|  |  |  |
| --- | --- | --- |
| 2021\_1\_C++ \_10 | 학번 :20185309 | 이름 :황명원 |
|  |  |  |

* **프로그램 과제**

1. Book 객체에 대하여 다음과 같은 연산을 수행할 수 있도록 연산자 함수를 멤버 함수로 구현하고 프로그램을 완성 하시오

class Book {

string title;

int price, pages;

public:

Book(string title = "", int price = 0, int pages = 0);

void show();

string getTitle() //title 반환

};

int main() {

Book a("청춘", 20000, 300), b("미래", 30000, 500);

a += 500; // 책 a의 가격 500원 증가

b -= 500; // 책 b의 가격 500원 감소

a.show();

b.show();

Book aa("명품 C++", 30000, 500), bb("고품 C++", 30000, 500);

if (aa == 30000) cout << "정가 30000원" << endl; // price 비교

if (aa == "명품 C++") cout << "명품 C++ 입니다." << endl; // 책 title 비교

if (aa == bb) cout << "두 책이 같은 책입니다." << endl; // title, price, pages 모두 비교

}

|  |
| --- |
| **[프로그램 소스]**  #include <iostream> #include <string> using namespace std;  class Book {  string title;  int price, pages; public:  Book(string title = "", int price = 0, int pages = 0){  this->title=title;  this->price=price;  this->pages=pages;  }  void show();  Book operator+=(int b2);  Book operator-=(int b2);  bool operator==(int b2);  bool operator==(string b2);  bool operator==(Book b2);  string getTitle(){  return title;  } //title 반환 }; Book Book::operator+=(int b2) {  Book tmp;  tmp.price=this->price+b2;  return tmp; } Book Book::operator-=(int b2) {  Book tmp;  tmp.price=this->price-b2;  return tmp; } bool Book::operator==(int b2) {  if(price==b2)  return true;  else  return false; } bool Book::operator==(string b2) {  if(title==b2)  return true;  else  return false; } bool Book::operator==(Book b2) {  if(this->title==b2.title and this->price==b2.price and this->pages==b2.pages)  return true;  else  return false; }  void Book::show() {  cout<<"책 title 은 "<<title<<endl;  cout<<"책 price 는 "<<price<<endl;  cout<<"책 pages 는 "<<pages<<endl; } int main() {  Book a("청춘", 20000, 300), b("미래", 30000, 500);  a += 500; // 책 a의 가격 500원 증가  b -= 500; // 책 b의 가격 500원 감소  a.show();  b.show();   Book aa("명품 C++", 30000, 500), bb("고품 C++", 30000, 500);  if (aa == 30000) cout << "정가 30000원" << endl; // price 비교  if (aa == "명품 C++") cout << "명품 C++ 입니다." << endl; // 책 title 비교  if (aa == bb) cout << "두 책이 같은 책입니다." << endl; // title, price, pages 모두 비교 } |
| **[실행 결과]**  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. Matrix 클래스에 대하여 다음과 같은 연산이 가능하도록 연산자 함수를 프렌드 함수로 구현하고 프로그램을 완성하시오.

class Matrix {

int ar[4];

public:

Matrix() { for (int i = 0; i < 4; i++) ar[i] = 0; }

Matrix(int a1, int a2, int b1, int b2) {

ar[0] = a1; ar[1] = a2; ar[2] = b1; ar[3] = b2;

}

void show() {

cout << "Matrix = { ";

for (int i = 0; i < 4; i++) cout << ar[i] << ' ';

cout << "}" << endl;

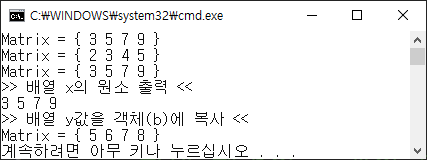
}

};

int main() {

Matrix a(1, 2, 3, 4), b(2, 3, 4, 5), c;

**c = a + b;**

 **a += b;**

a.show(); b.show(); c.show();

int x[4], y[4] = { 5,6,7,8 }; // 2차원 행렬의 4 개의 원소 값

**a >> x; // a의 각 원소를 배열 x에 복사**

**b << y; // 배열 y의 원소 값을 b의 각 원소에 설정**

for (int i = 0; i < 4; i++) cout << x[i] << ' '; // x[] 출력

cout << endl;

b.show();

}

|  |
| --- |
| **[프로그램 소스]**  #include <iostream> using namespace std;   class Matrix {  int ar[4]; public:  Matrix() { for (int i = 0; i < 4; i++) ar[i] = 0; }  Matrix(int a1, int a2, int b1, int b2) {  ar[0] = a1; ar[1] = a2; ar[2] = b1; ar[3] = b2;  }  friend Matrix operator+(Matrix m1,Matrix m2);  friend void operator+=(Matrix& m1,Matrix m2);  friend void operator >>(Matrix m1,int x[]);  friend void operator << (Matrix& m1 , int x[]);  void show() {  cout << "Matrix = { ";  for (int i = 0; i < 4; i++) cout << ar[i] << ' ';  cout << "}" << endl;  } };  Matrix operator+(Matrix m1,Matrix m2){  Matrix tmp;  for(int i=0 ; i<4 ; i++)  tmp.ar[i]=m1.ar[i]+m2.ar[i];  return tmp; } void operator+=(Matrix& m1,Matrix m2){  for(int i=0 ; i<4 ; i++)  m1.ar[i]=m1.ar[i]+m2.ar[i];  } void operator>>(Matrix m1,int x[]){  for(int i=0 ; i<4 ; i++)  x[i]=m1.ar[i]; } void operator<<(Matrix& m1,int x[]){  for(int i=0 ; i<4 ; i++)  m1.ar[i]=x[i]; } int main() {  Matrix a(1, 2, 3, 4), b(2, 3, 4, 5), c;  c = a + b;  a += b;  a.show(); b.show(); c.show();   int x[4], y[4] = { 5,6,7,8 }; // 2차원 행렬의 4 개의 원소 값  a >> x; // a의 각 원소를 배열 x에 복사  b << y; // 배열 y의 원소 값을 b의 각 원소에 설정   cout<<">> 배열 x의 원소 출력 <<"<<endl;  for (int i = 0; i < 4; i++) cout << x[i] << ' '; // x[] 출력  cout<<endl;  cout<<">> 배열 y값을 객체(b)에 복사 <<"<<endl;  b.show(); } |
| **[실행 결과]**  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 원을 추상화 한 Circle 클래스에 대하여 다음 연산이 가능하도록 프렌드 함수로 작성하고 프로그램을 완성 하시오.

class Circle {

int radius;

public:

Circle(int radius = 0);

void show();

};

int main() {

Circle a(5), b(4);

cout << "Circle 객체 : a" << endl;

a.show();

cout << "Circle 객체 : b" << endl;

b.show();

++a; // 반지름을 1 증가 시킨다.

b = a++; // 반지름을 1 증가 시킨다.

cout << "Circle 객체 : ++a" << endl;

a.show();

cout << "Circle 객체 : b=a++" << endl;

b.show();

b = 1 + a; // b의 반지름을 a의 반지름에 1을 더한 것으로 변경

cout << "Circle 객체 : b= 1+ a" << endl;

b.show();

}

|  |
| --- |
| **[프로그램 소스]**  #include <iostream> using namespace std;  class Circle {  int radius; public:  Circle(int radius = 0){  this->radius=radius;  };  friend void operator++(Circle& c1);  friend Circle operator++(Circle& c1,int x);  friend Circle operator+(int x,Circle c1);  void show(); }; void operator++(Circle& c1){  c1.radius++; } Circle operator++(Circle& c1,int x){  Circle tmp=c1;  c1.radius++;  return tmp; } Circle operator+(int x,Circle c1){  Circle tmp;  tmp.radius=x+c1.radius;  return tmp; } void Circle::show() {  cout<<"radius : "<<radius<<endl; } int main() {  Circle a(5), b(4);  cout << "Circle 객체 : a" << endl;  a.show();  cout << "Circle 객체 : b" << endl;  b.show();  ++a; // 반지름을 1 증가 시킨다.  b = a++; // 반지름을 1 증가 시킨다.  cout << "Circle 객체 : ++a" << endl;  a.show();  cout << "Circle 객체 : b=a++" << endl;  b.show();   b = 1 + a; // b의 반지름을 a의 반지름에 1을 더한 것으로 변경  cout << "Circle 객체 : b= 1+ a" << endl;  b.show(); } |
| **[실행 결과]**  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. []연산자를 중복정의하여 제시된 결과처럼 실행하는 프로그램을 작성하시오.

class Array

{

private:

double\* ptr;

int size;

public:

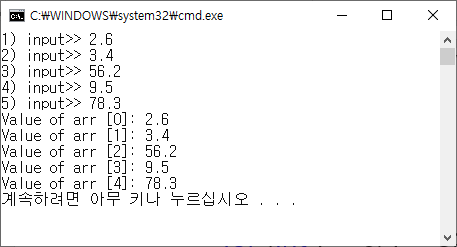
Array(int size); //size크기를 갖는 배열 동적 생성

~Array();

//[]연산자 중복

};

int main(){

 Array arr(5);

for (int i = 0; i < 5; i++) {

cout << i+1 <<") input>> ";

cin >> arr[i];

}

return 0;

}

|  |
| --- |
| **[프로그램 소스]**  #include <iostream> using namespace std;  class Array { private:  double\* ptr;  int size; public:  Array(int size){  this->size=size;  ptr=new double[size];  } //size크기를 갖는 배열 동적 생성  ~Array(){}  double& operator[](int index){  return ptr[index];  }  //[]연산자 중복 };   int main(){  Array arr(5);   for (int i = 0; i < 5; i++) {  cout << i+1 <<") input>> ";  cin >> arr[i];  }  for (int i = 0; i < 5; i++) {  cout <<"Value of arr["<<i<<"]: "<<arr[i]<<endl;  }   return 0; } |
| **[실행 결과]**  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 정수 배열을 항상 증가 순으로 유지하는 StoredArray 클래스를 작성하려고 한다. 아래의 main()함수가 작동할 수 있도록 SortedArray 클래스를 작성하고 +와 =연산자도 작성하시오.

class SortedArray {

int size; // 현재 배열의 크기

int \*p; // 정수 배열에 대한 포인터

void sort(); // 정수 배열을 오름차순으로 정렬

public:

SortedArray(); // p는 NULL로 size는 0으로 초기화

SortedArray(SortedArray& src); // 복사 생성자

SortedArray(int p[], int size); // 생성자. 정수 배열과 크기를 전달받음

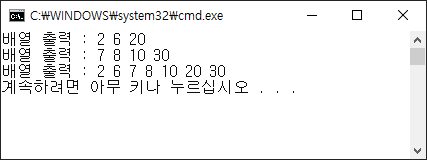
~SortedArray(); // 소멸자

SortedArray operator + (SortedArray& op2);

SortedArray& operator = (const SortedArray& op2); // 현재 배열에 op2 배열을 복사

void show(); // 배열의 원소 출력

};

int main() {

int n[] = { 2, 20, 6 };

int m[] = { 10, 7, 8, 30 };

SortedArray a(n, 3), b(m, 4), c;

c = a + b; // +, = 연산자 작성 필요

// + 연산자가 SortedArray 객체를 리턴하므로 복사 생성자 필요

a.show();

b.show();

c.show();

}

|  |
| --- |
| **[프로그램 소스]**  #include <iostream> using namespace std;  class SortedArray {  int size; // 현재 배열의 크기  int \*p; // 정수 배열에 대한 포인터  void sort(); // 정수 배열을 오름차순으로 정렬 public:  SortedArray(){  size=0;  p=NULL;  } // p는 NULL로 size는 0으로 초기화  SortedArray(SortedArray& src){  this->size=src.size;  this->p=new int[this->size];  for(int i=0 ; i<this->size ; i++){  this->p[i]=src.p[i];  }  } // 복사 생성자  SortedArray(int p[], int size){  this->p=new int[size];  this->size=size;  for(int i=0 ; i<this->size ; i++){  this->p[i]=p[i];  }  } // 생성자. 정수 배열과 크기를 전달받음  ~SortedArray(){  delete []p;  } // 소멸자  SortedArray operator + (SortedArray& op2);  SortedArray& operator = (const SortedArray& op2); // 현재 배열에 op2 배열을 복사  void show(); // 배열의 원소 출력 }; void SortedArray::sort() {  int swap;  for(int i=0 ; i<size ; i++) {  for (int j = i; j < size; j++) {  if (p[i] > p[j]) {  swap = p[j];  p[j] = p[i];  p[i] = swap;  }  }  } } SortedArray SortedArray::operator+(SortedArray &op2) {  SortedArray tmp;  tmp.size=this->size+op2.size;  tmp.p=new int[tmp.size];  for(int i=0 ; i<this->size; i++){  tmp.p[i]=this->p[i];  }  for(int i=this->size ; i<tmp.size; i++){  tmp.p[i]=op2.p[i-(this->size)];  }  return tmp; } SortedArray& SortedArray:: operator = (const SortedArray& op2){  this->size=op2.size;  this->p=new int[this->size];  for(int i=0 ;i<this->size ; i++){  this->p[i]=op2.p[i];  }  return \*this; } void SortedArray::show() {  sort();  cout<<"배열 출력 : ";  for(int i=0 ;i<size ; i++){  cout<<p[i]<<" ";  }  cout<<endl; }  int main() {  int n[] = { 2, 20, 6 };  int m[] = { 10, 7, 8, 30 };  SortedArray a(n, 3), b(m, 4), c;   c = a + b; // +, = 연산자 작성 필요  // + 연산자가 SortedArray 객체를 리턴하므로 복사 생성자 필요  a.show();  b.show();  c.show(); } |
| **[실행 결과]**  텍스트이(가) 표시된 사진  자동 생성된 설명 |