|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cpp\_09\_참조와복사생성자 | 학번 : | 20175260 | 이름 : | 최영빈 |

* **개념 확인 – 참조와복사생성자**

1. 다음 질문에 O, X로 답하세요.
2. 참조자는 선언과 동시에 초기화 되어야 한다. ( o )
3. 참조변수에도 새로운 공간이 별도로 할당된다. ( x )
4. 배열에서 범위기반 for를 사용하여 배열 원소 값 변경 방지와 복사를 피하고 싶으면 const 참조 변수를 사용한다. ( o )
5. 함수 실행 후 참조를 반환하는 경우, 변수 등과 같이 현존하는 공간에 대한 참조를 리턴 하는 것이다. ( o )
6. 얕은 복사(shallow copy)는 객체 복사 시 객체의 멤버를 1:1대로 복사한다. ( o )
7. 생성자와 소멸자의 비대칭 문제는 주소에 의한 호출만 해결할 수 있다. ( o )
8. 객체의 복사 생성 시 호출되는 특별한 생성자를 복사 생성자(copy constructor)라 한다. ( o )
9. 복사 생성자는 클래스의 생성자처럼 한 클래스에 여러 개 선언이 가능하다. ( x )
10. 자기 클래스에 대한 참조 매개변수를 가지는 생성자를 복사 생성자라 한다. ( o )
11. 복사 생성자가 선언되어 있지 않는 클래스에 복사 생성자 호출이 필요한 경우가 발생할 경우, 컴파일러는 오류를 발생시킨다. ( x )
12. 깊은 복사(deep copy)는 객체 복사 시 객체의 멤버변수에 동적메모리가 할당된 경우 원본이 가진 메모리의 크기만큼 사본에도 메모리를 별도로 동적 할당해야 한다. ( o )
13. 깊은복사(deep copy)는 사본과 원본이 같은 공간의 메모리를 공유하게 된다. ( x )
14. 변수처럼 이름과 주소를 가진 대상(지속되는 객체)을 Rvalue라고 한다. ( x )
15. Rvalue reference를 사용할 때에는 &&를 사용한다. ( o )
16. 객체나 값을 전달할 때 복사를 사용하지 않고 소유권을 완전히 이동 시키는 것은 불가능하다. ( x )
17. lvalue를 rvalue로 만들 때에는 move()를 사용한다. ( o )
18. 이동 생성자와 이동 대입 연산자를 구현 할 때에는 Rvalue Reference를 파라미터로 받는 함수를 작성한다. ( x )
19. 호출하는 코드에서 넘어온 값이 매개 변수에 복사되는 인자 전달 방식을 이르는 말은 무엇인가요?

==풀이 ==

call by value

1. 다음 코드에서 잘못된 부분을 설명하세요.

|  |
| --- |
| int x = 1000;  int& y = 2000; |

==풀이 ==

int& y = 2000;에서 &는 참조를 뜻한다. 따라서 변수가 와야한다.

1. 다음 코드에서 잘못된 부분을 설명하세요.

|  |
| --- |
| const int x = 100;  double& y = x; |

==풀이 ==

const 참조자는 참조만 가능하고 값 변경 금지이다.

1. 다음 코드의 실행 결과를 제시하세요.

|  |
| --- |
| int x = 10;  int& y = x;  cout << x << " " << y; |

==풀이 ==

10 10

1. 다음 코드의 실행 결과를 제시하세요.

|  |
| --- |
| int x = 100;  int& y = x;  int& z = x;  cout << x << " " << y << " " << z; |

==풀이 ==

100 100 100

1. 다음은 참조에 의한 호출을 사용하여 swap() 함수를 호출하는 문장입니다. swap() 함수 호출 후에는 두 변수의 내용이 서로 바뀝니다. swap() 함수의 원형만 제시하세요.

|  |
| --- |
| int a=60, b=90;  swap(a,b); |

==풀이 ==

void swap(int &a, int &b)

1. 주석에 해당하는 문장을 제시하세요.

|  |
| --- |
| int array[]={3,5,3,2,1,7};  int& f(int a){  return array[a];  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; //함수 f를 실행하여 배열 array의 네번째 원소를 60으로 변경 |

==풀이 ==

f(3) = 60;

1. 복사 생성자가 자동으로 호출되는 경우는 어떤 경우인지 설명하고 그 예를 보이세요.

==풀이 ==

리턴값이 객체인 경우

AA g(){  
 AA aa(1,2);

return aa;

}

1. Sample 객체 a에 대한 참조 변수 ref를 선언하는 문장을 제시하세요.

==풀이 ==

Sample &ref = a;

1. Sample 클래스의 복사 생성자를 선언하는 함수 원형을 제시하세요. 단, 매개변수는 s로 합니다.

==풀이 ==

Sample a(Sample s)

1. Person 클래스의 선언이 다음과 같은 경우 Person 클래스의 복사 생성자를 선언하는 문장을 제시하세요.

|  |
| --- |
| class Person { //Person 클래스 선언  char \*name;  int id;  public:  Person(const Person& p); // 복사 생성자  }; |

==풀이 ==

Person::Person(const Person& p){

}

1. Person 클래스의 선언이 다음과 같은 경우 Person 클래스의 이동 생성자를 선언하는 문장을 제시하세요.

|  |
| --- |
| class Person { //Person 클래스 선언  char \*name;  int id;  public:  Person(Person&& p); // 이동 생성자  }; |

==풀이 ==

Person::Person(Person&& p){

}

* **개념 적용 응용 프로그래밍 – 참조**

1. 아래에 제시된 함수 원형과 main() 함수를 참고하여 실행 결과를 예측하시오.

include <iostream>

using namespace std;

char &find2replace(char fstr[], char re, bool &success) {

int i = 0;

while (fstr[i]) {

if (fstr[i] == re) {

success = true; //발견함. 함수 성공

return fstr[i];

}

i++;

}

return fstr[0]; //발견하지 못함, return 값 의미 없음

}

int main() {

char str[] = "C++ programming";

char has = '+';

char replace = 'p';

bool result = false;

cout << "변경 전 문자열 = " << str << endl;

find2replace(str, has, result) = replace; //'+' 위치에 'p' 저장

if (result == true)

cout << "변경 후 문자열 = " << str << endl;

else

cout << str << "에서 " << has << "를 발견하지 못함." << endl;

return 0;

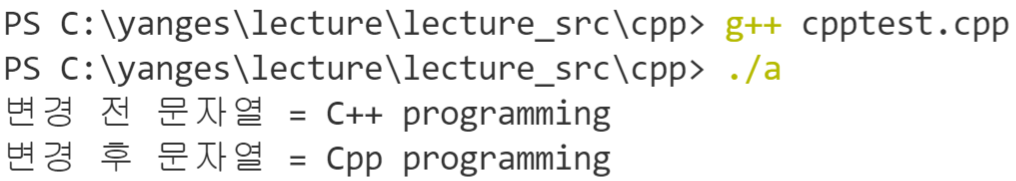
}

==풀이 ==

C++ programming

Cp+ programming

1. 아래에 제시된 main() 함수와 실행 결과를 참고하여 위 1번 문제의 부족한 부분을 해결 할 수 있는 find2replace() 함수를 작성하세요. 단, find2replace() 함수의 모든 매개변수는 참조를 사용한다.



include <iostream>

using namespace std;

//find2replace() 함수 구현-본인작성

int main() {

std::string str = "C++ programming";

std::string has = "+";

string replace = "p";

bool result = false;

cout << "변경 전 문자열 = " << str << endl;

find2replace(str, has, replace, result);

if (result == true)

cout << "변경 후 문자열 = " << str << endl;

else

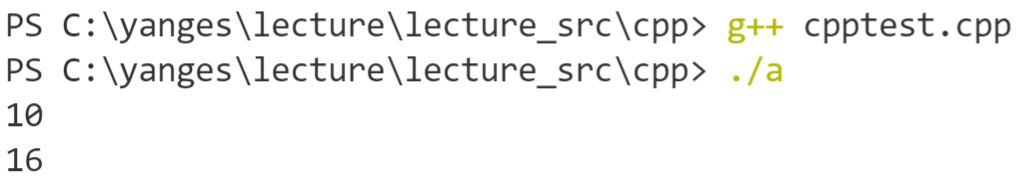
cout << str << "에서 " << has << "를 발견하지 못함." << endl;

return 0;

}

|  |
| --- |
| [프로그램 소스]  #include<iostream>  using namespace std;  char& find2replace(string &a, string &b,string&c, bool &d){  int i =0;  while(a[i]){  if(a[i]==b[0]){  d = true;  a[i] = c[0];  }  i++;    }  if(d == true)  return a[i];  else  return a[0];  }  int main()  {  std::string str = "C++ programming";  std::string has = "+";  string replace = "p";  bool result = false;  cout << "변경 전 문자열 = " << str << endl;  find2replace(str, has, replace, result);  if (result == true)  cout << "변경 후 문자열 = " << str << endl;  else  cout << str << "에서 " << has << "를 발견하지 못함." << endl;  return 0;  } |
| [실행 결과]  이미지 |

1. Accumulator 클래스는 생성자 매개변수로 전달된 값을 누적 저장하는 기능을 한다. 아래에 제시된 클래스 선언과 main() 함수 그리고 실행 결과를 참고하여 add() 함수를 작성하시오.



#include <iostream>

using namespace std;

class Accumulator {

int value;

public:

Accumulator(int val) : value{val} { };

Accumulator &add(int n); //\*this

int get() { return value; }

};

//add() 함수 구현 – 본인 작성

int main() {

Accumulator acc(10);

cout << acc.get() << endl; //10 출력

acc.add(1).add(2).add(3); //acc 객체의 value는 15가 됨.

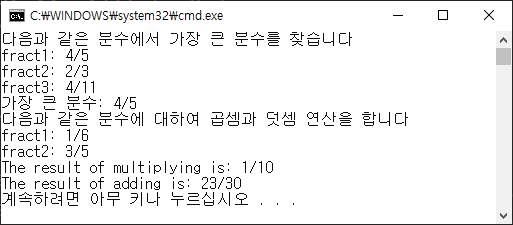
cout << acc.get() << endl; //15 출력

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  using namespace std;  class Accumulator  {  int value;  public:  Accumulator(int val) : value{val} {}; //누적저장  Accumulator &add(int n); //\*this  int get() { return value; }  };  Accumulator &Accumulator::add(int n){  value += n;  return \*this;  }  int main()  {  Accumulator acc(10);  cout << acc.get() << endl; // 10 출력  acc.add(1).add(2).add(3); // acc 객체의 value는 15가 됨.  cout << acc.get() << endl; // 16 출력  } |
| [실행 결과]  이미지 |

1. 참조로 전달과 참조로 리턴을 활용해서 2개의 분수를 곱하고 리턴 하는 함수와 3개의 분수 중에서 가장 큰 분수를 찾는 함수를 작성하고 테스트하는 프로그램을 작성하시오. 단, 제시된 분수를 사용하시오. (곱셈, 덧셈 함수 만들기)

//가장 큰 값을 찾고자 하는 분수

Fraction fract1(12, 15);

Fraction fract2(6, 9);

Fraction fract3(4, 11);

//곱셈과 덧셈 연산을 수행하는 분수

Fraction fract1(2, 12);

Fraction fract2(15, 25);

Fraction product; //연산 결과를 저장하는 분수

class Fraction{

int numer; //분자

int denom; //분모

public:

Fraction(int num, int den); //매개변수가 있는 생성자

Fraction(); //디폴트 생성자

~Fraction();

int getNumer() const; //접근자

int getDenom() const;

void setNumer(int num); //설정자

void setDenom(int den);

void print() const; //분수 출력

private:

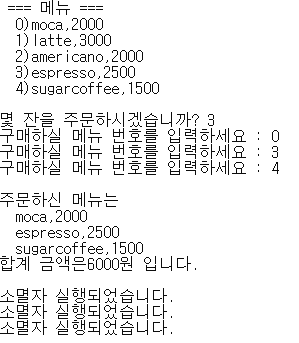
void normalize(); //gcd() 함수를 사용하여 기약분수로 처리

int gcd(int n, int m); //분모와 분자의 최대 공약수

};

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  class Fraction  {  int numer; //분자  int denom; //분모  public:  Fraction(int num, int den); //매개변수가 있는 생성자  Fraction(); //디폴트 생성자  ~Fraction(){;}  int getNumer() const{return numer;}  int getDenom() const{return denom;}  void setNumer(int num){numer = num;}  void setDenom(int den){denom = den;}  void print() const; //분수 출력  void add(Fraction& fr1, Fraction& fr2);  void mul(Fraction &fr1, Fraction &fr2);  private:  void normalize(); // gcd() 함수를 사용하여 기약분수로 처리  int gcd(int n, int m); //분모와 분자의 최대 공약수  };  Fraction &findLargest(Fraction &fr1, Fraction &fr2, Fraction &fr3);  Fraction &findLarger(Fraction &fract1, Fraction &fract2);  void Fraction::mul(Fraction &fr1, Fraction &fr2){  numer = fr1.getNumer() \* fr2.getNumer();  denom = fr1.getDenom() \* fr2.getDenom();  normalize();  cout << numer<<"/"<<denom<<endl;  }  void Fraction::add(Fraction &fr1, Fraction &fr2){  numer = fr1.getDenom() \* fr2.getNumer() +fr2.getDenom() \* fr1.getNumer();  denom = fr1.getDenom() \* fr2.getDenom();  normalize();  cout << numer << "/" << denom << endl;  }  Fraction::Fraction(int num, int den){  numer = num;  denom = den;  normalize();  }  Fraction::Fraction(){    }  void Fraction::print() const {  cout << numer <<" / "<<denom<<endl;  }  void Fraction::normalize(){  int n = numer, d = denom;  n /= gcd(numer, denom);  d /= gcd(numer, denom);  setNumer(n);  setDenom(d);  }  int Fraction::gcd(int n,int m){  int result = 1;  for(int i = 2 ; i < m; i++){  if(n % i ==0 && m % i == 0){  result = i;  }  }  return result;  }  Fraction &findLargest(Fraction &fr1, Fraction &fr2, Fraction &fr3){  return findLarger(findLarger(fr1,fr2),fr3);  }  Fraction &findLarger(Fraction &fract1, Fraction &fract2){  if(fract1.getNumer()\*fract2.getDenom() > fract2.getNumer() \* fract1.getDenom()){  return fract1;  }  return fract2;  }  int main()  {  cout << "다음과 같은 분수에서 가장 큰 분수를 찾습니다 "<<endl;  Fraction fract1(12, 15);  Fraction fract2(6, 9);  Fraction fract3(4, 11);  cout << "fract1 : "<<fract1.getNumer()<<"/"<<fract1.getDenom()<<endl;  cout << "fract2 : " << fract2.getNumer() << "/" << fract2.getDenom() << endl;  cout << "fract3 : " << fract3.getNumer() << "/" << fract3.getDenom() << endl;  Fraction a = findLargest(fract1,fract2,fract3);  cout << "가장 큰 분수 : " << a.getNumer() <<"/"<<a.getDenom()<<endl;  Fraction fract4(2, 12);  Fraction fract5(15, 25);  Fraction product;  cout << "다음과 같은 분수에 대하여 곱셈과 덧셈 연산을 합니다 "<<endl;  cout << "fract1 : " << fract4.getNumer() << "/" << fract4.getDenom() << endl;  cout << "fract2 : " << fract5.getNumer() << "/" << fract5.getDenom() << endl;  cout << "The result of multiplying is : ";  product.mul(fract4,fract5);  cout << "The result of adding is : ";  product.add(fract4,fract5);  } |
| [실행 결과]  이미지 |

1. <2021\_2\_중간고사> 제시된 결과처럼 실행되는 프로그램을 완성 하시오



class Coffee {

    string name;

    int price;

public:

    Coffee() = default;

    ~Coffee();

    void setName(string n); //매개변수로 받은 값을 멤버 변수 name에 대입

    string getName();        //멤버 변수 name 반환

    void setPrice(int p);    //매개변수로 받은 값을 멤버변수 price에 대입

    int getPrice();         //멤버 변수 price 반환

};

class CoffeeMachine{

    Coffee\* orderlist;

    int count;

public:

    CoffeeMachine();

    ~CoffeeMachine();

    void result();

};

int main() {

    CoffeeMachine cm;

    cm.result();

    return 0;

}

|  |
| --- |
| [프로그램 소스]  #include <iostream>  #include <string>  using namespace std;  //int write(Coffee \*b, int size);  class Coffee  {  string name;  int price;  public:  Coffee() = default;  ~Coffee(){cout << "소멸자 실행" <<endl;}  void setName(string n) { name = n;}  string getName() { return name;}  void setPrice(int p) { price = p;}  int getPrice(){return price;}  };  int write(Coffee \*b, int size);  class CoffeeMachine  {  Coffee \*orderlist;  int count;  public:  CoffeeMachine();  ~CoffeeMachine(){delete[] orderlist;}  void result();  };  CoffeeMachine::CoffeeMachine(){  string name[5] = {"moca", "latte", "americano", "espresso", "sugarcoffee"};  int price[5] = {2000,3000,2000,2500,1500};  cout << "=== 메뉴 ===" <<endl;  for(int i = 0; i < 5; i++){  cout << i<<")"<<name[i]<<","<<price[i]<<endl;  }  cout << endl;  cout << "몇 잔을 주문하시겠습니까 ?";  cin >> count;  orderlist = new Coffee[count];  for(int i = 0; i < count; i ++){  int index = -1;  while( index > 4 || index < 0){  cout << "구매하실 메뉴 번호를 입력하세요 : ";  cin >> index;  }  orderlist[i].setName(name[index]);  orderlist[i].setPrice(price[index]);  }  }  void CoffeeMachine::result(){  cout << "주문하신 메뉴는 "<<endl;  int total = write(orderlist,count);  cout << "합계 금액은 "<<total<<"원 입니다."<<endl<<endl;  }  int write(Coffee \*b, int size)  {  int sum = 0;  for(int i = 0; i < size; i++){  cout << b[i].getName()<<","<<b[i].getPrice()<<endl;  sum += b[i].getPrice();  }  return sum;  }  int main()  {  CoffeeMachine cm;  cm.result();  return 0;  } |
| [실행 결과]  이미지 |