**REPORT**

심화프로그래밍 실습

6주차

**과목명   |** 심화프로그래밍

**담당교수 |** 김가영 교수님

**학과     |** 컴퓨터공학과

**학번     |** 2018112028

**이름     |** 김지훈

**제출일   |**  2020.10.16

실습 1번)

|  |
| --- |
| 문제 분석 :  n개의 숫자(size 지정 x)와 연산자를 입력 받아 가능한 모든 수식을 생성한다. 단, 숫자의 순서는 바꾸면 안된다.  생성된 수식 중에서 제일 큰 수와 작은 수를 재귀적으로 구한다.  저번 실습 class에 getSize() , setStringNum(?), getNum(), changeNum()만 추가한다.  아래의 코드를 실행 했을 때 A의 값이 변화하지 않는다. 1) 변화하지 않는 이유를 작성하시오. (교안 스택 그림 참고)  2)목적대로 수행할 수 있도록 void change(cal A, cal B) 함수를 수정하고 3)수정한 이유를 작성 하시오. |
| 프로그램 설계 / 알고리즘 : |
| 소스코드 / 주석 : |
| 결과 / 결과 분석 :       * 두 결과 모두 max와 min이 잘 계산된다. |

실습 2번)

|  |
| --- |
| 문제 분석 :  다음 코드와 실행 결과를 참고하여 append() 함수를 작성하고 전체 프로그램을 완성 하시오.  append() 함수는 Buffer 객체에 문자열을 추가하고 Buffer 객체에 대한 참조를 반환하는 함수이다.   * 참조를 활용하여 객체에 접근한다. |
| 프로그램 설계 / 알고리즘 :  class Buffer  {  private:  string str;  public:  Buffer(string sStr);  void addStr(string aStr); // 문자열 +  void print(); // 객체의 str을 출력  };  처럼 클래스를 만든다. 추가로 append함수는 Buffer& 반환형으로, Buffer &buf와 aStr를 받아와 buf의 문자에 aStr을 추가하고 buf을 리턴시킨다. |
| 소스코드 / 주석 :   |  | | --- | | // 2018112028 김지훈  #include <iostream>  #include <string>  using namespace std;  class Buffer  {  private:  string str;  public:  Buffer(string sStr);  void setStr(string sStr); // sStr 추가해서 설정  string getStr(); // str 가져오기  void print(); // 객체의 str을 출력  };  Buffer::Buffer(string sStr)  {  this->str = sStr;  }  void Buffer::setStr(string sStr)  {  this->str = sStr;  }  string Buffer::getStr()  {  return this->str;  }  void Buffer::print()  {  cout << this->str << endl;  }  Buffer& append(Buffer &buf, string aStr)  {  buf.setStr(buf.getStr() + aStr);  return buf;  }  int main()  {  Buffer buf("Hello");  Buffer &temp = append(buf, "Guys");  temp.print(); // 출력  buf.print(); // 출력  } | |
| 결과 / 결과 분석 :     * Buf와 temp모두 print할 때, HelloGuys로 잘나온다. |

실습 3번)

|  |
| --- |
| 문제 분석 :  은행과 고객 클래스를 구현하여 입금, 출금 이 가능한 프로그램을 작성하시오.  참조자(&)를 적절히 활용한다.   * 참조자의 적절한 활용 |
| 프로그램 설계 / 알고리즘 :  class Bank  {  private:  int balance; // 잔고  public:  Bank(int money); // 생성자  void setBalance(int money); // 잔고 설정  void getInfo(); // 현재 정보 가져오기  };  class Customer  {  private:  int cash; // 보유 현금  public:  Customer(int money); // 생성자  void getInfo(); // 현재 정보 가져오기  void withDraw(Bank& bank, int money); // 출금  void deposit(Bank& bank, int money); // 입금  };  와 같이 클래스를 설정한다. Customer의 withDraw와 deposit의 매개변수에는 bank의 balance도 수정될 수 있게 참조자를 이용하여 불러온다. |
| 소스코드 / 주석 :   |  | | --- | | // 2018112028 김지훈  #include <iostream>  using namespace std;  class Bank  {  private:  int balance; // 잔고  public:  Bank(int money); // 생성자  void setBalance(int money); // 잔고 설정  void getInfo(); // 현재 정보 가져오기  };  class Customer  {  private:  int cash; // 보유 현금  public:  Customer(int money); // 생성자  void getInfo(); // 현재 정보 가져오기  void withDraw(Bank& bank, int money); // 출금  void deposit(Bank& bank, int money); // 입금  };  Bank::Bank(int money)  {  this->balance = money;  }  Customer::Customer(int money)  {  this->cash = money;  }  void Bank::setBalance(int money)  {  this->balance += money;  }  void Bank::getInfo()  {  cout << "은행잔고: " << this->balance << endl;  }  void Customer::getInfo()  {  cout << "보유현금: " << this->cash << endl << endl;  }  void Customer::withDraw(Bank& bank, int money)  {  this->cash += money;  bank.setBalance(-money); // 해당 매개변수를 +해주기 때문에 -를 붙여준다.  cout << "출금->" << money << endl;  }  void Customer::deposit(Bank& bank, int money)  {  this->cash -= money;  bank.setBalance(money);  cout << "입금->" << money << endl;  }  int main()  {  Bank bank(2000); // 잔고 2000원  Customer customer(1000); // 보유현금 1000원  customer.getInfo();  customer.withDraw(bank, 500);  bank.getInfo();  customer.getInfo();  customer.deposit(bank, 1000);  bank.getInfo();  customer.getInfo();  } | |
| 결과 / 결과 분석 :     * 은행 잔고의 동기화도 잘 이루어진다. |

실습 4번)

|  |
| --- |
| 문제 분석 :  정수 3개의 값을 서로 바꿔주는 swap() 함수를 구현해보자.  참조에 의한 인수 전달로 swap() 함수를 구현한다.  x는 y로, y는 z로, z는 x로 서로의 값을 변경한다.  주소에 의한 인수 전달로 swap() 함수를 구현하고 그 결과를 출력하시오.  참조에 의한 인수 전달로 swap() 함수를 구현하고 그 결과를 출력하시오.   * 참조에 “의한”, swap()함수 구현 |
| 프로그램 설계 / 알고리즘 :  void swapByAdress(int\* a, int\* b, int\* c);  void swapByReference(int& a, int& b, int& c);  이렇게 다른 참조방식의 두 함수를 만든다. |
| 소스코드 / 주석 :   |  | | --- | | // 2018112028 김지훈  #include <iostream>  using namespace std;  void swapByAdress(int\* a, int\* b, int\* c)  {  int \*tmp = a;  a = b;  b = c;  c = tmp;  }  void swapByReference(int& a, int& b, int& c)  {  int &tmp = a;  a = b;  b = c;  c = tmp;  }  int main()  {  int x = 1;  int y = 10;  int z = 100;  cout << "변경전 : " << x << ", " << y << ", " << z << endl << endl; // 변경 전 출력    cout << "주소의 의한 인수 전달" << endl;  swapByAdress(&x, &y, &z); // 주소 값을 인자로 넣어줌  cout << "변경후 : " << x << ", " << y << ", " << z << endl; // 출력  // 다시 원래 값으로 초기화  x = 1;  y = 10;  z = 100;  cout << "참조에 의한 인수 전달" << endl;  swapByReference(x, y, z);  cout << "변경후 : " << x << ", " << y << ", " << z << endl; // 출력    return 0;  } | |
| 결과 / 결과 분석 :     * 참조에 의한 인수 전달만 값이 바뀐 것을 확인할 수 있었다. |

소감 : 참조에 대하여 더 잘 이해할 수 있는 과제였다. 값에 의한 전달, 주소에 의한 전달, 참조에 의한 전달 등. 생각한대로 값이 그대로 옮겨지는 게 아니라는 것을 알았다.