setprecision(4);cout << 123.1234567 << endl;
    setw(10);setprecision(3);cout << 123.1234567 << endl;

    return 0;
}
```



```
C:\Windows\system32\cmd.exe
hello
hello
hello
123.123
123.123
계속하려면 아무 키나 누르십시오 . . .
```

실습 예제

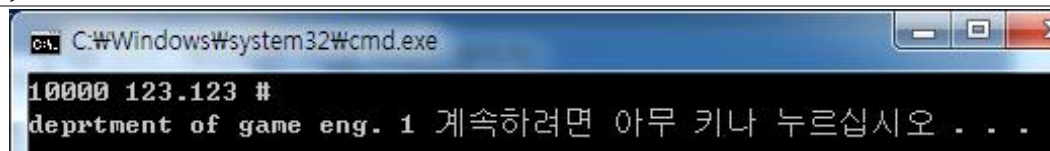
```
#include<iostream> // .h가 없음
#include<string> // string 타입을 사용하기 위함

using namespace std; // 작성 안할 경우 std::cout 식으로 일일이 붙여줘야함

int main()
{
    int inum = 10000;
    double dnum = 123.123456789;
    char nnum = '#';
    string str = "deptment of game eng.";
    bool flag = true;

    cout.setf(ios::left);cout.fill('#');
    cout << inum << " " << dnum << " " << nnum << " " <<endl;
    cout << str << " " << flag << " ";

    return 0;
}
```



```
C:\Windows\system32\cmd.exe
10000 123.123 #
deptment of game eng. 1
계속하려면 아무 키나 누르십시오 . . .
```

○ 문자열

* C 스트링 방식 : -'\0'으로 끝나는 문자 배열

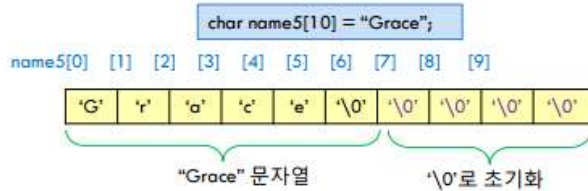
- 문자 하나하나를 넣을 때는 '\0'을 반드시 사용해야 문자열이 된다. (쓰지 않으면 단순 문자 배열)

● C++의 문자열 표현 방식 : 2가지

▶ C-스트링 방식 - '\0'로 끝나는 문자 배열

C-스트링 문자열
단순 문자 배열

```
char name1[6] = {'G', 'r', 'a', 'c', 'e', '\0'}; // name1은 문자열 "Grace"  
char name2[5] = {'G', 'r', 'a', 'c', 'e'}; // name2는 문자열이 아니고 단순 문자 배열
```



C-스트링 방식 실습 예제

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

void main(){
    char name[20];
    char password[20]="12510096 JKMeen";

    while(1){
        cout << "input password >> ";
        cin.getline(name, 30, '\n'); //\n을 할 때까지 문자열을 입력받는다.
        if (strcmp(name, password)){ //문자열을 비교, 참이면 0, 거짓이면 1
            cout << "< no match password! >" << endl << endl ;
        } else {
            cout << "< match password! >" << endl << endl;
            break;
        }
    }
}
```



* string 클래스 방식

- #include <string> 헤더 파일에 선언
- strcmp() : 스트링 값이 같으면 0 다르면 1을 리턴
- strlen() : 스트링 길이
- strcpy() : 스트링 복사
- 다양한 멤버 함수 제공, 문자열 비교, 복사, 수정 등

* cin.getline(charbuf[], int size, char delimitChar) : 공백이 낀 문자열을 입력 받는 방법

- delimitChar : 문자열을 끝낼 명령을 입력 // ex) \n

* getline(cin, singer); : string 타입의 문자열을 입력받기 위해 제공되는 전역 함수

* getchar(); : 버퍼를 비워줌

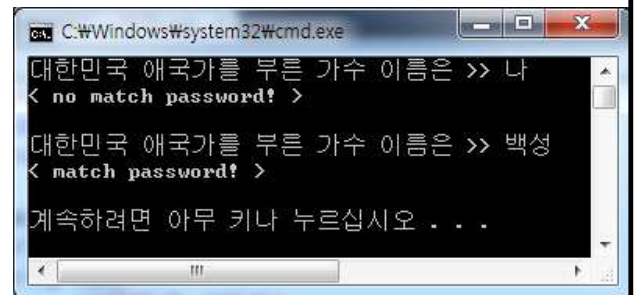
* cin.ignore(INT_MAX, '\n'); : cin에 \n을 했을 경우 버퍼를 비워줌

string 실습 예제

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

void main(){
    string song, singer;
    string name;
    song = "대한민국 애국가";
    singer = "백성";

    while(1){
        cout << song <<"를 부른 가수
이름은 >> ";
        // cin >> name; // string을
        두 가지 방법으로 입력받을 수 있다.
        getline(cin, name);
        if (singer == name){
            cout << "< match
password! >" << endl << endl;
            break;
        } else {
            cout << "< no match
password! >" << endl << endl;
        }
    }
}
```



string 실습 예제2

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

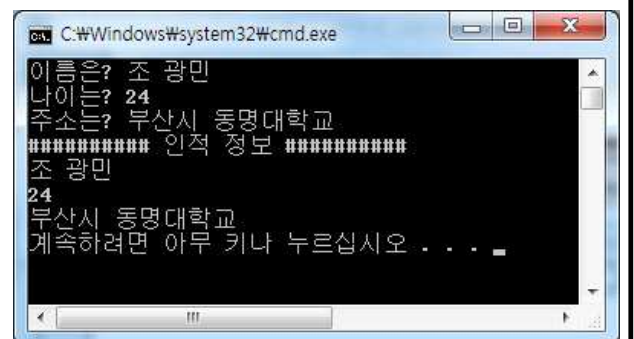
void main(){
    string name, year, home;

    cout << "이름은? ";
    getline(cin, name);

    cout << "나이는? ";
    getline(cin, year);

    cout << "주소는? ";
    getline(cin, home);

    cout << "##### 인적 정보
#####" << endl;
    cout << name << endl << year << endl
<< home << endl;
}
```



곱셈 테이블 실습 예제

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

void main(){

cout << "          곱셈 테이블          " <<
endl << "-----" <<
endl;
for (int i=1; i<=12; i++){
    for (int j=1; j<=10; j++){
        cout.width(3); cout << i*j;
    }
    cout << endl;
}

cout << "-----" <<
endl;
}
```

- * `isalpha` : 알파벳인지 체크
- * `tolower` : 대문자를 소문자로 바꿈
- * `strlen(변수)` : 배열 변수의 길이를 구함

영어 히스토그램 실습 예제

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

void main(){
    char get[10000];
    int world[26]={0};

    cout << "입력" << endl;
    cin.getline(get, 10000, ':');

    int length = strlen(get);

    for (int i=0; i<length; i++){
        if (isalpha(get[i])){
            char ch =
tolower(get[i]);
            world[ch-'a']++;
        }
    }
    int total = 0;
    for(int i=0; i<26; i++){
        total += world[i];
    }
    cout << "총 알파벳 개수는 : " << total
<< endl;

    for(int i=0; i<26; i++){
        cout << (char)('a'+i) << ": ( "
<< world[i] << " ) : ";
        for(int j=0; j<world[i]; j++){
            cout << "*";
        }
        cout << endl;
    }
}
```

□ 클래스와 객체

○ C++클래스 만들기

* **클래스** : 객체를 만드는 설계도

* **캡슐화** : 객체의 본질적인 특성, 객체를 캡슐로 싸서 그 내부를 보호하고 볼 수 없게 함
- private

* **인터페이스** : 외부에 객체의 일부분을 공개하는 것
- Tv-리모컨, 사람-눈, 코, 입, 귀, 피부
- public

* **클래스 작성**

* C++ 객체는 멤버 함수(행동)와 멤버 변수(상태)로 구성

* 클래스 선언부와 클래스 구현부로 구성

* **클래스 선언부**

* class 키워드를 이용하여 클래스 선언

* 멤버 변수와 멤버 함수 선언

- 멤버 변수는 클래스 선언 내에서 초기화할 수 없음
- 멤버 함수는 원형 형태로 선언

* 멤버에 대한 접근 권한 지정

- private, public, protected 중 하나
- 디폴트는 private
- public : 다른 모든 클래스나 객체에서 멤버의 접근이 가능함을 표시

* **클래스 구현**

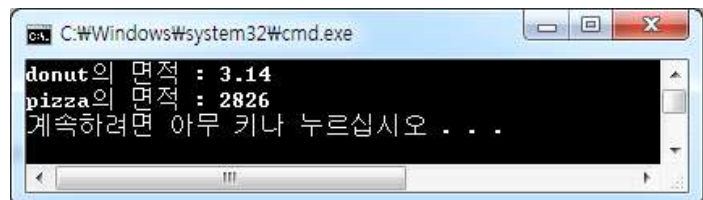
* 클래스에 정의된 모든 멤버 함수 구현

헤더파일, 클래스의 활용 예제 소스 코드

```
// Circle.h //  
class Circle{  
public:  
    int radius; // 멤버 변수  
  
public:  
    double getArea(); // 멤버 함수  
};
```

```
// Circle.cpp //  
#include "Circle.h"  
  
double Circle::getArea(){  
    return 3.14* this->radius*this->radius;  
}
```

```
// circle_main.cpp //  
#include "Circle.h"  
#include <iostream>  
using namespace std;  
  
void main(){  
    Circle donut, pizza;  
  
    donut.radius = 1;  
    pizza.radius = 30;  
  
    double area = pizza.getArea();  
  
    cout << "donut의 면적 : " << donut.getArea() << endl << "pizza의 면적 : " << area << endl;  
}
```

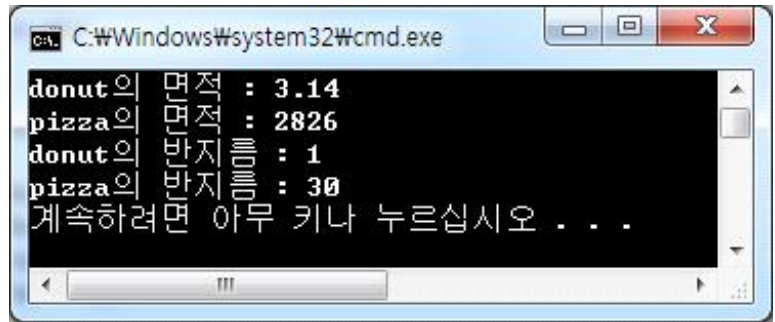


○ 객체생성과 활용

- * **헤더파일 선언유무 확인** : Circle파일을 여러 번 반복해서 포함시키더라도 if 문 때문에 바로 endif
 - * **#ifndef** CIRCLE_H : 만약 헤더파일이 선언되어있지 않으면,
 - * **#define** CIRCLE_H : 헤더파일을 포함시켜준다.
 - * **#endif** : 만약 선언되어 있으면 if를 끝낸다.
- * **setRadius, getRadius 함수를 선언하여 이용한 방법**

헤더파일, 클래스의 활용 예제 소스 코드

```
// Circle.h //  
#ifndef CIRCLE_H  
#define CIRCLE_H  
  
class Circle{  
public:  
    int radius;  
  
public:  
    double getArea();  
    void setRadius( int r );  
    int getRadius();  
};  
  
#endif
```



```
// Circle.cpp //  
#include "Circle.h"  
  
double Circle::getArea(){  
    return 3.14* this->radius*this->radius;  
}  
  
void Circle::setRadius(int r){  
    this->radius = r;  
}  
  
int Circle::getRadius(){  
    return radius;  
}
```

```
// circle_main.cpp //  
#include "Circle.h"  
#include <iostream>  
using namespace std;  
  
void main(){  
    Circle donut, pizza;  
  
    donut.setRadius(1);  
    pizza.setRadius(30);  
    donut.radius = 1;  
    pizza.radius = 30;  
  
    double area = pizza.getArea();  
  
    cout << "donut의 면적 : " << donut.getArea() << endl << "pizza의 면적 : " << area << endl;  
    cout << "donut의 반지름 : " << donut.getRadius() << endl << "pizza의 반지름 : " << pizza.getRadius() << endl;  
}
```


○ 생성자 (constructor) : 자바의 클래스 함수와 같음

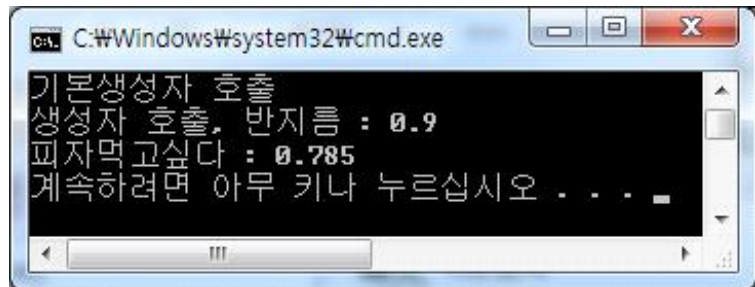
- * 객체가 생성되는 시점에서 자동으로 호출되는 멤버 함수
- * 클래스 이름과 동일한 멤버 함수
- * 생성자의 목적 : 객체가 생성될 때 객체가 필요한 초기화를 위해
 - 멤버 변수 값 초기화, 메모리 할당, 파일 열기, 네트워크 연결 등
- * 객체 생성 시 오직 한 번만 호출(자동으로 호출됨)
- * 생성자는 중복 가능
 - 한 클래스 내에 여러개 가능
 - 중복된 생성자 중 하나만 실행
- * 생성자가 선언되어 있지 않으면 기본 생성자 자동으로 생성
 - 기본 생성자 - 매개 변수가 없는 생성자
 - 컴파일러에 의해 자동 생성

생성자 예제1 소스 코드

```
// Circle.h //
#ifndef CIRCLE_H
#define CIRCLE_H
#include <iostream>
using namespace std;

class Circle{
private:
    double radius;
public:
    Circle();
    Circle(double);
    double getArea();
    void setArea( double r );
};

#endif
```



```
// Circle.cpp //
#include "Circle.h"

Circle::Circle(){
    this->radius = 1;
    cout << "기본생성자 호출\n";
}

Circle::Circle(double r){
    this->radius = r;
    cout << "생성자 호출, 반지름 : " << this->radius << endl;
}

double Circle::getArea(){
    return 3.14*this->radius*this->radius;
}

void Circle::setArea(double r){
    this->radius = r;
}
```

```
// circle_main.cpp //
#include "Circle.h"

using namespace std;

void main(){
    Circle pizza;
    pizza.setArea(0.5);
    Circle(0.9);
    cout << "피자먹고싶다 : " << pizza.getArea() << endl;
}
```

생성자 예제2 소스 코드

```
// Rect_main.cpp //
#include "Rect.h"

void main(){
    Rect();
    Rect(3);
    Rect(3, 5);
}

// Rect.cpp //
#include "Rect.h"

Rect::Rect(){
    this->width = 1;
    this->height = 1;
    cout << "Default 생성자, " << this->width << ", " << this->height << endl;
}

Rect::Rect(int w){
    this->width = w;
    this->height = w;
    cout << "매개변수1개, " << this->width << ", " << this->height << endl;
}

Rect::Rect(int w, int h){
    this->width = w;
    this->height = h;
    cout << "매개변수2개, " << this->width << ", " << this->height << endl;
}

int Rect::getArea(){
    return this->width * this->height;
}

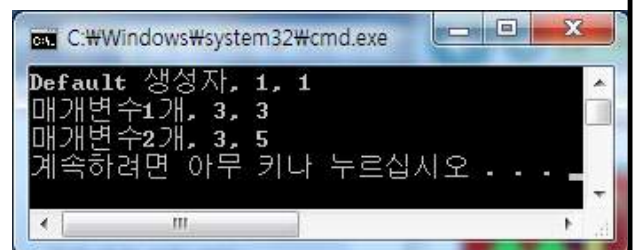
void Rect::setWidth(int w){
    this->width = w;
}

void Rect::setHeight(int h){
    this->height = h;
}
```

```
// Rect.h //
#ifndef RECT_H
#define RECT_H
#include <iostream>

using namespace std;
class Rect{
private:
    int width, height;
public:
    Rect();
    Rect(int w, int h);
    Rect(int w);
    int getArea();
    void setWidth(int w);
    void setHeight(int h);
};

#endif
```



○ **소멸자** : 객체가 소멸되는 시점에서 자동으로 호출되는 함수

* 생성자가 호출한 순서의 역순으로 소멸자가 호출된다.



* 오직 한 번만 자동 호출, 임의로 호출할 수 없음

* 객체 메모리 소멸 직전 호출됨

* **소멸자 함수의 이름은 클래스 이름 앞에 ~를 붙인다**

- ex) `Circle::~~Circle(){...}`

* 중복 불가능

* 객체가 선언된 위치에 따른 분류

- **지역 객체** : 함수 내에 선언된 객체, 함수가 종료하면 소멸

- **전역 객체** : 함수의 바깥에 선언된 객체로서, 프로그램이 종료할 때 소멸

○ **접근지정**

* **public** : 공개

* **private** : 비공개

* **protect** : 일부에게만 공개

○ **인라인 함수(Inline)** : **inline** 키워드로 선언된 함수

* 소스코드를 따로 안 만들고 헤더파일에서 구현을 안 하고, inline함수로 만들 경우, **메인 함수가 있는 파일에서 정의해야한다.**

* 인라인 함수를 호출하는 곳에 인라인 함수 코드를 확장 삽입

- 매크로와 유사, 코드 확장 후 인라인 함수는 사라짐

* **인라인 함수 호출**

- 함수 호출에 따른 오버헤드 존재하지 않음

- 프로그램의 실행 속도 개선

* 컴파일러에 의해 이루어짐

* **C++프로그램의 실행 속도 향상**

- 자주 호출되는 짧은 코드의 함수 호출에 대한 시간 소모를 줄임

- C++에는 짧은 코드의 멤버 함수가 많기 때문

* **장점** : 프로그램의 실행 시간이 빨라진다

* **단점** : 인라인 함수 코드의 삽입으로 컴파일된 전체 코드 크기 증가

과제 //////////////

연습문제 1 : 랜덤 클래스를 활용하여, 랜덤한 정수 10개 출력

랜덤 클래스의 생성자, next(), nextInRange()의 3개의 멤버 함수를 가지도록 작성

랜덤 수의 범위 : 0~32767까지

생성자 + 소멸자 + 인라인 함수 예제1 소스 코드

```
#include "Calculator.h"
```

```
void main(){
    Calculator cal;
    cal.run();
}
```

```
#include "Calculator.h"
```

```
#include "Adder.h"
```

```
#include "Sub.h"
```

```
#include "Mul.h"
```

```
#include "Div.h"
```

```
void Calculator::run(){
    cout<<"숫자 두개 입력 >>";
    int a,b;
    cin>>a>>b;
    Adder adder(a, b);
    Sub sub(a, b);
    Mul mul(a, b);
    Div div(a, b);
    cout << "덧셈 결과 : "<<adder.processor()<<endl;
    cout << "나눗셈 결과 : "<<div.processor()<<endl;
}
```

```
#ifndef CALCULATOR_H
```

```
#define CALCULATOR_H
```

```
#include <iostream>
```

```
using namespace std;
```

```
class Calculator{
public:
    void run();
};
```

```
#endif
```

```
#ifndef ADDER_H
```

```
#define ADDER_H
```

```
class Adder{
    int a,b;
public:
    Adder(int aa, int bb){
        this->a = aa; this->b = bb;
    }
    int processor(){
        return this->a + this->b;
    }
};
```

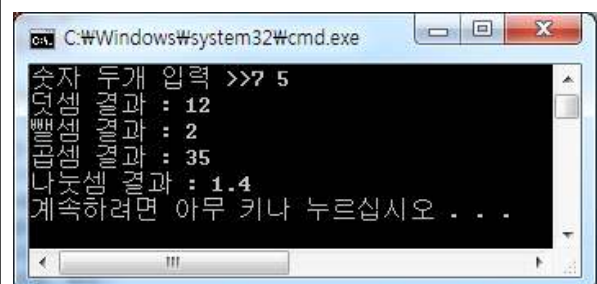
```
#endif
```

```
#ifndef DIV_H
```

```
#define DIV_H
```

```
class Div{
    double a,b;
public:
    Div(int aa, int bb){
        this->a = aa; this->b = bb;
    }
    double processor(){
        return this->a / this->b;
    }
};
```

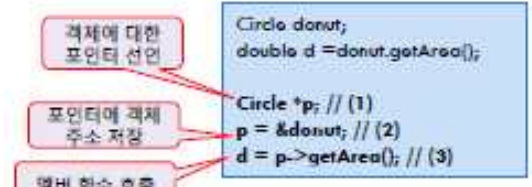
```
#endif
```



□ 객체 포인터와 객체 배열, 객체의 동적 생성

○ 객체 포인터

- * **동적 메모리 할당** : 동적으로 할당된 객체에 접근할 때 사용(힙 영역으로부터 할당)
- * **함수에서 객체 배열을 인수로 사용** : 함수의 인수로 객체 배열을 넘길 때 사용
- * **객체에 대한 포인터**
 - C언어의 포인터와 동일
 - 객체의 주소 값을 가지는 변수
- * **포인터 멤버를 접근할 때** : 객체 포인터 -> 멤버



// Circle *p처럼 포인터 변수 앞에는 변수타입이 붙는다. 그 주소를 따라가면 해당 변수타입이 있어야 한다.

- * **스택 영역** : 함수들마다 별도로 사용할 수 있는 영역, 정해진 것들만 사용
/ 프로그램이 수행(메인함수가 수행)되면 불러 졌다가 끝나면 사라짐
- * **힙 영역** : 프로그램들이 공동으로 사용하는 영역 (운영체제가 관리)

객체 포인터 예제1 소스 코드

```
#ifndef COLOR_H
#define COLOR_H
#include <iostream>
using namespace std;

class Color{
    int red, green, blue;

public:
    Color() { red = green = blue = 0; }
    Color(int r, int g, int b){
        red = r; green = g; blue = b;
    }

    void setColor(int r, int g, int b);
    void show();
};

#endif

#include "Color.h"

inline void Color::setColor( int r, int g, int b ){ // 인라인 함수
    red = r; green = g; blue = b;
}

inline void Color::show(){
    cout << "색상(red, green, blue) : " << red << " ";
    cout << green << " " << blue << endl;
}

void main(){
    Color screenColor(255, 0, 0); // 빨간색의 screenColor 객체 생성
    Color *p;
    p = &screenColor;
    /*두 가지의 형태는 같다*/
    p->show();
    (*p).show();

    Color colors[3];
    p = colors;
    p->setColor(255, 0, 0);
    p++;
    p->setColor(0, 255, 0);
    p = p+1;
    p->setColor(0, 0, 255);

    p = colors; //p[i]를 쓸 때 반드시 p를 초기값으로 바꾸고 실행해야 한다.
    for (int i=0; i<3; i++){
        /* 네가지의 경우가 다 같다 */
        (*p).show(); p++; // 첫 번째 방법
        //(p+i).show(); // 두 번째 방법
        p[i].show(); // 세 번째 방법
        colors[i].show(); // 네 번째 방법
    }
}
```

```
색상<red, green, blue> : 255 0 0
색상<red, green, blue> : 255 0 0
색상<red, green, blue> : 255 0 0
색상<red, green, blue> : 0 255 0
색상<red, green, blue> : 0 0 255
계속하려면 아무 키나 누르십시오 . . .
```

○ 객체 배열

* 2차원 배열 : ex) Color colors[3][2] // 행(세로) : 3, 열(가로) : 2

* 2차원 배열에서 포인터로 나타낼 경우

- 첫 번째 방법 : Color *p = colors[0];
- 두 번째 방법 : Color *p = &colors[0][0];

객체 포인터 예제1 소스 코드

```
#ifndef COLOR_H
#define COLOR_H
#include <iostream>
using namespace std;

class Color{
    int red, green, blue;

public:
    Color() { red = green = blue = 0; }
    Color(int r, int g, int b){
        red = r; green = g; blue = b;
    }

    void setColor(int r, int g, int b);

    void show();
};

#endif

#include "Color.h"

inline void Color::setColor( int r, int g, int b ){
    red = r; green = g; blue = b;
}

inline void Color::show(){
    cout << "색상(red, green, blue) : " << red << " ";
    cout << green << " " << blue << endl;
}

void main(){
    Color colors[3][2]={Color(255, 0, 0), Color(0, 255, 0),
                        Color(0, 0, 255), Color(0, 255, 255),
                        Color(10, 10, 10), Color(20, 20, 20)};

    //Color *p = colors[0];
    Color *p = &colors[0][0];

    for (int i =0; i<3; i++){
        for (int j=0; j<2; j++){
            //(*(p+i)).show();
            colors[i][j].show();
        }
    }
}
```

```
색상<red, green, blue> : 255 0 0
색상<red, green, blue> : 255 0 0
색상<red, green, blue> : 255 0 0
색상<red, green, blue> : 0 255 0
색상<red, green, blue> : 0 0 255
계속하려면 아무 키나 누르십시오 . . .
```


○ 정적 할당

* 변수 선언을 통해 필요한 메모리 할당

- 스택(stack)으로부터 할당
- 많은 양의 메모리는 배열 선언을 통해 할당

○ 동적 메모리 할당 및 반환 (생성자 / 소멸자)

* 프로그램이 실행된 후에 필요한 메모리를 할당 받음 : 필요한 만큼 메모리를 할당

* 실행 중에 운영체제로부터 할당 받음

- 힙(heap)으로부터 할당
- 힙은 운영체제가 소유하고 관리하는 메모리로 모든 프로세스가 공유할 수 있는 메모리

* C++의 동적 메모리 할당 / 반환 : new 연산자, delete 연산자

* new 연산자

- 기본 타입 메모리 할당, 배열 할당, 객체 할당, 객체 배열 할당
- 객체의 동적 생성 - 힙 메모리로부터 객체를 위한 메모리 할당 요청
- 객체 할당 시 생성자 호출

* delete 연산자

- new로 할당 받은 메모리 반환
- 객체의 동적 소멸 - 소멸자 호출 뒤 객체를 힙에 반환
- 배열일 경우 `delete [] p;` 형식으로 소멸

* new / delete 연산자의 사용 형식

- 반드시 포인터 타입으로 선언

```
데이터타입 *포인터변수(스택영역에 저장) = new 데이터타입(힙 영역에 저장);  
delete 포인터변수;
```

* delete 사용 시 주의사항

- 동적으로 할당 받지 않는 메모리 반환 -> 오류

```
int n;  
int *p = &n;  
delete p; // 실행 시간 오류  
// 포인터 p가 가리키는 메모리는 동적으로 할당 받은 것이 아님
```

- 동일한 메모리 두 번 반환 -> 오류

```
int *p = new int;  
delete p; // 정상적인 메모리 반환  
delete p; // 실행 시간 오류. 이미 반환한 메모리를 중복 반환할 수 없음
```

* 객체의 동적 생성 및 반환

- * default 함수는 ()를 생략해줘도 된다. ex) `Circle *p Circle(); => Circle *p Circle;`

```
클래스이름 *포인터변수 = new 클래스이름;  
클래스이름 *포인터변수 = new 클래스이름(생성자매개변수리스트);  
delete 포인터변수;
```

생성자, 소멸자 예제 소스 코드

```
#include <iostream>
using namespace std;
void main(){
    int num;
    double *pdata, sum=0;
    cout << "input data num << ";
    cin >> num;
    pdata = new double[num];
    for (int i=0; i<num; i++){
        cin >> pdata[i];
        sum += pdata[i];
    }
    cout << "-----" << endl;
    for (int i=0; i<num; i++){
        cout << "data" << i << " >> " << (*pdata+i) << endl;
        /*cout << *pdata << " ";
        pdata++;*/ // 마지막 다음주소를 가리키기 때문에
        delete [] pdata; //에서 오류가 남
    }
    cout << "-----" << endl;
    cout << "averrage >> " << sum/num << endl;
    cout << "-----" << endl;
    delete [] pdata;
}
```

```
C:\Windows\system32\cmd.exe
input data num << 5
99.9
12.2
13.7
45.4
50
-----
data0 >> 99.9
data1 >> 12.2
data2 >> 13.7
data3 >> 45.4
data4 >> 50
-----
averrage >> 44.24
계속하려면 아무 키나 누르십시오 . . .
```

생성자, 소멸자 예제 소스 코드2

```
#ifndef COLOR_H
#define COLOR_H
#include <iostream>
using namespace std;
class Color{
    int red, green, blue;
public:
    Color() { red = green = blue = 0; }
    Color(int r, int g, int b){
        red = r; green = g; blue = b;
    }
    void setColor(int r, int g, int b);
    void show();
    int getRed() { return red; }
    int getGreen() { return green; }
    int getBlue() { return blue; }
    int getMix() { return red + green + blue; }
};
#endif
```

```
C:\Windows\system32\cmd.exe
Input Color Num <<4
-----
Input R, G, B << 27 79 32
Input R, G, B << 32 44 255
Input R, G, B << 255 255 255
Input R, G, B << 0 0 150
-----
0 st Mix Color >> 138
3 st Mix Color >> 150
계속하려면 아무 키나 누르십시오 . . .
```

```
#include "Color.h"
inline void Color::setColor( int r, int g, int b ){
    red = r; green = g; blue = b;
}
inline void Color::show(){
    cout << "색상(red, green, blue) : " << red << " ";
    cout << green << " " << blue << endl;
}
void main(){
    Color *colors;
    int r, g, b;
    int num; // 칼라 갯수 입력받고 배열에 넣음, 배열안에 저장된 혼합칼라색이 100~200인것만 출력
    cout << "Input Color Num <<";
    cin >> num;
    colors = new Color[num];
    cout << "-----" << endl;
    for (int i=0; i<num; i++){
        cout << "Input R, G, B << ";
        cin >> r >> g >> b;
        colors[i].setColor(r, g, b); // 배열 함수불러올때 . 연산자
        //(*colors+i).setColor(r, g, b);
    }
    cout << "-----" << endl;
    for (int i=0; i<num; i++){
        if (colors[i].getMix() >= 100 && colors[i].getMix() <= 200){
            cout << i << " st Mix Color >> " << colors[i].getMix() << endl;
        }
    }
    cout << "-----" << endl;
    delete [] colors;
}
```

○ 객체와 객체 배열의 동적 생성 및 반환

* :

○ This 포인터

* :

○ String 클래스

* :