cpp_05_객체포인터와 동적생성

□ 개념 확인 학습

1. 다음 질문에 O, X로 답하세요.

A. 포인터로 멤버를 접근 할 때 객체 "포인터->멤버변수명"을 사용한다. ( )

B. "포인터->멤버변수명"와 같은 의미는 "*(객체포인터.멤버변수명)"이다. ( )

C. nullptr로 초기화한 포인터는 아무것도 가리키고 있지 않음을 의미한다. ( )

D. 객체 배열 선언 시에는 각 원소인 객체마다 매개변수 없는 디폴트 생성자를 호출한다. ( )

E. 객체 배열을 초기화 하면서 생성 할 때에는 생성자를 지정할 수 있다. ( )

F. 동적 메모리 할당을 위해 new 연산자를 사용한다. ( )

A. Cat *dim = new Cat[3] { Cat("a",4), Cat("b",3), Cat("f",5) };는 가능하다. ( )

G. 동적으로 배열을 생성하면서 초기화 할 때 배열 크기는 생략할 수 있다. ( )

H. 동적 메모리 반환 순서는 생성 순서와 동일해야 한다. ( )

I. 객체 포인터 변수는 초기화 없이 사용할 수 있다. ( )

J. delete 연산자는 정적으로 할당된 메모리를 해제할 때도 사용할 수 있다. ( )

K. 배열 형태로 동적 생성한 것은 배열 형태로 삭제해야 한다. ( )

L. 클래스 멤버 변수에 대한 동적 생성은 생성자에서 할당하고 소멸자에는 멤버 변수에 대한 동적 메모리를 해제 한다. ( )

M. 스마트 포인터는 할당된 메모리를 자동으로 해제한다. ( )

N. 스마트 포인터는 메모리누수와 같은 문제를 해결하기 위해 사용한다. ( )

O. this는 전역 함수에서 사용할 수 있다. ( )

P. this는 static 멤버 함수에서 사용할 수 없다. ( )

Q. this는 컴파일러가 삽입해주는 전역변수이다. ( )

R. 스마트포인터는 매개변수로 전달할 수 있다. ( )

2. 빈 괄호를 채워 넣으세요.

A. ( ) 연산자는 변수의 주소를 추출하기 위해 사용한다.

B. 동적으로 할당된 메모리를 반환하고자 할 때는 ( ) 연산자를 사용한다.

C. ( ) 연산자는 할당되는 동적 메모리의 시작 주소를 반환한다.

D. 동적으로 할당되는 메모리는 ( ) 영역에 할당 받는 메모리이다.

E. new를 이용하여 할당 받은 메모리를 해제 할 때에는 ( )를 사용한다.

F. 객체 포인터로 멤버를 접근할 때에는 ( ) 연산자를 사용한다.

G. ( )은 null pointer를 의미하는 것으로 NULL 매크로 사용 시 함수 매개변수로 전달하는 경우 int 타입으로 추론되는 문제점을 해결할 수 있다.

H. new 연산자를 사용하여 객체를 동적으로 할당할 때 ( )가 호출된다.

I. 동적으로 할당 된 객체 소멸 시 ( )가 호출된다.

J. 함수 선언 시 ( )를 사용하면 멤버 변수의 값을 변경할 수 없다.

K. 스마트 포인터 중 ( )는 포인터를 공유할 수 없다.

L. 스마트 포인터를 사용하려면 ( )헤더 파일이 필요하다.

3. 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;
}
```

4. 다음 코드의 실행 결과를 제시하세요.

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

5. 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);
}
```

A. Rec 클래스에 대한 포인터 변수 p를 선언한 후 p에 객체 r의 주소를 지정하세요.

B. 포인터 변수 p를 이용하여 write 함수를 호출할 수 있는 두 가지 방법을 제시하세요.

C. 크기가 4인 Rec 객체 배열 arr를 동적으로 생성하는 문장을 new연산자를 사용하여 제시하세요.

D. 크기가 4인 Rec 객체 배열 arr를 동적으로 생성하는 문장을 공유가 허락되지 않는 스마트 포인터를 사용하여 제시하세요.

E. 위 D번에서 new 연산자로 할당 받은 동적메모리를 반환하는 문장을 제시하세요.

F. D번에서 생성된 배열 arr에서 배열 원소 두 번째에 저장된 객체의 write() 멤버 함수를 호출하는 문장을 두 가지 방법으로 제시하세요. 단, 배열 원소 참조 시 []는 사용하지 않습니다.

G. 크기가 3인 Rec 객체 배열 dim을 동적으로 할당하면서 매개변수가 있는 생성자를 사용하여 초기화하는 문장을 제시하세요. 단, 초기화 값은 본인이 임의로 결정합니다.

H. Rec array[] = { Rec(13,6), Rec(5,8), Rec(3,12) };와 같이 초기화 된 객체 배열을 사용하여 멤버 함수 wirte()를 호출하는 문장을 제시하세요. 단, 범위 기반 for를 사용하세요.

6. 스마트 포인터의 종류와 특징에 대하여 설명하세요.

□ 응용 프로그래밍

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 실수 값을 입력 받아 배열에 저장합니다.
- 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 main() 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)

- Array 클래스는 배열 마지막 위치에 데이터를 추가하는 멤버 함수 insert()를 갖습니다.
- capacity를 넘겨 데이터를 추가하려고 할 경우에는 '배열이 가득 차 추가할 수 없다'는 메시지를 출력합니다.
- 배열의 내용을 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 count인 배열 생성

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

9. 주어진 main() 함수와 실행 화면을 참고하여 Person 클래스를 완성하세요.

```
#include <memory>
#include <iostream>

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;
}
```

```
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
```

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

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

```
PS C:\yanges\lecture\lecture_src\cpp> g++ ctest.cpp
PS C:\yanges\lecture\lecture_src\cpp> ./a
피자 몇 판? 3
피자 크기는(small, medium, large)? large
0) large Pizza Yammy
1) large Pizza Yammy
2) large Pizza Yammy

소멸자 I had it all.
소멸자 I had it all.
소멸자 I had it all.
PS C:\yanges\lecture\lecture_src\cpp>
```

```
//처리 하는 방식은 두 가지입니다. 두 방식 모두 Pizza 클래스는 같습니다.

1. Pizza 클래스 완성 후 실행 화면의 모든 내용을 main() 함수에서 작성하는 방식.
    class Pizza {
        :
    };
    int main() {
        :
    }

2. Pizza 클래스 완성 후 main()에서 처리하던 모든 내용을 PizzaManager 클래스에 넣어 작성하는 방식. (cpp에서 선호하는 방식)
    class Pizza {
        :
    };
    class PizzaManager {
        :
    };
    int main() {
        PizzaManager pm;
        pm.status(); //Pizza 클래스 타입의 배열의 각 요소마다 getSize() 호출.
        return 0;
    }
```