|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2021\_1\_C++\_09\_개념확인 | 학번 : | 20185309 | 이름 : | 황명원 |
|  |  |  |  |  |

* 기본 개념 확인

1. 다음 질문에 O, X로 답하시오
   1. 함수 중복은 매개변수 타입이 다르거나 개수가 달라야 한다( o )
   2. 함수 중복은 리턴 타입만 달라도 가능하다 ( x )
   3. 소멸자는 매개변수를 가지지 않으므로 중복할 수 없다 ( o )
   4. 디폴트 매개변수 선언은 위치와 상관없다 ( x )
   5. 참조 매개변수로 인하여 모호성이 발생할 수 있기때문에 함수 중복 시 주의해야 한다( o )
   6. static 멤버 함수는 non-static 멤버는 사용할 수 없다( o)
   7. statc 멤버 함수에서 this는 선택적으로 사용할 수 있다 ( x )
   8. static 멤버는 클래스이름으로만 접근할 수 있다 (x )
   9. 프렌드 함수는 멤버 함수 이므로 상속이 가능하다 ( x )
   10. 모든 연산자를 중복할 수 있다 ( x )
   11. 연산자 중복 시 디폴트 매개변수는 사용할 수 없다( o )
   12. 피 연산자의 개수를 변경하여 연산자를 중복할 수 있다 (x )
   13. 전역함수만 프렌드 함수로 사용할 수 있다 ( x )
   14. 디폴트 매개변수를 사용하여 함수 중복을 간소화할 수 있다 ( o )
   15. 중복 함수와 디폴트 매개변수를 가진 함수를 함께 사용할 수 있다(x )
2. 다음과 같은 디폴트 매개변수를 갖는 함수를 호출할 때 컴파일 오류가 발생하는 문장을 모두 고르시오

void msg(char id, string text="Hello“ );

1. msg('a');
2. msg();
3. msg('b', "default");
4. msg("C++");

답:2번,4번

1. 다음과 같이 중복된 함수를 호출하는 문장을 제시하시오
2. int sum(int a, int b, int c) {

return a + b + c;

}

cout<<sum(1,2,3);

1. double sum(double a, double b) {

return a + b;

}

cout<<sum(3.0,4.0);

1. int sum(int a, int b) {

return a + b;

}

cout<<sum(1,2);

1. F\_exam 클래스에서 다음과 같은 외부 함수 isCheck()와 클래스 F\_exam\_m 멤버를 프렌드로 초대하는 선언문을 작성하세요

class F\_exam;

bool isCheck(int i, F\_exam fe);

class F\_exam\_m{

void fmethod(F\_exam f);

}

class F\_exam{

//1번) 외부함수를 프렌드로 선언

friend bool isCheck(int i, F\_exam fe);

//2번) F\_exam\_m 클래스의 멤버 함수 fmethod를 프렌드로 선언

friend void F\_exam\_m::fmethod(F\_exam f);

//3번) F\_exam\_m 클래스 전체를 프렌드로 선언

friend F\_exam\_m;

}

1. Quiz\_9 클래스의 static 멤버 함수를 호출하는 방법을 모두 제시하시오.

class Quiz\_9 {

public:

static int cnt;

static void print();

};

Int main(){

Quiz\_9 q;

Quiz\_9 \*p;

p=&lee;

(첫번째 방법) q.print();

(두번째 방법) p->print();

(세번째 방법) Quiz\_9::print();

}

1. 위 5번 문제의 static 멤버 변수에 대한 외부 선언 문장을 제시하시오

Int Quiz\_9::cnt=0;

1. 다음과 같은 연산이 +연산이 가능하도록 연산자 함수를 외부 함수와 멤버 함수로 각각 제시하시오.

class Power {

int kick, punch;

public:

Power(int kick=0, int punch=0){ this->kick = kick; this->punch = punch;}

}

int main(){

Power a(3,5), b(4,6), c

c=a\*b;

};

//1번) 외부 함수로 구현

Power operator \*(Power p1,Power p2);

class Power {

int kick, punch;

public:

Power(int kick=0, int punch=0){ this->kick = kick; this->punch = punch;}

friend Power operator \*(Power p1,Power p2);

};

Power operator\*(Power p1,Power p2){

Power tmp;

tmp.kick=p1.kick\*p2.kick;

tmp.punch=p1.punch\*p2.punch;

return tmp;

}

int main(){

Power a(3,5), b(4,6), c

c=a\*b;

}

//2번) 멤버 함수로 구현

class Power {

int kick, punch;

public:

Power(int kick=0, int punch=0){ this->kick = kick; this->punch = punch;}

Power operator \*(const Power& p1);

};

Power Power::operator\*(const Power& p1){

Power tmp;

tmp.kick=this->kick\*p1.kick;

tmp.punch=this->punch\*p1.punch;

return tmp;

}

int main(){

Power a(3,5), b(4,6), c

c=a\*b;

}

1. 디폴트 매개변수에 대하여 설명하시오

->매개 변수에 값이 넘어오지 않는 경우, 디폴트 값을 받도록 선언된 매개 변수

(‘매개 변수 = 디폴트값’ 형태로 선언)

1. 프렌드 함수에 대하여 설명하시오

->클래스의 멤버 함수가 아닌 외부 함수 - 상속불가

1)전역 함수  
 -다른 클래스의 멤버 함수  
 -프렌드 함수 개수에 제한 없음

2)friend 키워드로 클래스 내에 선언된 함수  
 -클래스의 모든 멤버를 접근할 수 있는 권한 부여

3)프렌드 선언의 필요성

-클래스의 멤버로 선언하기에는 무리가 있고, 클래스의 모든 멤버를 자유롭게 접근해야 할 경우

1. 함수 중복에 대하여 설명하시오.

->동일한 이름의 함수가 공존

-> function overloading  
->함수 중복이 가능한 범위

1)보통 함수들 사이  
 2)클래스의 멤버 함수들 사이  
 3)상속 관계에 있는 기본 클래스와 파생 클래스의 멤버 함수들 사이

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) {

this->title = title;

this->price = price;

this->pages = pages;

}

void show() { cout << title << ' ' << price << "원 " << pages << " 페이지" << endl; }

string getTitle() { return title; }

};

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();

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  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;  }  Book& operator+=(int price);  Book& operator-=(int price);   void show() { cout << title << ' ' << price << "원 " << pages << " 페이지" << endl; }   string getTitle() { return title; } }; Book& Book::operator+=(int price) {  this->price+=price;  return \*this; } Book& Book::operator-=(int price) {  this->price-=price;  return \*this; }   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(); } |
| [실행결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. Point 객체에 대하여 다음과 같은 연산이 가능하도록 연산자함수를 구현 하시오. 단 외부 함수로 구현하시오

class Point {

int x, y;

public:

Point(int xp = 0, int yp = 0) :x(xp), y(yp) {}

void show(string obj) {

cout << obj << "=> x: " << x << ", y: " << y << endl;

}

};

int main() {

Point p1(3, 4);

Point p2(5, 6);

Point p3;

p3 = p1 + p2;

p1.show("p1");

p2.show("p2");

p3.show("P1+p2");

p3 = p1 \* p2;

p3.show("P1\*p2");

}

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std;  class Point {  int x, y;  public:  Point(int xp = 0, int yp = 0) :x(xp), y(yp) {}  void show(string obj) {  cout << obj << "=> x: " << x << ", y: " << y << endl;  }  friend Point operator+(Point p1,Point p2);  friend Point operator\*(Point p1,Point p2); }; Point operator+(Point p1,Point p2){  Point tmp;  tmp.x=p1.x+p2.x;  tmp.y=p1.y+p2.y;  return tmp; } Point operator\*(Point p1,Point p2){  Point tmp;  tmp.x=p1.x\*p2.x;  tmp.y=p1.y\*p2.y;  return tmp; }  int main() {  Point p1(3, 4);  Point p2(5, 6);  Point p3;  p3 = p1 + p2;   p1.show("p1");  p2.show("p2");  p3.show("P1+p2");  p3 = p1 \* p2;  p3.show("P1\*p2"); } |
| [실행결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |