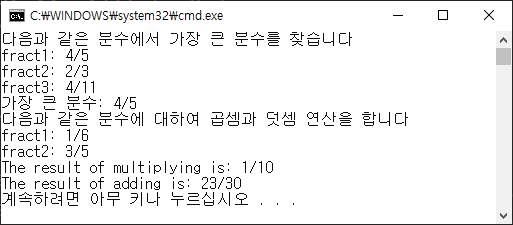
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2021\_1\_C++\_06\_개념활용 | 학번 : | 20185309 | 이름 : | 황명원 |

* 개념 활용

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. 분수는 반드시 기약분수로 출력되어야 한다.

//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 결과는 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:

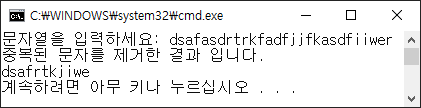
void normalize(); //gcd() 함수를 사용하여 기약분수로 처리

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

};

|  |
| --- |
| [프로그램 소스]  #include <iostream> 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; //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 }; Fraction::Fraction(){} Fraction::Fraction(int num, int den) {  numer=num;  denom=den;  normalize(); } Fraction::~Fraction() {  } int Fraction::getNumer()const{  return numer; } int Fraction::getDenom() const {  return denom; } void Fraction::setNumer(int num) {  numer=num; } void Fraction::setDenom(int den) {  denom=den; } void Fraction::print() const {  cout<<numer<<"/"<<denom<<endl; } void Fraction::normalize() {  int divistor=gcd(numer,denom);  numer=numer/divistor;  denom=denom/divistor; } int Fraction::gcd(int n, int m) {  int gcd=1;  for(int k=1; k<=n && k<=m ;k++ )  {  if(n%k==0 && m%k==0)  gcd=k;  }  return gcd; }  Fraction& findLarger(Fraction& fr1, Fraction& fr2); Fraction& findLargest(Fraction& fr1, Fraction& fr2, Fraction& fr3); Fraction& add(Fraction& fr4 , Fraction& fr5); Fraction& mul(Fraction& fr4 , Fraction& fr5);    int main() {  //가장 큰 값을 찾고자 하는 분수  Fraction fract1(12, 15);  Fraction fract2(6, 9);  Fraction fract3(4, 11);  cout<<"다음과 같은 분수에서 가장 큰 분수를 찾습니다"<<endl;  cout<<"fract1: ";  fract1.print();  cout<<"fract2: ";  fract2.print();  cout<<"fract3: ";  fract3.print();  cout<<"가장 큰 분수는 : ";  findLargest(fract1,fract2,fract3).print();  //곱셈과 덧셈 연산을 수행하는 분수  Fraction fract4(2, 12);  Fraction fract5(15, 25);  Fraction product; //연산 결과는 저장하는 분수   cout<<"다음과 같은 분수에 대하여 곱셈과 덧셈 연산을 합니다."<<endl;  cout<<"fract4: ";  fract4.print();  cout<<"fract5: ";  fract5.print();   product=add(fract4,fract5);  cout<<"The result of adding is: ";  product.print();  product=mul(fract4,fract5);  cout<<"The result of multiplying is: ";  product.print();  } Fraction& findLargest(Fraction& fr1, Fraction& fr2, Fraction& fr3){  return findLarger(findLarger(fr1,fr2),fr3); }  Fraction& findLarger(Fraction& fr1, Fraction& fr2){  if(fr1.getNumer()\*fr2.getDenom()>  fr2.getNumer()\*fr1.getDenom())  return fr1;  return fr2; } Fraction& add(Fraction& fr4 , Fraction& fr5){   int num1;  int num2;  num1=(fr4.getNumer()\*fr5.getDenom()+fr5.getNumer()\*fr4.getDenom());  num2=(fr4.getDenom()\*fr5.getDenom());  Fraction add\_fra(num1,num2);  return add\_fra; } Fraction& mul(Fraction& fr4 , Fraction& fr5){  int num1;  int num2;  num1=(fr4.getNumer()\*fr5.getNumer());  num2=(fr4.getDenom()\*fr5.getDenom());  Fraction mul\_fra(num1,num2);  return mul\_fra; } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 다음 C++ 문자열에서 중복되는 문자를 제거해서 문자 중복이 없게 만드는 함수(AAABBBBCCDEFAA 를 매개변수로 넣었다면, ABCDEF를 리턴)를 작성하고 테스트하는 프로그램을 작성하시오



int main()

{

string strg1;

cout << "문자열을 입력하세요: ";

getline(cin, strg1);

removeDuplicates(strg1); //중복된 문자를 제거하는 함수, 참조에 의한 매개변수 전달

cout << "중복된 문자를 제거한 결과 입니다. " << endl;

cout << strg1<<endl;

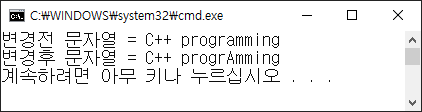
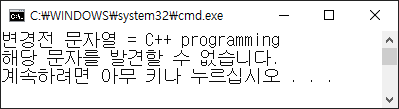
return 0;

}

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std; #include <string>  void removeDuplicates(string& s){   for(int i=0 ; i<s.length()-1 ; i++){  for(int j=i+1 ; j<s.length() ; j++){  if(s.at(i)==s.at(j)){  s.erase(j,j);  j--;  }  }  } }  int main() {  string strg1;  cout << "문자열을 입력하세요: ";  getline(cin, strg1);  removeDuplicates(strg1); //중복된 문자를 제거하는 함수, 참조에 의한 매개변수 전달  cout << "중복된 문자를 제거한 결과 입니다. " << endl;  cout << strg1<<endl;  return 0; } |
| [실행 결과] |

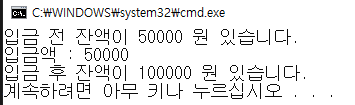
1. 문자열 a에서 문자 c를 찾아 문자 c가 있는 공간에 대한 참조를 리턴 하는 find()함수를 구현하여 제시된 결과처럼 실행되는 프로그램을 작성하시오. 단, find()함수에서 문자 c를 찾을 수 없으면 success 참조 매개변수에 false를 설정한다.

char& find(char a[], char c, bool& success)

|  |
| --- |
| [프로그램 소스]  #include <iostream> #include <string> using namespace std;   char& find(char a[], char c, bool& success);  int main(){  bool suc= false;  char a[]="C++ programming";    cout<<"변경전 문자열 = "<<a<<endl;   find(a,'a',suc)='A';    if (suc) {  cout << "변경후 문자열 = " << a;  } else {  cout << "해당 문자를 발견할 수 없습니다.";  }  }  char& find(char a[], char c, bool& success) {  for (int i = 0; a[i]!='\0'; i++) {  if (a[i] == c) {  success = true;  return a[i];  }   }  return c; } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명  (검색을 z로 했을경우)  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. Account 객체 account의 잔액을 money 만큼 증가 하고자 한다. 제시된 main() 함수를 참고하여 프로그램을 완성하시오



class Account {

int money; //잔액

public:

Account(int money); //매개변수로 받은 값으로 잔액 초기화

int getMoney(); //잔액 반환

void setMoney(int money); //매개변수로 받은 값 만큼 잔액 증가

void show(); //잔액 출력

};

int main() {

Account account(50000);

int money;

cout << "입금 전 "; account.show();

cout << "입금액 : "; cin >> money;

**increaseBy(account, money);**

cout << "입금 후 "; account.show();

}

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std;  class Account {  int money; //잔액 public:  Account(int money); //매개변수로 받은 값으로 잔액 초기화  int getMoney(); //잔액 반환  void setMoney(int money); //매개변수로 받은 값 만큼 잔액 증가  void show(); //잔액 출력 };  Account::Account(int money) { //잔액 생성  this->money=money; } int Account::getMoney() { //잔액 반환  return money; } void Account::setMoney(int money) { //잔액 증가  this->money+=money; } void Account::show() {//잔액 출력  cout<<"잔액이 "<<money<<" 원 있습니다"<<endl; }  void increaseBy(Account &a, int mny);//잔액 증가하는 함수 만들기 int main() {  Account account(50000);  int money;  cout << "입금 전 ";  account.show();  cout << "입금액 : ";  cin >> money;  increaseBy(account, money);  cout << "입금 후 "; account.show(); }  void increaseBy(Account &a, int mny){   a.setMoney(mny); } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |

1. 클래스 Accumulator는 add()함수를 통해 계속 값을 누적하는 클래스로서 다음과 같이 선언된다. Accumulator 클래스를 구현하라

class Accumulator{

int value;

public:

Accumulator(int value); //매개변수 value로 멤버 value를 초기화 한다

Accumulator& add(int n); //value에 n값을 더해 값을 누적한다

int get() const; //누적된 값 value를 리턴한다

}

int main(){

Accumulator acc(10);

acc.add(5).add(6).add(7); //acc의 value 멤버가 28이 된다

Cout<<acc.get(); //28출력

}

|  |
| --- |
| [프로그램 소스]  #include <iostream> using namespace std;  class Accumulator{  int value; public:  Accumulator(int value); //매개변수 value로 멤버 value를 초기화 한다  Accumulator& add(int n); //value에 n값을 더해 값을 누적한다  int get() const; //누적된 값 value를 리턴한다 };  Accumulator::Accumulator(int value) {  this->value=value; } Accumulator &Accumulator::add(int n) {  value+=n;  return \*this;   } int Accumulator::get() const {  return value; }  int main(){  Accumulator acc(10);  acc.add(5).add(6).add(7); //acc의 value 멤버가 28이 된다  cout<<acc.get(); //28출력 } |
| [실행 결과]  텍스트이(가) 표시된 사진  자동 생성된 설명 |