|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2021\_1\_C++\_04\_개념확인 | 학번 : | 20185309 | 이름 : | 황명원 |

* 기본 개념 확인

1. 빈 괄호를 채워 넣으시오.
2. ( & )연산자는 변수의 주소를 추출하기위해 사용한다
3. 동적으로 할당된 메모리를 반환하고자 할 때 ( delete ) 연산자를 사용한다
4. new 연산자는 할당되는 동적 메모리의 ( 객체 )를 반환한다
5. 동적으로 할당되는 메모리는 ( 힙 )영역에서 할당 받는 메모리이다
6. delete 사용 시 ‘포인터 변수’는 반드시 ( new )를 이용하여 할당 받은 메모리의 주소여야 한다
7. 객체 포인터로 멤버를 접근할 때 ( -> ) 연산자를 사용한다
8. ( nullptr ) 은 null pointer를 의미하는 것으로 NULL 매크로 사용시 함수 매개변수로 전달하는 경우 int타입으로 추론되는 문제점을 해결할 수 있다
9. new 연산자를 사용하여 객체를 동적으로 할당할 때 ( 생성자 )를 호출한다.
10. 동적으로 할당 된 객체 소멸 시 ( 소멸자 )를 호출한다
11. 다음 질문에 O, X로 답하시오
12. 동적 메모리 할당을 위해 new 함수를 사용한다 ( o )
13. 배열은 동적 할당 시 초기화를 할 수 없다( o )
14. 동적 메모리 반환 순서는 생성 순서와 동일해야 한다 (x )
15. C++ 동적 메모리 할당 시 크기와 형 변환이 반드시 있어야 한다( o )
16. 객체 배열 생성 시 기본 생성자를 호출한다. ( o )
17. 동적으로 배열을 생성하면서 초기화 할 때 배열 크기는 생략할 수 있다 ( x )
18. 객체 포인터 변수는 초기화 없이 사용할 수 있다 ( o )
19. delete 연산자는 정적으로 할당된 메모리를 해제할 때도 사용할 수 있다 (x )
20. 배열 형태로 동적 생성한 것은 배열 형태로 삭제해야 한다(o )
21. 클래스 멤버 변수에 대한 동적 생성은 생성자에서 할당하고 소멸자에서 동적 메모리를 해제해야 한다( o )
22. 제시된 클래스에 대하여 질문에 답하시오

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 선언하시오

Rec \*p=new Rec();

1. 선언된 포인터 변수 p에 객체 r의 주소를 지정하시오

p=&r;

1. 포인터 변수 p를 이용하여 write 함수를 호출할 수 있는 두 가지 방법을 제시하시오.

1)p->write();

2)(\*p).write();

1. 크기가 4인 Rec 객체 배열 arr를 동적으로 생성하는 문장을 제시하시오

Rec \*arr=new Rec[4];

1. 4번에서 할당 받은 동적메모리를 반환하는 문장을 제시하시오

delete []arr;

1. 4번에서 생성된 배열에서 배열 원소 두번째에 저장된 객체의 write() 멤버를 참조하는 문장을 두 가지 방법으로 제시하시오. 단, 배열 원소 참조 시 인덱스는 사용하지 말 것.

1)(arr+1)->write();

2)(\*(arr+1)).write();

1. 크기가 3인 Rec 객체 배열 dim을 동적으로 할당하면서 매개변수가 있는 생성자를 사용하여 초기화하는 문장을 제시하시오. 단, 초기화 값은 본인이 임의로 결정

Rec \*dim=new Rec[3];

dim[0]=Rec(1,2);

dim[1]=Rec(2,3);

dim[2]=Rec(3,4);

1. 다음과 같이 초기화 된 객체 배열을 사용하여 멤버 함수 wirte를 호출하는 문장을 제시하시오. 단, 범위 기반 for를 사용할 것

Rec array[] = { Rec(13,6), Rec(5,8), Rec(3,12) };

for(auto obj : array)

obj.write();

1. 다음 프로그램의 실행 결과를 제시하시오.

|  |
| --- |
| #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; |
| } |

텍스트이(가) 표시된 사진

자동 생성된 설명

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

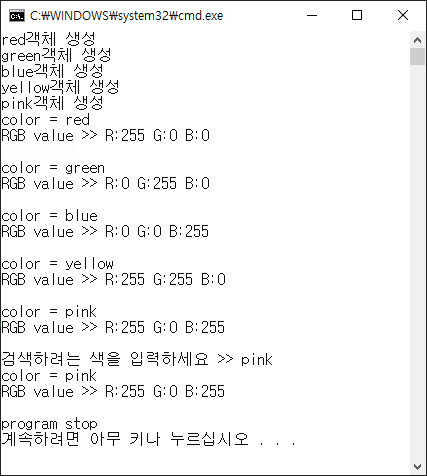
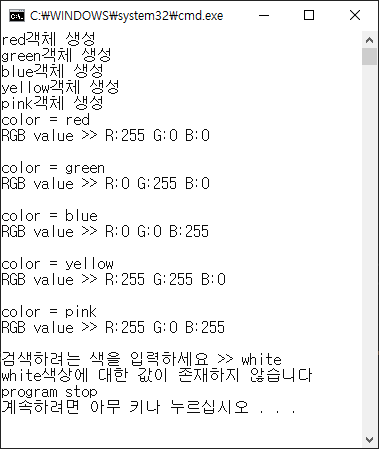
1. 다음 코드의 실행 결과를 제시하시오.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 제시된 Color 클래스를 구현한 후 제시된 조건대로 처리하는 프로그램을 작성하시오

* 크기가 5인 객체 배열을 생성하면서 초기화한다. (초기화 값은 본인이 임의로 할 것)
* 입력된 색상에 해당하는 RGB 코드값을 출력한다 – 색상을 반환하는 접근자 함수 추가할 것
* 전체 배열 원소를 출력한다

|  |
| --- |
| class Color {  int red, green, blue;  string color;  public:  Color():Color(0, 0, 0, "black") {}  Color(int r, int g, int b, string c) : red(r), green(g), blue(b), color(c){ }  void setColor(int r, int g, int b, string c); //색상 변경  void show() const; //멤버 변수 출력  //색상을 반환하는 접근자 함수 추가  };  void search(Circle \*p, string color, int size); //매개변수로 전달된 색상에 해당하는 RGB 값 출력, 전역 함수  ㅡ소스코드  #include <iostream> #include <string>  using namespace std;  class Color {  int red, green, blue;  string color; public:  Color();   Color(int r, int g, int b, string c);   ~Color();   void setColor(int r, int g, int b, string c);   string getColor();   void show() const; };  Color::Color() : Color(0, 0, 0, "black") {}  Color::Color(int r, int g, int b, string c) : red(r), green(g), blue(b), color(c) {  cout << color << "객체생성" << endl; }  void Color::setColor(int r, int g, int b, string c) {  red = r;  green = g;  blue = b;  color = c; }  string Color::getColor() {  return color; }  void Color::show() const {  cout << "color = " << color << endl;  cout << "RGB value >> R:" << red << " G:" << green << " B:" << blue << endl;  cout << endl; }  Color::~Color() {  }  void search(Color \*p, string color, int size) {  cout << "검색하려는 색을 입력하세요 >>" << color << endl;  for (int i = 0; i < size; i++) {   if (color == (\*(p + i)).getColor()) {  (p + i)->show();  break;  } else if (i == (size - 1))  cout << color << "색상에 대한 값이 존재하지 않습니다.";  }  }  int main() {  Color cl[5] = {Color(255, 0, 0, "red"),  Color(0, 255, 0, "green"),  Color(0, 0, 255, "blue"),  Color(255, 255, 0, "yellow"),  Color()};   for (int i = 0; i <= 4; i++) {  (cl + i)->show();  }  search(cl, "black", 5);  return 0;  }  ㅡ실행결과  텍스트이(가) 표시된 사진  자동 생성된 설명 |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 사용자로부터 배열 크기를 입력 받아 실수 배열을 동적 할당 받고 입력 받은 값으로 초기화한다. 배열 원소를 오름차순으로 정렬한 후 출력한다. 단, 정렬 전 배열 원소를 반드시 출력한다

ㅡ소스코드

#include <iostream>  
#include <string>  
using namespace std;  
  
  
  
int main() {  
 int num;  
 int min;  
 int temp;  
  
 cout<<"배열 크기를 입력하세요 >>";  
 cin>>num;  
  
 double \*p=new double[num]; //동적 배열 생성  
  
 for(int i=0 ; i<num ; i++){  
 cout<<i+1<<"번째 배열의 실수를 입력하세요 >>";  
 cin>>p[i]; //배열 입력  
 }  
 for(int i=0 ; i<num ; i++){ //배열 출력하는 for 문 생성  
 cout<<"p["<<i<<"] = "<<p[i]<<endl;  
 }  
  
 for(int i=0 ; i<num-1;i++){  
 min=p[i]; //min이라는 최소값을 표현하는 변수에 p[i]저장  
 for(int j=i+1 ; j<num ; j++){  
 if (min>p[j]) { //최소값으로 지정한 min보다 작은값있으면 바꿔주는 조건문 생성  
 min = p[j]; //조건이 true이면 바꿔준다  
 temp = p[i]; //temp를 이용하여 바꿔주기  
 p[i] = p[j];  
 p[j] = temp;  
 }  
 }  
  
 }  
 cout<<"오름차순 정렬"<<endl;  
  
 for(int i=0 ; i<num ; i++){ //출력 하면 오름차순이 나옵니다  
 cout<<"p["<<i<<"] = "<<p[i]<<endl;   
 }  
  
}

ㅡ실행결과

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 제시된 Circle 클래스를 구현하고 입력 받은 원의 개수만큼 객체 배열을 생성한다. 면적이 300~600사이의 원의 개수를 출력한다. 단, 반지름은 디폴트 생성자에서 10~40사이의 난수로 초기화 한다.

class Circle {

int radius;

public:

Circle();

~Circle();

void setRadius(int r) ;

double getArea() const;

int getRadius() const;

};

class CircleManage {

Circle \*p; //동적으로 할당된 배열의 시작 주소 저장

int size;

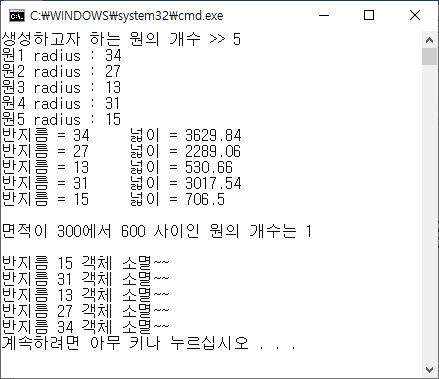
public:

CircleManage(); //입력 받은 크기만큼 Circle 객체 배열 생성

~CircleManage(); //동적으로 할당 받은 메모리 반환

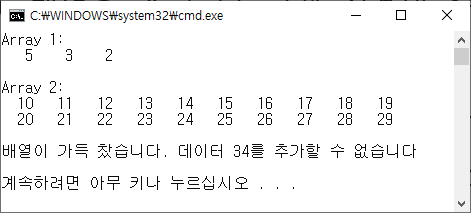
void CircleWrite(); //원의 반지름과 넓이 300~600사이의 원의 개수 출력

};



#include <iostream>  
#include <cstdlib>  
#include <ctime>  
  
using namespace std;  
class Circle {  
 int radius;  
public:  
 Circle();  
 ~Circle();  
 void setRadius(int r) ;  
 double getArea() const;  
 int getRadius() const;  
};  
Circle::Circle() {  
 radius=rand()%40+1;  
}  
  
class CircleManage {  
 Circle \*p;  
 int size;  
public:  
 CircleManage();  
 ~CircleManage();  
 void CircleWrite();  
};  
CircleManage::CircleManage() {  
 cout<<"생성하고자 하는 원의 개수 >>";  
 cin>>size;  
 p=new Circle[size];  
  
 int radius;  
 for(int i=0 ; i<size ;i++){  
 cout<<"원"<<i+1<<" radius :"<<p[i].getRadius()<<endl;  
  
  
 }  
}  
double Circle::getArea() const {  
 return 3.14\*radius\*radius;  
}  
Circle::~Circle() {  
 cout<<"반지름 "<<radius<<"객체 소멸~~~"<<endl;  
}  
void Circle::setRadius(int r) {  
 radius=r;  
}  
int Circle::getRadius() const {  
 return radius;  
}  
void CircleManage::CircleWrite() {  
 int count=0;  
 Circle \*pArray=p;  
 for(int i=0 ; i< size ; i++){  
 cout<<"반지름 = "<<pArray->getRadius()<<"\t넓이= "<<pArray->getArea()<<endl;  
 if((pArray->getArea()>=300) && (600>=pArray->getArea()))  
 count++;  
 pArray++;  
 }  
 cout<<endl<<"면적이 300에서 600 사이인 원의 개수는 "<<count<<endl<<endl;  
}  
CircleManage::~CircleManage() {  
 delete []p;  
}  
  
int main(){  
 CircleManage cm;  
 cm.CircleWrite();  
 return 0;  
  
}

1. 배열을 조금 더 쉽게 사용할 수 있게 Array 클래스를 작성하시오. Array 클래스는 데이터 멤버로 capacity, size, arr(힙에 생성한 배열의 첫 번째 요소를 가리키는 포인터)를 갖고, 배열 마지막 위치에 요소를 추가하는 멤버 함수 insert와 배열의 요소를 출력하는 멤버 함수 print를 갖도록 설계하시오. 또한 이렇게 만들어진 클래스를 테스트하는 애플리케이션도 작성한다. capacity를 넘어서 요소를 추 가할 경우에는 ‘배열이 꽉 차서 요소를 추가할 수 없다 ’라는 메시지를 출력한다



int main(){

//크기가 10인 배열 생성

Array array1(10);

array1.insert(5);

array1.insert(3);

array1.insert(2);

cout << "Array 1: " << endl;

array1.print();

cout << endl;

//크기가 20인 배열 생성

Array array2(20);

for (int i = 0; i < 20; i++){

array2.insert(i+10);

}

cout << "Array 2: " << endl;

array2.print();

array2.insert(34);

cout << endl;

return 0;

}

ㅡ소스코드

#include <iostream>  
using namespace std;  
  
class Array{  
  
 int \*arr;  
 int size;  
 int capacity;  
   
public:  
 Array();  
 Array(int s);  
 ~Array();  
 void print();  
  
 void insert(int n);  
  
};  
Array::Array() {  
 arr=new int[0];  
  
}  
Array::Array(int s) {  
 capacity=s;  
 arr=new int[capacity];  
 size=0;  
}  
Array::~Array() {  
 delete[] arr;  
}  
void Array::insert(int n) {  
 if(size<capacity) {  
 arr[size] = n;  
 size++;  
 }  
 else  
 cout<<"배열이 가득 찼습니다. 데이터"<<n<<"를 추가할 수 없습니다.";  
}  
  
void Array::print(){  
 for(int j=0 ; j<size; j++){  
 if((j%10==0) and j>0)  
 cout<<endl;  
 cout<<arr[j]<<" ";  
  
 }  
 cout<<endl;  
}  
  
  
int main(){  
 Array array1(10);  
 array1.insert(5);  
 array1.insert(3);  
 array1.insert(2);  
 cout<<"Array 1: "<<endl;  
 array1.print();  
 cout<<endl;  
  
 Array array2(20);  
 for (int i = 0; i < 20; i++){  
 array2.insert(i+10);  
 }  
 cout << "Array 2: " << endl;  
 array2.print();  
 array2.insert(34);  
 cout << endl;  
  
  
}

ㅡ실행결과텍스트이(가) 표시된 사진

자동 생성된 설명