Титульный лист

Цель работы

Цель работы – приобрести практические навыки по использованию механизмов наследования и виртуальных функций при разработке объектно-ориентированных программ.

Для достижения поставленной цели надо решить следующие главные задачи:

- ознакомиться с теоретическими сведениями; - программно реализовать иерархию классов;

- выполнить тестирование программного кода;

- оформить отчет по проделанной работе

Задание

Вариант 4

Разработать абстрактный базовый класс вектор (одномерный массив произвольной длины) с виртуальным методом сумма (сумма элементов вектора) и производные классы вектор_целых_чисел (int*), вектор_чисел_float (float*) и вектор_чисел_double (double*) со своими реализациями этого метода

Описание программы

Согласно заданию необходимо разработать абстрактный класс, от которого будут наследоваться остальные классы, и переопределять методы базового класса. Для удобства работы с программой в программе необходимо создать меню для выбора действий.

Базовый класс имеет следующую структуру:

```
class Vector
{
public:
     int count = 0;
     virtual void sum() = 0;
     virtual void generating_numbers() = 0;     //    заполнение    вектора
случайными значениями
     virtual void print() = 0;
};
```

Листинг 1. Класс Vector

Класс VectorInt имеет следующее определение:

```
class VectorInt : Vector
{
public:
	int* mass;

	VectorInt(int n)
	{
		mass = new int[n] {0};
		count = n;
	}

	void sum()
	{
		int s = 0;
		for (int i = 0; i < count; i++)
		{
			s += mass[i];
		}
		cout << "Сумма чисел: " << s << endl;
	}

	void generating_numbers()
	{
		random_device rd;
		mt19937_64 gen(rd());
		uniform_int_distribution<unsigned long long> dis;
		for (int i = 0; i < count; i++)
		{
			mass[i] = dis(gen) % 100;
		}
	}

	void print()
	{
		for (int i = 0; i < count; i++)
		{
			cout << mass[i] << " ";
		}
		cout << endl << endl;
	}
};
```

Листинг 2. Класс VectorInt

В классе переопределены функции базового класса.

По аналогии реализованы остальные классы, указанные в задание. Листинг реализаций классов приведен в приложение.

В функции main реализовано следующее меню:

```
cout << "Меню" << endl;
cout << "1. Создать вектор с типом Int" << endl;
cout << "2. Создать вектор с типом Float" << endl;
cout << "3. Создать вектор с типом Double" << endl;
cout << "4. Вывод вектора с типом Int" << endl;
cout << "5. Вывод вектора с типом Float" << endl;
cout << "6. Вывод вектора с типом Double" << endl;
cout << "7. Вывод суммы элемеентов вектора с типом Int" << endl;
cout << "8. Вывод суммы элемеентов вектора с типом Float" << endl;
cout << "9. Вывод суммы элемеентов вектора с типом Double" << endl;
cout << "0. Выход" << endl << endl;
```

Листинг 3. Меню

С его помощью можно работать с каждым классом отдельно.

Тестирование

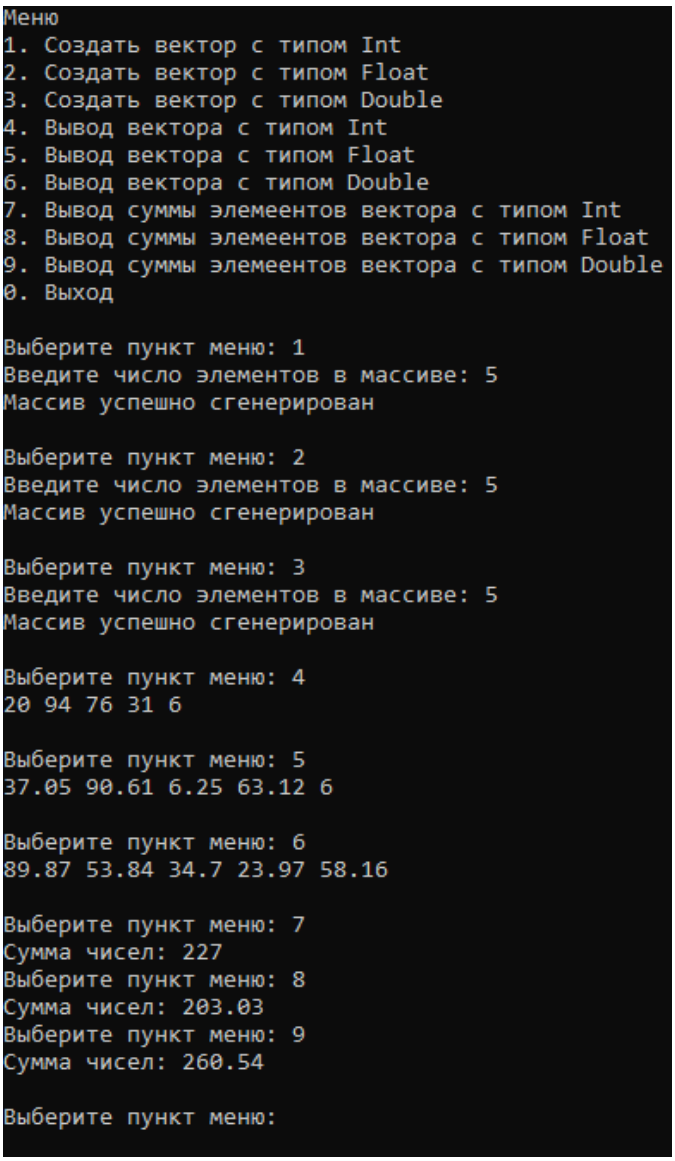

Рисунок 1. Тест

Вывод по работе

В ходе выполнения работы было успешно разработано приложение по заданной теме. Приложение успешно прошло тестирование и показало высокую надежность. Были выполнены все заявленные цели лабораторной работы.

Список литературы

1. Campbell Parallel Programming with Microsoft® Visual C++® / Campbell. - Москва: Гостехиздат, 2018. - 784 с.

2. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - Москва: Высшая школа, 2017. - 882 с.

3. Балена, Франческо Современная практика программирования на C++ / Франческо Балена , Джузеппе Димауро. - М.: Русская Редакция, 2019. - 640 с.

4. Боровский, А. C++ / А. Боровский. - М.: БХВ-Петербург, 2020. - 544 с.

5. Давыдов, В. Visual C++. / В. Давыдов. - М.: БХВ-Петербург, 2019. - 576 с.

Приложение

Листинг

main.cpp

```
// lab_1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.
//

#include "Header.h"


int main()
{
	setlocale(LC_ALL, "Russian");  // подключение кириллицы
	cout << "Меню" << endl;
	cout << "1. Создать вектор с типом Int" << endl;
	cout << "2. Создать вектор с типом Float" << endl;
	cout << "3. Создать вектор с типом Double" << endl;
	cout << "4. Вывод вектора с типом Int" << endl;
	cout << "5. Вывод вектора с типом Float" << endl;
	cout << "6. Вывод вектора с типом Double" << endl;
	cout << "7. Вывод суммы элемеентов вектора с типом Int" << endl;
	cout << "8. Вывод суммы элемеентов вектора с типом Float" << endl;
	cout << "9. Вывод суммы элемеентов вектора с типом Double" << endl;
	cout << "0. Выход" << endl << endl;
	string select;
	VectorInt* vectorInt = nullptr;			// вектор с типом Int
	VectorFloat* vectorFloat = nullptr;		// вектор с типом float
	VectorDouble* vectorDouble = nullptr;// вектор с типом double
	int n = 0;
	while (true)
	{
		cout << "Выберите пункт меню: ";
		cin >> select;	// выбор пункта меню
		switch (select[0])
		{
		case '1':	// создание вектора с типом int
			n = input("Введите число элементов в массиве: ");
```

```
                vectorInt = new VectorInt(n);
                vectorInt->generating_numbers();
                cout << "Массив успешно сгенерирован" << endl << endl;
                break;
        case '2':    // создание вектора с типом float
                n = input("Введите число элементов в массиве: ");
                vectorFloat = new VectorFloat(n);
                vectorFloat->generating_numbers();
                cout << "Массив успешно сгенерирован" << endl << endl;
                break;
        case '3':    // создание вектора с типом double
                n = input("Введите число элементов в массиве: ");
                vectorDouble = new VectorDouble(n);
                vectorDouble->generating_numbers();
                cout << "Массив успешно сгенерирован" << endl << endl;
                break;
        case '4':    // вывод вектора с типом int
                if (vectorInt == nullptr)
                {
                        cout << "Список не был создан. Создайте список" << endl
<< endl;
                        continue;
                }
                vectorInt->print();
                break;
        case '5':    // вывод вектора с типом float
                if (vectorFloat == nullptr)
                {
                        cout << "Список не был создан. Создайте список" << endl
<< endl;
                        continue;
                }
                vectorFloat->print();
                break;
        case '6':    // вывод вектора с типом double
                if (vectorDouble == nullptr)
                {
                        cout << "Список не был создан. Создайте список" << endl
<< endl;
                        continue;
                }
                vectorDouble->print();
                break;
        case '7':    // вывод суммы элемеентов вектора с типом Int
                if (vectorInt == nullptr)
                {
                        cout << "Список не был создан. Создайте список" << endl
<< endl;
                        continue;
                }
                vectorInt->sum();
                break;
        case '8':    // вывод суммы элемеентов вектора с типом float
                if (vectorFloat == nullptr)
                {
```

```
                        cout << "Список не был создан. Создайте список" << endl
<< endl;
                        continue;
                  }
                  vectorFloat->sum();
                  break;
            case '9':   // вывод суммы элемеентов вектора с типом double
                  if (vectorDouble == nullptr)
                  {
                        cout << "Список не был создан. Создайте список" << endl
<< endl;
                        continue;
                  }
                  vectorDouble->sum();
                  break;
            case '0':
                  return 0;
            default:
                  break;
            }
      }
}
```

Header.h

```
#pragma once
/*
Разработать абстрактный базовый класс вектор (одномерный массив произвольной
длины)
с виртуальным методом сумма (сумма
элементов вектора) и
производные классы вектор_целых_чисел (int*),
вектор_чисел_float (float*) и
вектор_чисел_double (double*)
со своими реализациями этого метода
*/

#include <iostream>
#include <random>
#include <string>
using namespace std;

int input(string mess)   // ввод числа с проверкой на корректность
{
      int val = 0;
      string text;
      while (true)
      {
            try
            {
                  cout << mess;
                  cin >> text;
                  val = stoi(text);
                  if (val > 0) // число должно быть положительное
```

```
                        return val;
                else
                        cout << "Введите положительное число" << endl;
        }
        catch (...)
        {
                cout << "Некорреткный ввод, повторите" << endl;
        }
    }
}

class Vector // базовый класс веткора
{
public:
    int count = 0;                  // число элементов в векторе
    virtual void sum() = 0;  // функция вывода суммы элементов вектора
    virtual void generating_numbers() = 0;      // заполнение вектора случайными
значениями
    virtual void print() = 0;       // печать списка
};

class VectorInt : Vector // вектор с типом Int
{
public:
    int* mass;

    VectorInt(int n)   // конструктор с параметрами
    {
        mass = new int[n] {0};
        count = n;
    }

    void sum()   // перегруженная функция базового класса для вывода суммы
элементов
    {
        int s = 0;
        for (int i = 0; i < count; i++)
        {
            s += mass[i];      // суммирование элементов списка
        }
        cout << "Сумма чисел: " << s << endl;
    }

    void generating_numbers()       // генерация случайных чисел
    {
        random_device rd;
        mt19937_64 gen(rd());
        uniform_int_distribution<unsigned long long> dis;
        for (int i = 0; i < count; i++)
        {
            mass[i] = dis(gen) % 100;       // генерация числе от 0 до 100
        }
    }

    void print()
    {
```

```
            for (int i = 0; i < count; i++)
            {
                  cout << mass[i] << " ";  // вывод чисел в строку
            }
            cout << endl << endl;
      }
};

class VectorFloat : Vector
{
public:
      float* mass;

      VectorFloat(int n)
      {
            mass = new float[n] {0};
            count = n;
      }

      void sum()  // перегруженная функция базового класса для вывода суммы
элементов
      {
            float s = 0;
            for (int i = 0; i < count; i++)
            {
                  s += mass[i];      // суммирование элементов списка
            }
            cout << "Сумма чисел: " << s << endl;
      }

      void generating_numbers()      // генерация случайных чисел
      {
            random_device rd;
            mt19937_64 gen(rd());
            uniform_int_distribution<unsigned long long> dis;
            for (int i = 0; i < count; i++)
            {
                  mass[i] = dis(gen) % 10000 / 100.0;  // генерация чисел от 0.0
до 100.0
            }
      }

      void print() // вывод списка на печать
      {
            for (int i = 0; i < count; i++)
            {
                  cout << mass[i] << " ";  // вывод чисел в строку
            }
            cout << endl << endl;
      }
};

class VectorDouble : Vector
{
public:
      double* mass;
```

```
	VectorDouble(int n)
	{
		mass = new double[n] {0};
		count = n;
	}

	void sum()  // перегруженная функция базового класса для вывода суммы
элементов
	{
		double s = 0;
		for (int i = 0; i < count; i++)
		{
			s += mass[i];	// суммирование элементов списка
		}
		cout << "Сумма чисел: " << s << endl << endl;
	}

	void generating_numbers()	// генерация случайных чисел
	{
		random_device rd;
		mt19937_64 gen(rd());
		uniform_int_distribution<unsigned long long> dis;
		for (int i = 0; i < count; i++)
		{
			mass[i] = dis(gen) % 10000 / 100.0;  // генерация чисел от 0.0
до 100.0
		}
	}

	void print() // вывод списка на печать
	{
		for (int i = 0; i < count; i++)
		{
			cout << mass[i] << " ";  // вывод чисел в строку
		}
		cout << endl << endl;
	}
};
```