

목록

cpp_02_기본_ex (2).....	1
cpp_03_클래스와객체_ex (1).....	6
cpp_04_객체포인터와동적생성_ex (3).....	10
cpp_04_객체포인터와동적생성_ex_sol_08.....	15
cpp_04_객체포인터와동적생성_ex_sol_10 (1).....	17

□ 기본 개념 확인

(1) O, X로 답하세요.

- 1) C++는 C와 호환되지 않는다. ()
- 2) C++는 절차 지향 언어이다. ()
- 3) 실행 시간 효율성 저하를 최소화 하기 위하여 재귀 함수를 사용한다. ()
- 4) C++ 프로그램은 이미 컴파일 된 C언어의 목적 파일은 링크시켜 사용할 수 없다. ()
- 5) C++ 헤더 파일은 확장자를 생략한다. ()
- 6) main() 함수는 반드시 return문을 가지고 있어야 한다. ()
- 7) 이름 공간을 선언할 때 using 을 사용한다. ()
- 8) bool 타입의 변수에 저장된 값은 true 또는 false로 출력된다. ()
- 9) static_cast 연산자는 기본 자료형에 대하여 사용한다. ()
- 10) string 클래스를 사용하여 문자열을 처리할 때 이름공간은 생략해도 된다. ()

(2) 빈 괄호를 채워 넣으세요.

- 1) C++는 () 언어와의 호환성과 () 개념 추가를 목표로 설계되었다.
- 2) 작은 크기의 멤버 함수를 자주 호출할 때 발생하는 실행 시간 효율성 저하 문제는 ()함수로 해결한다
- 3) C++ 프로그램의 확장자는 () 이다.
- 4) () 헤더 파일은 표준 입출력을 위한 클래스와 객체, 변수 등이 선언되어 있다.
- 5) 이름 공간을 생략하려면 () 지시어를 사용한다.
- 6) bool 타입으로 선언된 변수에 저장된 값을 불 리터럴로 출력하려면 조작자 ()를 사용해야 한다.
- 7) C++ 에서 문자열을 처리하는 방법은 문자 배열 또는 () 클래스를 사용한다.
- 8) 공백이 포함된 문자열을 처리하려면 전역 함수 ()을 사용한다.
- 9) uniform initialization는 ()을 사용하여 변수, 배열, 객체 등을 초기화 한다.
- 10) 표준 입력 장치인 키보드로 데이터를 입력 받을 때 () 객체를 사용한다.

(3) 질문에 답하세요.

- 1) C++ 표준 라이브러리 그룹 3개를 설명하세요.
- 2) C++ 프로그램이 실행을 시작하는 함수 원형을 제시하세요.
- 3) C++ 표준에서 cin, cout 객체는 어떤 헤더 파일에 선언되어 있나요?
- 4) C++ 표준 라이브러리가 모두 선언된 이름 공간은 무엇인가요?
- 5) 이름공간을 선언할 때 사용하는 키워드는 무엇인가요?
- 6) 컴파일 때 변수에 초기화 된 값으로 형을 결정하는 자료형은 무엇인가요?
- 7) namespace hallym에 정의된 함수 software를 호출하는 문장을 쓰세요. 단, software 함수는 반환 값과 매

개변수를 가지지 않습니다.

- 8) 다음 문장에 오류가 발생하지 않도록 수정하세요. using 지시어는 사용하지 않습니다.

```
cout>>"C++";
```

- 9) 다음 중 컴파일 오류가 발생하는 문장은 무엇인가요?

① int d { 3.6 };

② int f = 3.6;

- (4) 다음 문장 중에서 틀린 부분을 올바르게 수정하세요.

① #include <iostream.h>

② using std namespace;

③ cin<<data;

④ 100>>cout;

- (5) 이름 공간을 지정하는 문장을 사용하지 않으려고 합니다. 아래의 코드를 수정하세요.

```
#include <iostream>
```

```
#include <string>
```

```
int main() {  
    string name;  
    cout << "이름을 입력하세요 : ";  
    cin >> name;  
    cout << "name : " << name << endl;  
    return 0;  
}
```

- (6) 제시된 프로그램에 대하여 질문에 답하세요.

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int main() {  
    string name;  
    cout << "이름을 입력하세요 : ";  
    cin >> name;  
    cout << name << " 님 환영합니다 " << endl;  
    return 0;  
}
```

- 6-1) 다음과 같이 키보드로 "C++"을 입력하였을 때 결과를 제시하세요.

이름을 입력하세요? C++

6-2) 다음과 같이 키보드로 "C++ OOP"을 입력하였을 때 결과를 제시하세요.

이름을 입력하세요? C++ OOP

6-3) 공백이 포함된 문자열을 입력 받을 수 있도록 위의 프로그램을 수정 하세요. 단, 수정한 문장만 제시합니다.

(7) 컴파일 오류가 발생하지 않도록 빈칸을 채워 넣으세요.

```
#include <iostream>
```

```
int main()
{
    int count;
    cout << "반복 횟수를 입력하세요 : ";
    std::cin >> count;
    cout << count << " 회 반복합니다. "<<endl;
    return 0;
}
```

(8) 다음 프로그램을 cpp로 수정한 소스와 결과를 함께 제시하세요. auto변수로 변환 가능한 곳은 모두 변경하세요.

```
#include <stdio.h>
int sum(int s, int e); // 함수 원형 선언

int sum(int s, int e)
{
    int tmp, res = 0;

    if(s>e) {
        tmp = e;
        e = s;
        s = tmp;
    }
    for (int k = s; k <= e; k++)
    {
        res += k;
    }
    return res;
}
```

```

}
int main()
{
    int a, b;
    printf("첫 수와 마지막 수를 입력하세요 : ");
    scanf("%d %d", &a, &b);

    printf("%d부터 %d까지의 합은 %d 입니다\n", a, b, sum(a, b));

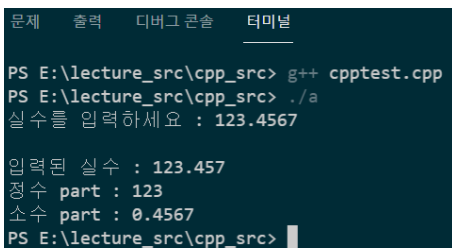
    return 0;
}

```

[프로그램 소스]

[실행 결과]

(9) 입력된 실수에서 정수 부분과 소수점 아래부분을 분리해서 출력하는 프로그램을 작성하세요



```

문제   출력   디버그 콘솔   터미널
PS E:\lecture_src\cpp_src> g++ cpptest.cpp
PS E:\lecture_src\cpp_src> ./a
실수를 입력하세요 : 123.4567

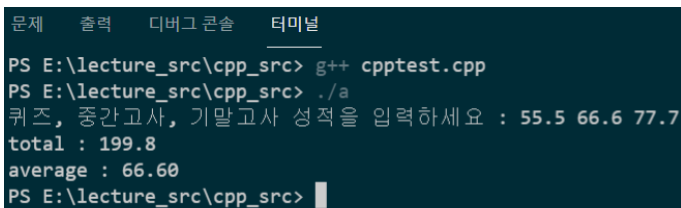
입력된 실수 : 123.457
정수 part : 123
소수 part : 0.4567
PS E:\lecture_src\cpp_src>

```

[프로그램 소스]

[실행 결과]

(10) 퀴즈, 중간고사, 기말고사의 성적을 사용자로부터 입력 받아 총합과 평균을 계산하는 프로그램을 작성하세요. 단, 이름공간은 선언하지 않습니다.



```

문제   출력   디버그 콘솔   터미널
PS E:\lecture_src\cpp_src> g++ cpptest.cpp
PS E:\lecture_src\cpp_src> ./a
퀴즈, 중간고사, 기말고사 성적을 입력하세요 : 55.5 66.6 77.7
total : 199.8
average : 66.60
PS E:\lecture_src\cpp_src>

```

[프로그램 소스]

[실행 결과]

(cpp_02_기본 슬라이드 마지막 페이지 실습 - 기본 입출력 활용) 입력한 10진 정수를 다양한 진법으로 출력하기 : 제어문 & 조작자 & 문자열 활용.

```
문제   출력   디버그 콘솔   터미널

PS E:\lecture_src\cpp_src> g++ cpptest.cpp
PS E:\lecture_src\cpp_src> ./a
10진수 입력 : 55

여러 진법으로 출력 하기
oct(8), hex(16), digit(10)
해당 진법 입력 : 16
=> 16진법 : 37
PS E:\lecture_src\cpp_src> ./a
10진수 입력 : 55

여러 진법으로 출력 하기
oct(8), hex(16), digit(10)
해당 진법 입력 : hex
=> 16진법 : 37
PS E:\lecture_src\cpp_src> █
```

[프로그램 소스]

[실행 결과]

•(cpp_02_기본 슬라이드 마지막 페이지 실습 - initializer_list 활용) 두번째 인수와 가장 가까운 거리에 있는 문자 출력하기.

```
int main() {
    cout << "{ 'd', 'p', 'r', 'w', 'g', 'f' }문자 중 h와 가까운 문자는 : ";
    cout << list_exam({ 'd', 'p', 'r', 'w', 'g', 'f' }, 'h') << endl;

    cout << "{ 'k', 'q', 'b', 'r', 'a', 'e', 'v', 'z'}문자 중 w와 가까운 문자는 : ";
    cout << list_exam({ 'k', 'q', 'b', 'r', 'a', 'e', 'v', 'z'}, 'w') << endl;
}
```

```
{ 'd', 'p', 'r', 'w', 'g', 'f' }문자 중 h와 가까운 문자는 : g
{ 'k', 'q', 'b', 'r', 'a', 'e', 'v', 'z'}문자 중 w와 가까운 문자는 : v
```

[프로그램 소스]

[실행 결과]

□ 기본 개념 확인

(1) 빈 괄호를 채워 넣으세요.

- ① ()란 객체를 만들기 위해 정의된 설계도이다.
- ② C++에서 클래스는 클래스 선언부와 클래스 ()로 구성된다.
- ③ 클래스 선언 시 멤버에 대한 접근 권한이 생략되면 디폴트는 ()이다.
- ④ ()는 객체가 생성되는 시점에서 자동으로 호출되는 멤버함수이다.
- ⑤ 생성자가 선언되어 있지 않으면 ()에 의해 기본 생성자가 자동으로 생성된다.
- ⑥ 클래스 멤버의 디폴트 접근 권한은 ()이며 구조체 디폴트 접근 권한은 ()이다.
- ⑦ 한정자 ()를 사용하여 멤버를 상수화 할 수 있다.
- ⑧ ()는 함수 호출에 따른 오버헤드를 줄이기 위해 함수를 호출하는 곳에 함수의 코드를 삽입한다.
- ⑨ 함수를 인라인으로 하려면 키워드 ()을 사용한다.
- ⑩ () 키워드는 상수화 된 대상에 대한 쓰기 작업을 허용하기 위한 목적으로 사용한다.

(2) 다음 질문에 O, X로 답하세요.

- ① 소멸자는 여러 번 구현할 수 있다. ()
- ② 생성자는 반환값이 없다. ()
- ③ 프로그램이 종료하면 객체가 생성된 순서로 소멸된다. ()
- ④ private 접근 권한은 생성자에 사용할 수 없다. ()
- ⑤ 인라인 함수는 inline 키워드로 선언된 함수이다. ()
- ⑥ 생성자는 객체 생성 시 한 번만 호출할 수 있다. ()
- ⑦ 소멸자는 매개변수를 가질 수 없다. ()
- ⑧ 생성자와 소멸자의 실행 순서는 동일하다. ()
- ⑨ 인라인 선언은 크기가 큰 함수의 경우 효과적이다. ()
- ⑩ 인라인 함수를 사용하면 전체 프로그램의 크기가 작아져서 효과적이다. ()

(3) 질문에 답하세요.

- ① 다음과 같은 생성자와 소멸자가 있다고 할 때 어떤 것이 디폴트 생성자인지, 소멸자인지, 복사 생성자인지 구분하세요.

```
Account ();  
~Account ();  
Account (const Account & acc);
```

- ② 다음과 같이 Rectangle 클래스의 생성자를 선언하면 오류가 발생합니다. 이유를 설명하세요.

```
int Rectangle (int length, int height);
```

- ③ 다음과 같이 Rectangle 클래스의 소멸자를 선언하면 오류가 발생합니다. 이유를 설명하세요.

```
int ~Rectangle (int var);
```

- ④ 다음과 같은 생성자를 초기화 리스트를 사용하는 코드로 변경하세요.

```
Rectangle :: Rectangle (int len, int wid){
    length = len;
    height = wid;
}
```

- ⑤ 다음과 같은 클래스 정의가 있을 때 객체 생성 코드 중 오류가 발생하는 라인을 제시하세요.

```
class Sample {
private :
    int x;
public:
    int getX() const;
};

int main() {
    Sample s1(4);
    Sample s2();
    Sample s3{};
}
```

- ⑥ 다음 클래스 선언에서 오류가 발생하는 코드를 수정하세요.

```
class Object{
    double x;

public;

    double const getX();
}
```

- ⑦ value가 클래스의 데이터 멤버라고 할 때, 다음 클래스의 멤버 함수 정의에서 발생하는 오류를 수정 하세요.

```
Member::int getValue(){
    return value;
}
```

- ⑧ 다음과 같은 클래스 선언에 대하여 생성자와 접근자 함수를 구현 하세요

```
class Hallym{
    string dept;
public:
    Hallym(string d);    //매개 변수값을 멤버 변수로 초기화
    string getDept();    //멤버 변수 값 반환
}
```


(4) 다음과 같은 멤버를 갖는 Person 클래스를 선언, 구현, 애플리케이션 코드로 각각 분리하여 프로그램을 작성하세요.

- 데이터 멤버 - name과 age
- 접근자 멤버 함수 - getName과 getAge
- 설정자 멤버 함수 - setName과 setAge
- 매개변수가 있는 생성자와 소멸자

(cpp_03_클래스와객체 슬라이드 마지막 페이지 실습1) - 난수생성하기

- 주어진 큰 값과 작은 값 범위 내의 난수 생성하기
- 객체 생성시 난수의 범위를 매개변수로 전달 - 매개변수 있는 생성자 필요
- 난수 생성 결과를 출력하는 멤버 함수 print() - 변경 작업이 필요 없으므로 const

//C++ 난수 생성 방법

```
#include <random>
```

```
random_device rd; //시드값을 얻기 위한 random_device 생성
```

```
mt19937 gen(rd()); //random_device를 통해 난수 생성 엔진 초기화
```

```
uniform_int_distribution<int> dis(low, high); //low~high 사이의 난수 및 분포 정의
```

```
Int value = dis(gen); //난수 엔진을 전달하여 범위 내 생성된 난수를 value에 저장
```

```
PS E:\lecture_src\cpp_src> g++ randint.cpp randintmain.cpp
PS E:\lecture_src\cpp_src> ./a
Random number between 100 and 200 : 156
Random number between 400 and 600 : 512
Random number between 1500 and 2000 : 1781
r3객체 소멸
r2객체 소멸
r1객체 소멸
PS E:\lecture_src\cpp_src> □
```

```
int main() {
    RandInt r1{ 100, 200, "r1" };
    r1.print();
    RandInt r2(400, 600, "r2");
    r2.print();
    RandInt r3(1500, 2000, "r3");
    r3.print();
    return 0;
}
```

(cpp_03_클래스와객체 슬라이드 마지막 페이지 실습2) - 계좌 관리

- 제시된 main()소스코드를 참조하여 계좌 관리를 위한 클래스 Account를 구현하세요.
- 단, 클래스 구현과 선언은 분리하여 작성합니다.

```
//account.h
enum class MENU { DEPOSIT = 1, WITHDRAW, CHECK };
:
:
```

```
//accountmain.cpp
#include <iostream>
#include "account.h"

using namespace std;
using Bank::Account;
using Bank::MENU;

int main() {
    Account a("C++", 50000);
    int menu, money;
    cout << "menu : 1. 입금, 2. 출금 3. 조회 >> ";
    cin >> menu;
    switch (menu)
    {
        case (static_cast<int> (MENU::DEPOSIT)):
            cout << "입금액 >> ";
            cin >> money;
            a.deposit(money);
            cout << a.getOwner() << "의 입금 액은 " << money << endl;
            cout << a.getOwner() << "의 잔액은 " << a.check() << endl;
            break;
        case (static_cast<int> (MENU::WITHDRAW)):
            cout << "출금액 >> ";
            cin >> money;
            cout << a.getOwner() << "의 출금 액은 " << a.withdraw(money) << endl;
            cout << a.getOwner() << "의 잔액은 " << a.check() << endl;
            break;
        case (static_cast<int> (MENU::CHECK)):
            cout << a.getOwner() << "의 잔액은 " << a.check() << endl;
    }
    return 0;
}
```

□ 개념 확인

(1) 빈 괄호를 채워 넣으세요.

- ① () 연산자는 변수의 주소를 추출하기 위해 사용한다.
- ② 동적으로 할당된 메모리를 반환하고자 할 때는 () 연산자를 사용한다.
- ③ () 연산자는 할당되는 동적 메모리의 시작 주소를 반환한다.
- ④ 동적으로 할당되는 메모리는 () 영역에 할당 받는 메모리이다.
- ⑤ new를 이용하여 할당 받은 메모리를 해제 할 때에는 ()를 사용한다.
- ⑥ 객체 포인터로 멤버를 접근할 때에는 () 연산자를 사용한다.
- ⑦ ()은 null pointer를 의미하는 것으로 NULL 매크로 사용시 함수 매개변수로 전달하는 경우 int타입으로 추론되는 문제점을 해결할 수 있다.
- ⑧ new 연산자를 사용하여 객체를 동적으로 할당할 때 ()가 호출된다.
- ⑨ 동적으로 할당 된 객체 소멸 시 ()가 호출된다.
- ⑩ 함수 선언 시 ()를 사용하면 멤버 변수의 값을 변경할 수 없다.
- ⑪ 스마트 포인터 중 ()는 포인터를 공유할 수 없다.
- ⑫ 스마트 포인터를 사용하려면 () 헤더 파일이 필요하다.

(2) 다음 질문에 O, X로 답하세요.

- ① 동적 메모리 할당을 위해 new 함수를 사용한다. ()
- ② 배열은 동적 할당 시 초기화를 할 수 없다. ()
- ③ 동적 메모리 반환 순서는 생성 순서와 동일해야 한다. ()
- ④ 객체 배열 생성 시 기본 생성자를 호출한다. ()
- ⑤ 동적으로 배열을 생성하면서 초기화 할 때 배열 크기는 생략할 수 있다. ()
- ⑥ 객체 포인터 변수는 초기화 없이 사용할 수 있다. ()
- ⑦ delete 연산자는 정적으로 할당된 메모리를 해제할 때도 사용할 수 있다. ()
- ⑧ 배열 형태로 동적 생성한 것은 배열 형태로 삭제해야 한다. ()
- ⑨ 클래스 멤버 변수에 대한 동적 생성은 생성자에서 할당하고, 소멸자에는 멤버 변수에 대한 동적 메모리를 해제 한다. ()
- ⑩ 스마트 포인터는 할당된 메모리를 자동으로 해제한다. ()
- ⑪ 스마트 포인터는 메모리누수와 같은 문제를 해결하기 위해 사용한다. ()
- ⑫ this는 전역 함수에서 사용할 수 있다. ()
- ⑬ this는 static 멤버 함수에서 사용할 수 없다. ()
- ⑭ this는 컴파일러가 삽입해주는 전역변수이다. ()
- ⑮ 스마트포인터는 매개변수로 전달할 수 있다. ()

(3) 제시된 클래스에 대하여 질문에 답하세요

```
class Rec{
    int w, h;
public:
    int getW();
```

```

    int getH();
    Rec(){ }
    Rec(int a, int b) : w(a), h(b){}
    void write();
};

int main(){
    Rec r(3,4);
}

```

- ① Rec 클래스에 대한 포인터 변수 p를 선언하세요.
- ② 선언된 포인터 변수 p에 객체 r의 주소를 지정하세요.
- ③ 포인터 변수 p를 이용하여 write 함수를 호출할 수 있는 두 가지 방법을 제시하세요.
- ④ 크기가 4인 Rec 객체 배열 arr를 동적으로 생성하는 문장을 new연산자를 사용하여 제시하세요.
- ⑤ 크기가 4인 Rec 객체 배열 arr를 동적으로 생성하는 문장을 공유가 허락되지 않는 스마트 포인터를 사용하여 제시하세요.
- ⑥ 4번에서 new 연산자로 할당 받은 동적메모리를 반환하는 문장을 제시하세요.
- ⑦ 4번에서 생성된 배열에서, 배열 원소 두번째에 저장된 객체의 write() 멤버 함수를 호출하는 문장을 두 가지 방법으로 제시하세요. 단, 배열 원소 참조 시 []는 사용하지 않습니다.
- ⑧ 크기가 3인 Rec 객체 배열 dim을 동적으로 할당하면서 매개변수가 있는 생성자를 사용하여 초기화하는 문장을 제시하세요. 단, 초기화 값은 본인이 임의로 결정합니다.
- ⑨ 다음과 같이 초기화 된 객체 배열을 사용하여 멤버 함수 write()를 호출하는 문장을 제시하세요. 단, 범위 기반 for를 사용하세요.
Rec array[] = { Rec(13,6), Rec(5,8), Rec(3,12) };

(4) 다음 프로그램의 실행 결과를 제시하세요.

```
#include <iostream>
using namespace std;
void fun (int* x){
    cout << *(x + 2);
}
int main (){
    int sample[] = {0, 10, 20, 30, 40};
    fun (sample);
    return 0;
}
```

(5) 다음 코드의 실행 결과를 제시하세요.

```
int sample [5] = {5, 10, 15, 20, 25};
cout << *sample + 2 << endl;
cout << *(sample + 2);
```

(6) 스마트 포인터 종류에 대하여 설명하세요.

(7) 아래 설명에 해당하는 배열을 생성하고, 실행 화면처럼 동작하는 프로그램을 작성하세요.

- 사용자로부터 배열의 크기를 입력 받아, 실수를 저장할 수 있는 배열을 동적 할당 받습니다.
- 이후, 배열의 크기만큼 값을 입력 받아, 입력 받은 값으로 배열을 초기화합니다.
- 동적 배열의 생성은 vector를 이용하는 방법과 new를 이용하는 방법 모두 구현합니다.

```
array size? 3
== vector array ==
value? 1.1
value? 1.2
value? 1.3
varr[0]=1.1
varr[1]=1.2
varr[2]=1.3
== new array ==
value? 2.1
value? 2.2
value? 2.3
narr[0]=2.1
narr[1]=2.2
narr[2]=2.3
```

(8) 주어진 메인 함수와 실행 화면을 참고하여 배열을 조금 더 쉽게 사용할 수 있는 Array 클래스를 작성하고, 실행 화면처럼 동작하는 프로그램을 작성하세요.

- Array 클래스는 데이터 멤버로 capacity, size, arr을 갖습니다.
 - capacity(데이터를 담을 수 있는 최대 용량),
 - size(데이터가 저장된 수),
 - arr(힙에 생성한 배열의 첫 번째 요소를 가리키는 포인터)
- 배열 마지막 위치에 요소를 추가하는 멤버 함수 insert()를 갖습니다. capacity를 넘어서 요소를 추가할 경우에는 '배열이 가득 차 요소를 추가할 수 없다'는 메시지를 출력합니다.
- 배열의 요소를 출력하는 멤버 함수 print()도 갖습니다.

```
array size: 4
arr = 10 11 12 13
데이터 34는 추가할 수 없습니다. 배열이 가득 찼습니다
```

```
int main(){
    int count;
    cout << "array size? ";
    cin >> count;

    Array array1(count); //크기가 count 인 배열 생성

    for (int i = 0; i < count; i++) { array1.insert(i+10); }
    array1.print();
    array1.insert(34);
    cout << endl;
    return 0;
}
```

(9) 제시된 메인 함수와 실행 화면을 참고하여 Person 클래스를 작성하세요.

```
p1) name = benny, age = 17
main() : p1.use_count() : 1
p1) name = daniel, age = 17
p2) name = daniel, age = 17
main() : p1.use_count() : 2
메모리 해제
```

```
#include <memory>
#include <iostream>
class Person {
};
int main() {
    auto p1 = std::make_shared<Person>("benny", 17);
    p1->display("p1");
    std::cout << "main() : p1.use_count() : " << p1.use_count() << std::endl;

    std::shared_ptr<Person> p2 = p1;
    p2->changeName("daniel");
    p1->display("p1");
    p2->display("p2");
    std::cout << "main() : p1.use_count() : " << p1.use_count() << std::endl;
}
```

(10) 다음 제시된 클래스를 사용하여 실행 화면과 같이 동작하는 프로그램을 작성하세요. Pizza 객체 배열은 입력 받은 피자 판의 개수만큼 new를 이용하여 동적으로 생성합니다.

```
피자 몇 판? 3
피자 크기는(small, medium, large)? small

0) small Pizza Yammy
1) small Pizza Yammy
2) small Pizza Yammy

소셜자 I Had it all.
소셜자 I Had it all.
소셜자 I Had it all.
```

```
class Pizza {
    string *size;
public:
    Pizza() = default;
    ~Pizza();
    void setSize(string s); //s를 size에 대입
    string getSize();
};
```

//처리 하는 방식은 두 가지 입니다.
//두 방식 모두 Pizza 클래스는 같습니다.

- 1) Pizza 클래스 완성 후 실행 화면의 모든 내용을 main() 함수에서 작성하는 방식.
- 2) Pizza 클래스 완성 후 main()에서 처리하던 모든 내용을 PizzaManager 클래스에 넣어 작성하는 방식.

```
int main() {
    PizzaManager pm;
    pm.status(); //Pizza 클래스 타입의 배열의 각 요소마다 getSize() 호출.
    return 0;
}
```

-

8) 주어진 메인 함수와 실행 화면을 참고하여 배열을 조금 더 쉽게 사용할 수 있는 Array 클래스를 작성하고, 실행 화면처럼 동작하는 프로그램을 작성하세요.

- Array 클래스는 데이터 멤버로 capacity, size, arr을 갖습니다.
 - capacity(데이터를 담을 수 있는 최대 용량),
 - size(데이터가 저장된 수),
 - arr(힙에 생성한 배열의 첫 번째 요소를 가리키는 포인터)
- 배열 마지막 위치에 요소를 추가하는 멤버 함수 insert()를 갖습니다. capacity를 넘어서 요소를 추가할 경우에는 '배열이 가득 차 요소를 추가할 수 없다'는 메시지를 출력합니다.
- 배열의 요소를 출력하는 멤버 함수 print()도 갖습니다.

```
array size? 4
arr = 10 11 12 13
데이터 34는 추가할 수 없습니다. 배열이 가득 찼습니다.
```

```
int main(){
    int count;
    cout << "array size? ";
    cin >> count;

    Array array1(count); //크기가 count 인 배열 생성

    for (int i = 0; i < count; i++){
        array1.insert(i+10);
    }
    array1.print();
    array1.insert(34);
    cout << endl;
    return 0;
}
```

=

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



```

class Array{
private:
    int * arr;
    int capacity;
    int size;
public:
    Array(int capacity); //매개변수로 받은 값을 크기로 갖는 배열을 생성
    ~Array();
    void insert(int value);
    void print() const;
};

// Constructor
Array::Array(int cap) : capacity(cap) {
    arr = new int[capacity];
    size = 0;
}

// Destructor
Array::~~Array(){
    delete [] arr;
}

// the insert member function
void Array::insert(int value){
    if (size == capacity){
        cout << "데이터 " << value << "는 추가할 수 없습니다. 배열이 가득 찼습니다."
<< endl;
        return;
    }
    arr[size] = value;
    size++;
}

// The print member function
void Array::print() const{
    cout << "arr = ";
    for (int i = 0; i < size; i++) {
        cout << setw(4) << arr[i] << " ";
    }
    cout << endl;
}

int main(){
    int count;
    cout << "array size? ";
    cin >> count;

    Array array1(count); //크기가 count 인 배열 생성

    for (int i = 0; i < count; i++){
        array1.insert(i+10);
    }
    array1.print();
    array1.insert(34);
    cout << endl;
    return 0;
}

```

10) 제시된 클래스를 사용하여 실행 화면과 같이 동작하는 프로그램을 작성하세요. **Pizza** 객체 배열은 입력 받은 피자 판의 개수만큼 **new** 를 이용하여 동적으로 생성합니다.

```
피자 몇 판? 3
피자 크기는(small, medium, large)? small

0) small Pizza Yammy
1) small Pizza Yammy
2) small Pizza Yammy

소셜자 I Had it all.
소셜자 I Had it all.
소셜자 I Had it all.
```

```
class Pizza {
    string *size;
public:
    Pizza() = default;
    ~Pizza();
    void setSize(string s); //s 를 size 에 대입
    string getSize();
};
```

//처리 하는 방식은 두 가지 입니다.
//두 방식 모두 Pizza 클래스는 같습니다.

- 1) Pizza 클래스 완성 후 실행 화면의 모든 내용을 main() 함수에서 작성하는 방식.
- 2) Pizza 클래스 완성 후 main()에서 처리하던 모든 내용을 PizzaManager 클래스에 넣어 작성하는 방식.

```
int main() {
    PizzaManager pm;
    pm.status(); //Pizza 클래스 타입의 배열의 각 요소마다 getSize() 호출.
    return 0;
}
```

```

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

//== main() 에서 처리 ==

class Pizza {
    string *size;
public:
    Pizza() = default;
    ~Pizza();
    void setSize(string s); //s 를 size 에 대입
    string getSize();
};
Pizza::~~Pizza() {
    delete size;
    cout << "소멸자 I Had it all." <<endl;
}
void Pizza::setSize(string s) {
    Pizza::size = new string(s);
}
string Pizza::getSize() {
    return *Pizza::size;
}
int main() {
    int count;
    Pizza *p;
    string sml;

    cout << "피자 몇 판? ";
    cin >> count;
    p = new Pizza[count]; //객체 생성 - 디폴트생성자

    cout << "피자 크기는(small, medium, large)? ";
    cin >> sml;
    for (int i = 0; i < count; i++) {
        p[i].setSize(sml);
    }
    cout << endl;
    for (int i = 0; i < count; i++) {
        cout << i << ") " << p[i].getSize() << " Pizza Yammy" << endl;
    }
    cout << endl;

    delete[] p; //멤버 메모리 해제
    return 0;
}

//== PizzaManager 사용 ==

```

```

class Pizza {
    string *size;
public:
    Pizza() = default;
    ~Pizza();
    void setSize(string s); //s 를 size 에 대입
    string getSize();
};
Pizza::~~Pizza() {
    delete size;
    cout << "소멸자 I Had it all." <<endl;
}
void Pizza::setSize(string s) {
    Pizza::size = new string(s);
}
string Pizza::getSize() {
    return *Pizza::size;
}
class PizzaManager {
    int count;
    Pizza *p;
    string sml;
public:
    PizzaManager() {
        cout <<"피자 몇 판? ";
        cin >> count;
        p = new Pizza[count]; //객체 생성 - 디폴트생성자`

        cout << "피자 크기는(small, medium, large)? ";
        cin >> sml;
        for (int i = 0; i < count; i++) {
            p[i].setSize(sml);
        }
    }
    void status() {
        //Pizza 클래스 타입의 배열의 각 요소마다 getSize() 호출
        for (int i = 0; i < count; i++) {
            cout << i << ") " << p[i].getSize() << " Pizza Yammy" << endl;
        }
        cout << endl;
    }
    ~PizzaManager() {
        delete[] p; //멤버 메모리 해제
    }
};
int main() {
    PizzaManager pm;
    pm.status(); //Pizza 클래스 타입의 배열의 각 요소마다 getSize() 호출
    return 0;
}

```