ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
ДИМИТРОГРАДСКИЙ ИНСТИТУТ ТЕХНОЛОГИЙ,
УПРАВЛЕНИЯ И ДИЗАЙНА

Лабораторная работа №6
по курсу: “ООП”
“ *Конструктор копирования и оператор присваивания*”
Вариант №6

Выполнил студент гр. ВТ-31: Потеренко А.Г.
Проверил преподаватель: Наскальнюк А.Н.

Димитровград 2006 г.

Содержание.

Задание для работы.

Создать иерархию классов вектор и безопасный вектор с проверкой выхода за пределы. Безопасный вектор определяет нижний и верхний предел. Переопределить вывод в поток и ввод из потока, оператор присваивания через соответствующие функции базового класса.

Теория к заданиям.

1. Операция присваивания определена в любом классе по умолчанию как поэлементное копирование. Эта операция вызывается каждый раз, когда одному существующему объекту присваивается значение другого. Если класс содержит поля, память под которые выделяется динамически, необходимо переопределить данную операцию. Чтобы сохранить семантику присваивания:

- операция-функция должна возвращать ссылку на объект, для которого она вызвана
- принимать в качестве параметра единственный аргумент – ссылку на присваиваемый объект.

Операция присваивания определяется только как метод класса. Она не наследуется.

```
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class CLASS_A
  {
    private:
     float *x;
     unsigned int size;
    public:
     CLASS_A(unsigned int BEGIN);
     ~CLASS_A();
     void PROC(unsigned int h,unsigned int i);
     float & operator  = (unsigned int i);
  };
//---------------------------------------------------------------------------
void CLASS_A::PROC(unsigned int h,unsigned int i)
{
  x[i]=h;
};
//---------------------------------------------------------------------------
CLASS_A::~CLASS_A()
{
  delete[] x;
};
//---------------------------------------------------------------------------
CLASS_A::CLASS_A(unsigned int BEGIN)
{
  size=BEGIN;  //size = число элементов динамического массива
  x= new float [size];
};
//----------------Перегрузка операции присваивания-----------------------------
float & CLASS_A::operator  = (unsigned int i)
{
  if (i<0 || i>=size)
    {
      ShowMessage("Вы ошиблись с индексом!");
      exit(0);
    };
  return x[i];
};
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
```

```
__published:            // IDE-managed Components
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:                // User declarations
public:                      // User declarations
        __fastcall TForm1(TComponent* Owner);
};
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#include <vcl.h>
#include "Unit1.h"
#pragma hdrstop
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner): TForm(Owner){}
//////////////////////////////////////////////////////////////////////////////
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 CLASS_A CL1(20);     //Создаем массив из 20 чисел типа unsigned int
 CL1.PROC(230,19);     //x[19]=230
 int BI=0;
 BI=CL1.operator =(19);  //BI=x[19];
}
```

2. Синтаксис оператора << и >> следующий:

```
inline ostream& operator << (ostream& os, Currency& arg); //экран
inline istream& operator >> (istream& is, Currency& arg); //клавиатура
```

Важно перегрузить данный оператор для вывода пользовательских типов. Напишем программу, использующую данную возможность:

Алгоритм работы программ.

Создаем два класса. Первый – просто класс вектора. Второй – предусматривает защиту от пользователя. В нем проверяется все – задание значений индексов вектора.
PT – указатель на динамическую переменную – динамический вектор. SIZE – размер вектора.
CLASS_A() – конструктор класса.
int SHOWSIZE() – показывает порядок вектора.
int SHOWPT(int i) – возвращает элемент вектора по его индексу.
void NEWVECT() – выделяет память под вектор и заполняет его случайным образом.
virtual bool Attention(int p) – проверка на ввод пользователя порядка вектора. В безопасном векторе пользователю не дается возможность ввести некорректный данные. В классе CLASS_B данная функция заменяется другой (для безопасного вектора).
int& operator =(int i) – перегруженный оператор присваивания.
ostream& operator << (ostream& os, CLASS_A &M) – перегруженные операции вывода – выводят весь вектор целиком.

Листинги.

```
#pragma hdrstop
#pragma argsused
#include <stdlib.h>
#include <ostream.h>
#include <iostream.h>
#include <conio.h>
//////////////////////Класс ВЕКТОР/////////////////////////////////////////////
class CLASS_A
  {
    protected:   //Данные поля доступны в производных классах и не доступны пользователю
      int * PT;
      unsigned int SIZE;  //размер вектора
    public:
```

```
      CLASS_A ()
       {
         PT=NULL;
         SIZE=0;
       };
      int SHOWSIZE()
        {
          return SIZE;
        };
      int SHOWPT(int i)
        {
          return PT[i];
        };
      void NEWVECT()
        {
           randomize();
           PT = new int [SIZE];
           for (int j=0;j<SIZE;j++)
            {
              PT[j]=random(100);
            }
        };
      ~CLASS_A ()
       {
         if (PT!=NULL) //Если мы выделили память под вектор
           delete [] PT;
       };
       virtual bool Attention(int p)
        {
          cout << "RAZMER MASSIVA:";
          cout << p;
          cout << endl;
          SIZE=p;
          return true;
        }
       int& operator =(int i)
        {
            if (i<0 || i>=SIZE)
              {
                cout << endl;
                cout << "INDEX=ERROR!";
                getch();
                exit(0);
              };
            return PT[i];
        };
  };
//////////////////Перегрузка операций вывода//////////////////////////////////
ostream& operator << (ostream& os, CLASS_A &M)
 {
   for (int i=0;i<M.SHOWSIZE();i++)
          {
             os << M.SHOWPT(i);
             os << "   ";
             cout << endl << endl;
          };
 };
//////////////////////Класс Безопасный вектор//////////////////////////////////
class CLASS_B: public CLASS_A
  {
    public:
      CLASS_B ()  //Конструктор не наследуется
       {
         PT=NULL;
       };
      bool Attention(int p)
        {
          if (p>=0 && p<=100)
           {
            cout << "RAZMER MASSIVA:";
            cout << p;
            cout << endl;
            SIZE=p;
            return true;
           }
          else
           {
```

```
              cout << "VNIMANIE!!! PORADOK VECTORA - ERROR!!!";
              getch();
              exit(0);
            };
         }
  };
////////////////////////////////////////////////////////////////////////////
int main(int argc, char* argv[])
{
   int a;
   cout << "BBEDITE PORADOK VECTORA:" << endl;
   cin >> a;
   cout << endl;
   ////////////////////////////////////////////////////////////////////////////
   CLASS_A CL1;
   CLASS_A *UK;  //Указатель на класс
   UK=&CL1;      //Инициализируем
   if (UK->Attention(a)==true)
    {
     CL1.NEWVECT();
     operator << (cout,CL1);
    };
   ////////////////////////////////////////////////////////////////////////////
   cout << "BBEDITE PORADOK BEZOPASNOGO VECTORA (<101):" << endl;
   cin >> a;
   cout << endl;
   CLASS_B CL2;
   UK=&CL2;  //Переопределяем адрес
   if (UK->Attention(a)==true)
    {
      CL2.NEWVECT();
      operator << (cout,CL2);
    };
   /////////Выводим элемент безопасного вектора по индексу пользователя/////////
   cout << endl;
   cout << "BBEDITE INDEX:";
   cin >> a;
   int BI=0;
   BI=UK->operator =(a-1);
   cout << endl;
   cout << "VECTOR[" << a <<"]=" << BI;
   getch();
}
```