МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РД
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РЕСПУБЛИКИ ДАГЕСТАН
«КОЛЛЕДЖ ЭКОНОМИКИ И ПРАВА»

Отчет
по учебной практике

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем»
09.02.07«Информационные системы и программирование»

Выполнил студент 3 курса

Группы 39 ИСП 2
Исмаилов Тарлан Низамиевич
(Ф.И.О. студента)
Руководитель практики
Преподаватель технических дисциплин
_____ А.Н.Магомедова

_____
дата оценка

Дербент – 2022 г.

СОДЕРЖАНИЕ

1. ДНЕВНИК ПО УЧЕБНОЙ ПРАКТИКЕ

1.1. Индивидуальный план выполняемых заданий по учебной практике

№ п/п	Наименование	Сроки	Отметка о выполнении
1	Введение и ознакомление с программой учебной практики	2.12.2022	
2	Основы C++, ООП		
3	Списки, кортежи и словари	3.12.2022	
4	Работа с файлами	5.12.2022	
5	Основные встроенные модули		
6	Разработка программного кода для каждого задания на выбранном языке программирования.	6.12.2022	
7	Отладка программных модулей	7.12.2022-	
8	Тестирование программных модулей	8.12.2022	
9	Оптимизация программного кода	9.12.2022-	
10	Разработка условия практической задачи	11.12.2022	

1.2. Тематический план выполняемых работ по учебной практике

№ п/п	Виды работ	Содержание работ	Оценка работы
1	Введение и ознакомление учебной практикой	Ознакомление с местом прохождения практики. Инструкция по технике безопасности, пожарной безопасности и должностные инструкции	
2	Основы С++. Объектно-ориентированное программирование	Написание программ на Microsoft visual studio C++, а также закрепление пройденного материала основ С++ (переменные, функции, типы данных, циклы и т.д.) и свойства ООП.	
3	Списки, кортежи и словари	Изучить списки, кортежи и словари с++	
		Разработка программ включая списки, кортежи и словари	
4	Работа с файлами	1Изучения открытие и закрытие файлов. Бинарные файлы. Модуль OS и работа с файловой системой	
		2. Создать объект класса в режиме записи, связать объект класса с файлом, в который будет производиться запись. записать строку в файл и закрыть файл.	
5	Основные встроенные модули	1. Изучение модулей random , math, модуль locale и модуль decimal	
		2. Написание программ с использованием одно из модулей	
		Запуск программы на выполнение	
6	Разработка программного кода для каждого задания на выбранном языке программирования.	1.Выбрать язык реализации программных продуктов, исходя из разработанной спецификации.	
		2.Разработать программный код для каждого значения на языке программирования.	
		3. Проверка соответствие кода со спецификацией программного продукта.	
7	Отладка программных модулей.	Выполнить отладку разработанных программ.	
		1. В случае найденных несоответствий спецификации, исправить выявленные ошибки и дефекты.	
		2. Провести автоматическую отладку средствами выбранной автоматизированной системы.	
8	Тестирование программных модулей.	Разработать план тестирования программных модулей.	
		Исправить выявленные при тестировании ошибки.	
9	Оптимизация программного кода.	Оценить программные модули с точки зрения эффективности использования ресурсов.	

		При выявленной необходимости предложить пути оптимизации.	
		Анализ оптимизации	
10	Разработка условия практической задачи	Разработать техническую документацию (техническое задание, условие задач) для готовых программных модулей.	
		Составить условие задачи практической направленности, при решении которой можно было бы использовать данный программный код	
		Проверка условия задания	

Студент___Исмаилов Тарлан Низамиевич_____

(Ф.И.О. студента)

Дата _____ Подпись _____

ВВЕДЕНИЕ

Целью прохождения учебной практики является получение представлений о практической части специальности и её конкретизация, укрепление ранее полученных теоретических знаний, в процессе обучения и приобретение необходимых навыков непосредственно самой практической работы с прикладными программами.

Основные **задачи** учебной практики состоят в следующем:

1. Ознакомиться с сутью и значением в обществе будущей профессии
2. Разработать модульный программный продукт на основе готовых спецификаций.
3. Обучиться рефакторингу кода.
4. Заняться отладкой программных модулей с использованием специализированных программных средств.
5. Осуществить тестирование программных модулей одним возможных способов.

Учебную практику проходил на базе Государственного бюджетного профессионально образовательного учреждения Республики Дагестан "Колледж экономики и права", расположенный по адресу, Республика Дагестан, город Дербент, переулок С. Стальского, дом 26 в период с 02.12.2022 по 12.02.2022 гг.

2. Теоретическое обоснование прохождения учебной практики

2.1. Основы C++. Объектно-ориентированное программирование

Языку C++ почти 40 лет, но он по-прежнему актуален и востребован: сегодня он применяется повсеместно — от разработки игр и десктопных программ до «мозгов» умного дома 1) поддержка абстракции данных (типов данных, определяемых пользователем);

При написании программ в языках C и C++ используются следующие понятия:

- алфавит;
- константы;
- идентификаторы;
- ключевые слова;
- комментарии.

Язык C++ представляет собой набор команд, которые говорят компьютеру, что необходимо сделать. Командами являются или «функции» или «ключевые слова». Ключевые слова (зарезервированные слова C/C++) являются основными строительными блоками языка. Функции являются сложными строительными блоками, так как записаны они в терминах более простых функций — вы это увидите в нашей самой первой программе, которая показана ниже. Такая структура функций напоминает содержание книги. Содержание может показывать главы книги, каждая глава в книге может иметь своё собственное содержание, состоящее из пунктов, каждый пункт может иметь свои подпункты. Хотя C++ предоставляет много общих функций и зарезервированных слов, которые вы можете использовать, все-таки возникает потребность в написании своих собственных функций.

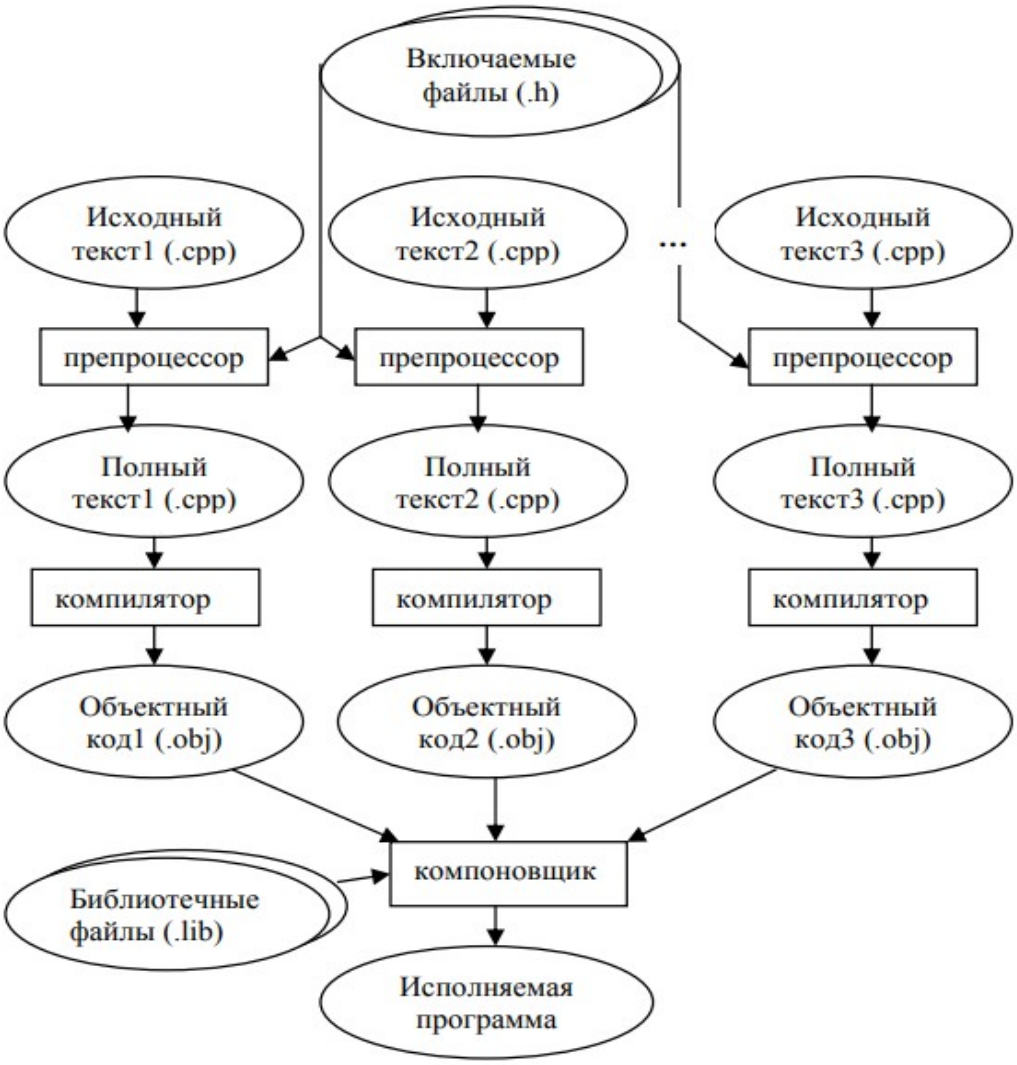


Рис1. Этапы создания исполняемой программы.

Класс - в объектно-ориентированном программировании, представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов.

Создав класс можно создать его экземпляр — объект. **Объект** — это функционирующий прототип класса, которому можно задавать свойства и вызывать методы.

У каждого созданного класса могут быть свойства и методы. **Свойства** — это все что может хранить информацию, которую вы потом можете заполнять (переменные, массивы и т.д.).

Методы — это обычные функции, в функционале которых можно использовать свойства.

В ООП входит такие свойства:

Инкапсуляция — это возможность задавать разную область видимости определенной части класса.

Наследование — это свойство создавать новый класс на базе старого. Такие классы называют потомками.

Полиморфизм — возможность создать объекты с одинаковым интерфейсом, но с разной их реализацией

Абстракция — это возможность выбирать только те свойства или функции, которые нам необходимы. Например, при создании класса про работника понадобится указать его имя, возраст, образование, но никак его цвет волос, глаз, рост и тому подобное.

2.2. Списки, кортежи и словари

Кортеж - коллекция элементов с фиксированным размером. Любая связанная пара, тройка, четверка и т.д элементов является кортежем. В качестве элементов кортежа могут выступать переменные произвольного типа.

Кортежи можно создавать, сравнивать, распаковывать. Некоторые языки позволяют менять значения в элементах кортежей. C++ не имеет встроенных кортежей. Первая реализация кортежей для C++ появилась в библиотеке boost в 2003 году.

Путем инстанцирования шаблона std::tuple<> можно создавать кортежи

```
std::tuple<int> t1;
std::tuple<int, std::string> t2;
```

Можно также задавать значения у элементов кортежа путем их последовательного перечисления в конструкторе:

```
std::tuple<int> t1(1);
std::tuple<int, std::string> t2(1, "hello");
```

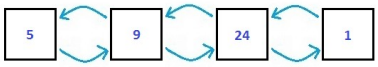
Если элемент кортежа является ссылочным типом, то его значение должно быть определено:

```
std::tuple<double&> t(); // error
double d = 5;
std::tuple<double&> t(d); // ok
```

Класс list(список)

Класс списка стандартной библиотеки C++ — это шаблон класса контейнеров последовательностей, которые поддерживают их элементы в линейном расположении и разрешают эффективные вставки и удаления в любом расположении в последовательности. Последовательность хранится в

виде двунаправленного связанного списка элементов, каждый из которых содержит элемент определенного типа *Type*.



У двусвязного списка нет индексов, но вместо их в C++ есть итераторы.

```
i_am_list[2] = 8;  // ошибка!
```

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, по произвольному, называется **словарем** или ассоциативным массивом. Ассоциативные контейнеры обеспечивают быстрый доступ к данным по ключу. Такие контейнеры построены на основе сбалансированных деревьев. К ассоциативным контейнерам относятся: словари (map), словари с дубликатами (multimap), множества (set), множества с дубликатами (multiset) и битовые множества (bitset). Для использования указанных контейнеров необходимо подключить заголовочные файлы: map, set, bitset.

2.3. Основные встроенные модули

Программа C++ почти всегда состоит из нескольких раздельно транслируемых "модулей". Каждый "модуль" обычно называется исходным файлом, но иногда - единицей трансляции. Он состоит из последовательности
описаний типов, функций, переменных и констант. Описание extern позволяет
из одного исходного файла ссылаться на функцию или объект, определенные в другом исходном файле. Например:

```
extern "C" double sqrt ( double );
extern ostream cout;
```

Самый распространенный способ обеспечить согласованность описаний
внешних во всех исходных файлах - поместить такие описания в специальные файлы, называемые заголовочными. Заголовочные файлы можно включать во все исходные файлы, в которых требуются описания внешних. Например, описание функции sqrt хранится в заголовочном файле стандартных математических функций с именем math.h, поэтому, если нужно извлечь квадратный корень из 4, можно написать:

```
#include <math.h>
//...
```

```
x = sqrt ( 4 );
```

2.4. Оптимизация программного кода

Оптимизация кода или **оптимизация программного** обеспечения — это один из способов преобразования кода, приводящий к улучшению его характеристик и повышению производительности программы. Среди целей оптимизации можно выделить уменьшение размера кода, объема используемой оперативной памяти, повышение скорости выполнения программы, уменьшение количества операций ввода – вывода. При оптимизации важно провести оценку соответствующего кода и полностью понять, как будет использоваться код.

Проблемы оптимизации

В рамках процесса оптимизации может возникнуть множество проблем на разных его этапах. Эти проблемы включают в себя как программный характер, так и человеческий фактор.

Время оптимизации на прямую зависит от количества параметров, интервалов их возможных значении и количества тиков в истории. Также в тестировщике стратегий можно указывать ограничения на показатели работы системы, которые прекратят тестировать конкретный набор параметров при выходе этих значений из зоны ограничений.

Абсурдные результаты часто возникают при отсутствии здравых и четко описанных ограничениях на возможных комбинациях параметров. Проблема в том, что система не может отделять такие результаты от остальных и выводит показатели системы вместе с остальными. Это значительно увеличивает время поиска оптимальных значений параметров после завершения процесса оптимизации.

Скрытые ошибки в коде программа сама по себе не может определить точность следования за стратегией, которую задумал автор. Часто получается так, что не учтены какие-либо мелкие возможные условия рынка, и система выдает результаты для алгоритма, который не полностью соответствует задуманному.

2.5. Тестирование и отладка программного обеспечения

Отладка и **тестирование** — это два четко различимых и непохожих друг на друга этапа:

- при отладке происходит локализация и устранение синтаксических ошибок и явных ошибок кодирования
- в процессе же тестирования проверяется работоспособность программы, не содержащей явных ошибок

Тестирование должно заранее планироваться и систематически проводиться специальными независимыми специалистами — тестировщиками. **Тестировщик** осуществляет поиск вероятных ошибок и сбоев в функционировании программного обеспечения, моделирует различные ситуации, которые могут возникнуть в процессе использования программного продукта. **Тестирование завершается** созданием отчетом о результатах тестирования.

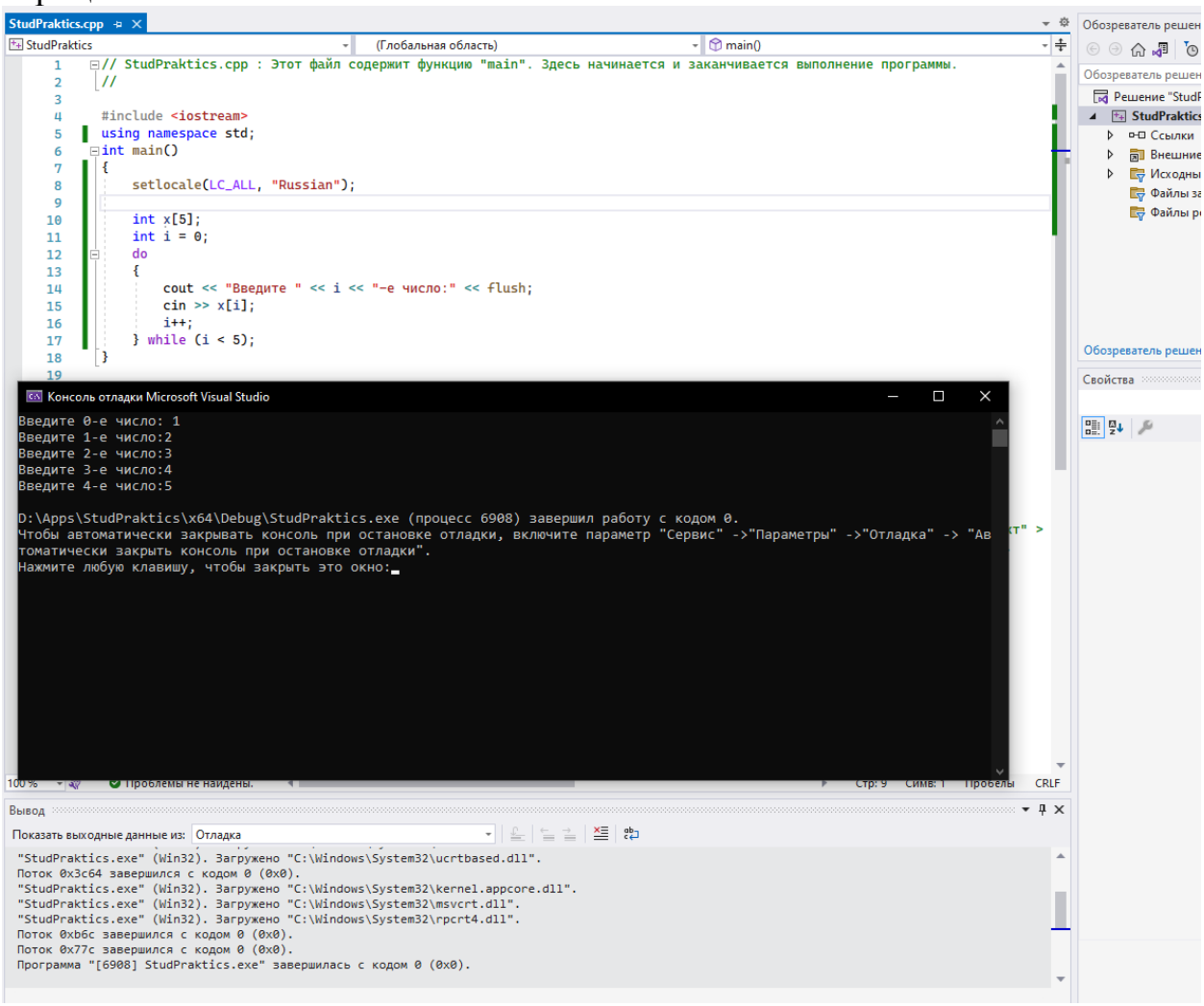
К тестированию относятся отладка, контроль и испытание**.**

Отладка — тестирование программного кода на этапе разработки программного обеспечения. ***Контроль*** — поиск ошибок при выполнении программ в тестовой или моделируемой среде. ***Испытание*** — попытка найти ошибки при выполнении программы в реальной среде.

3. Практическое применение модулей программного обеспечения для компьютерных систем

3.1. Задания по системному программирование.

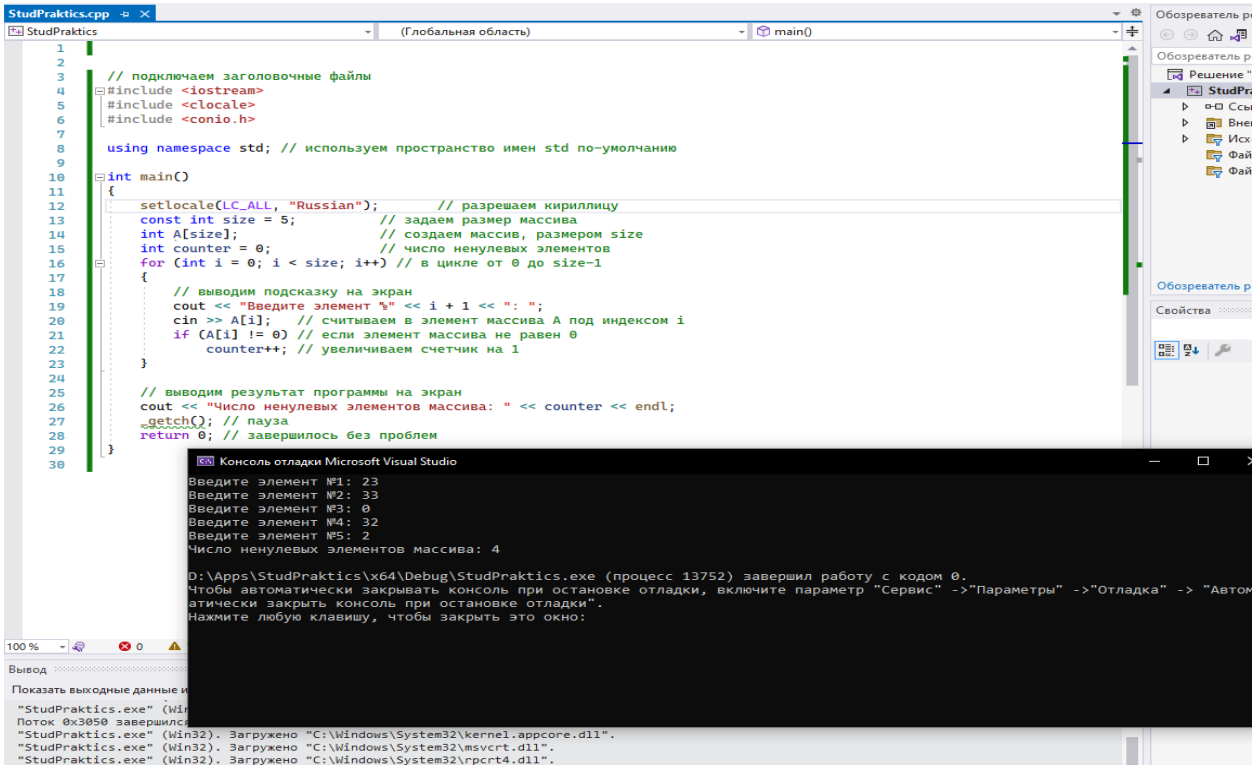
Задание-1. Создайте приложение, запрашивающее у пользователя 5 целых чисел, и для каждого числа выводящее на экран положительное оно или отрицательное.




```
#include <iostream>

using namespace std;

int main(){
   int a,b,c,d,e;
   cout<<"Введите чисчлло (a):"<<endl;
         cin>>a;
   cout<<"Введите числло (b):"<<endl;
   cin>>b;
   cout<<"Введите числло (c):"<<endl;
   cin>>c;
   cout<<"Введите числло (d):"<<endl;
   cin>>d;
   cout<<"Введите числло (e):"<<endl;
   cin>>e;
   if (a>=0){cout<<"Число (a) положительное"<<endl;}
   else{cout<<"Число (a) отрицательное"<<endl;}
   cout<<"--------------------"<<endl;
```

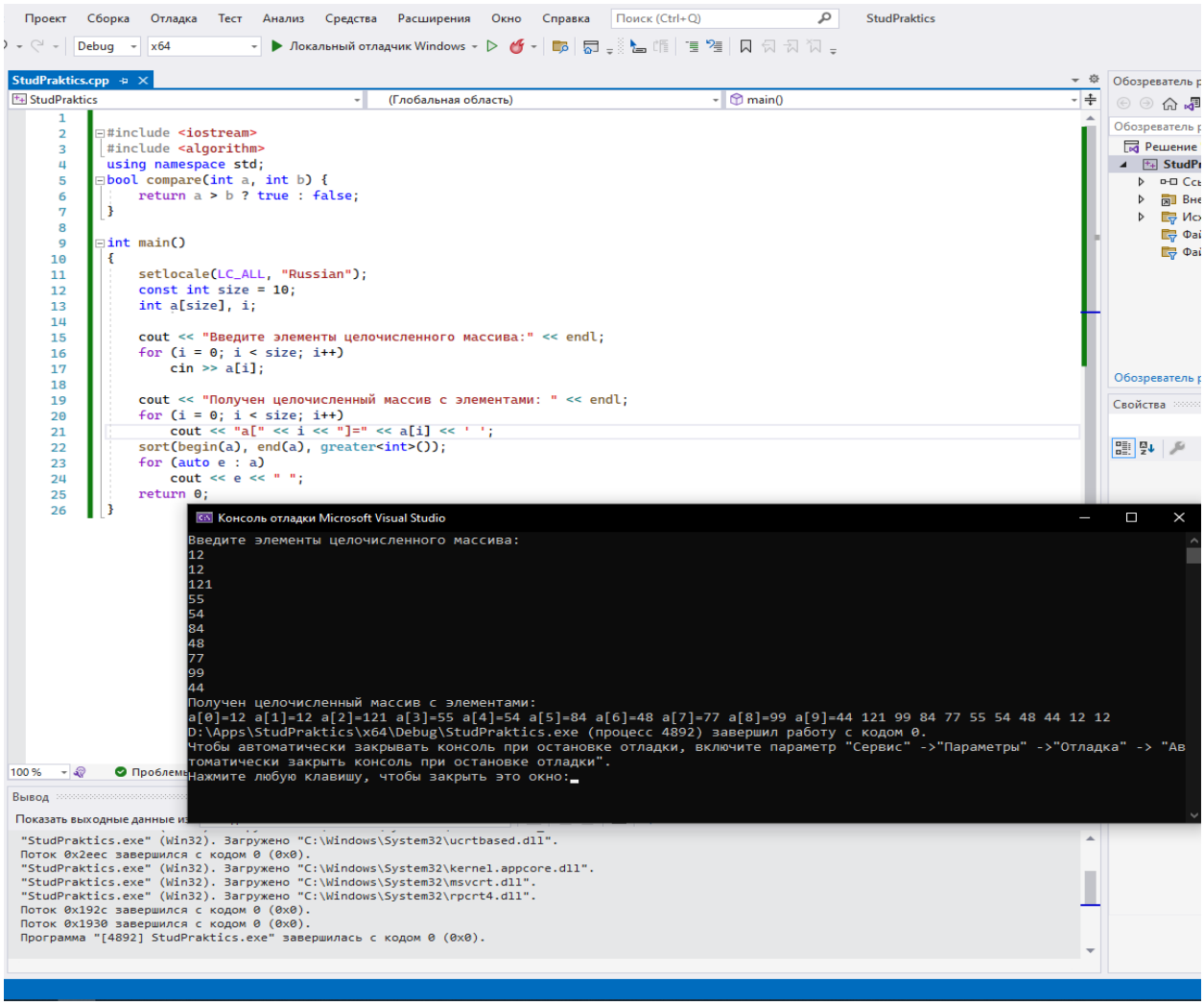
```
if (b>=0){cout<<"Число (b) положительное"<<endl;}
else{cout<<"Число (b) отрицательное"<<endl;}
cout<<"--------------------"<<endl;
if (c>=0){cout<<"Число (c) положительное"<<endl;}
else{cout<<"Число (c) отрицательное"<<endl;}
cout<<"--------------------"<<endl;
if (d>=0){cout<<"Число (d) положительное"<<endl;}
else{cout<<"Число (d) отрицательное"<<endl;}
cout<<"--------------------"<<endl;
if (e>=0){cout<<"Число (e) положительное"<<endl;}
else{cout<<"Число (e) отрицательное"<<endl;}
cout<<"--------------------"<<endl;}
```

Задание 2. Написать программу, которая вводит с клавиатуры одномерный массив из 10 целых чисел, после чего выводит количество ненулевых элементов. Перед вводом каждого элемента должна выводиться подсказка с номером элемента.




```
using namespace std;

int main()
{
   int n;
   cout<<"Введите кол-во элементов"<<endl;
   cin>>n;
   float arr[n];
   int count = 0;
   for(int i = 0; i < n; i++)
   {
      cout << "Введите " << i+1 << " элемент: ";
      cin >> arr[i];
      count += !!arr[i];
   }
   cout << "Кол-во ненулевых: " << count << endl;
   return 0;
}
```

Задание 3 Написать программу, которая сортирует по убыванию введенный с клавиатуры одномерный массив.

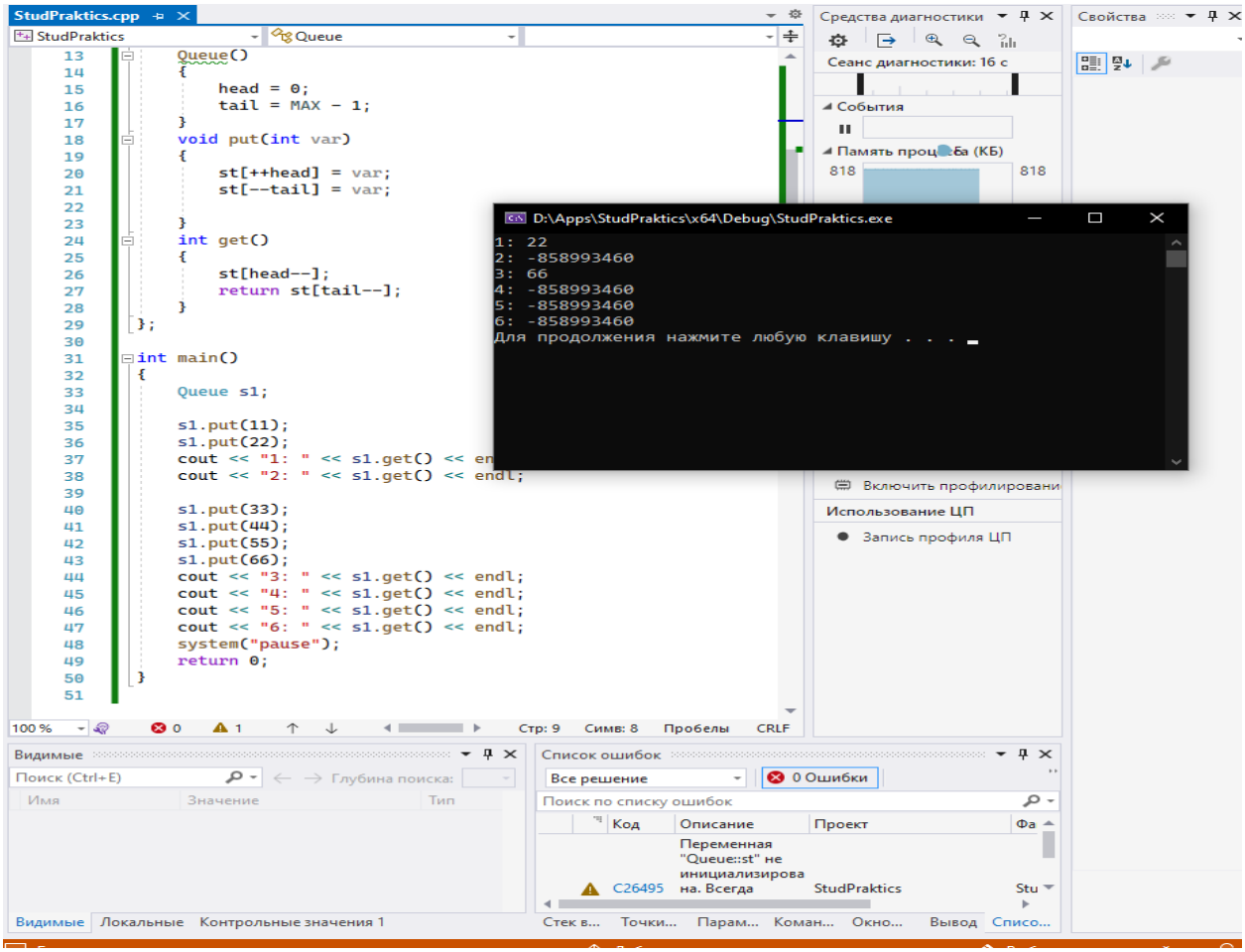



```
#include <iostream>
int comp2 (const void * a, const void * b)
{
  return ( *(int*)b - *(int*)a );
}
using namespace std;

int main(){
   int list[8]={32,76,3,21,38,2,90,28};//Сортируемые данные
   int i;
   qsort (list, 8, sizeof(int),comp2);
   for(i=0; i<8; i++)
     cout<<list[i]<<" ";
   cout<<"\n";
   cout<<"Сортировка завершена"<<endl;
```

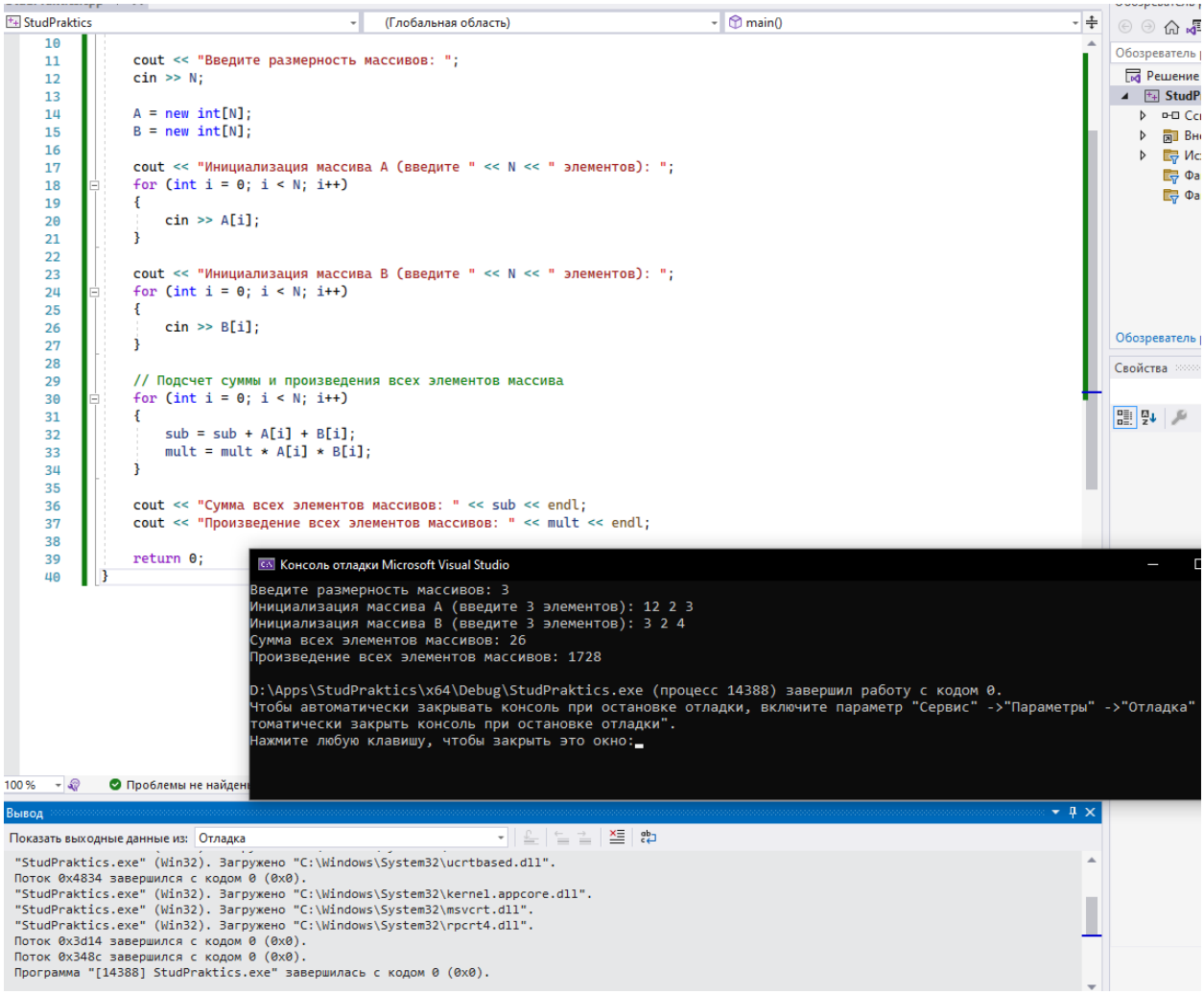
Задание 4 Создайте стек из 10 элементов на базе массива (Используйте структуру FIFO (first in – first out) «первый пришел – первый вышел». Первый введенный элемент массива должен иметь последний номер, а

последний введенный элемент начальный номер). Выведите его на экран. Найдите максимальный элемент стека.

```
#include <iostream>
using namespace std;

class Queue
{
private:
   enum { MAX = 10 };
   int st[MAX];
   int head;
   int tail;
public:
   Queue()
   {
      head = 0;
      tail = MAX-1;
   }
   void put(int var)
   {
      st[++head] = var;
      st[--tail] = var;

   }
   int get()
   {
      st [head--];
      return st[tail--];
   }
};

int main()
{
   Queue s1;

   s1.put(11);
   s1.put(22);
   cout<<"1: "<<s1.get()<<endl;
   cout << "2: " << s1.get() << endl;

   s1.put(33);
   s1.put(44);
   s1.put(55);
   s1.put(66);
   cout<<"3: "<< s1.get() << endl;
   cout << "4: " << s1.get() << endl;
   cout << "5: " << s1.get() << endl;
   cout << "6: " << s1.get() << endl;
   system ("pause");
   return 0;
}
```



Задание 5 Разработать программу, которая принимает от пользователя два одномерных произвольных целочисленных массива, а затем выводит на экран сумму и произведение всех элементов этих массивов.

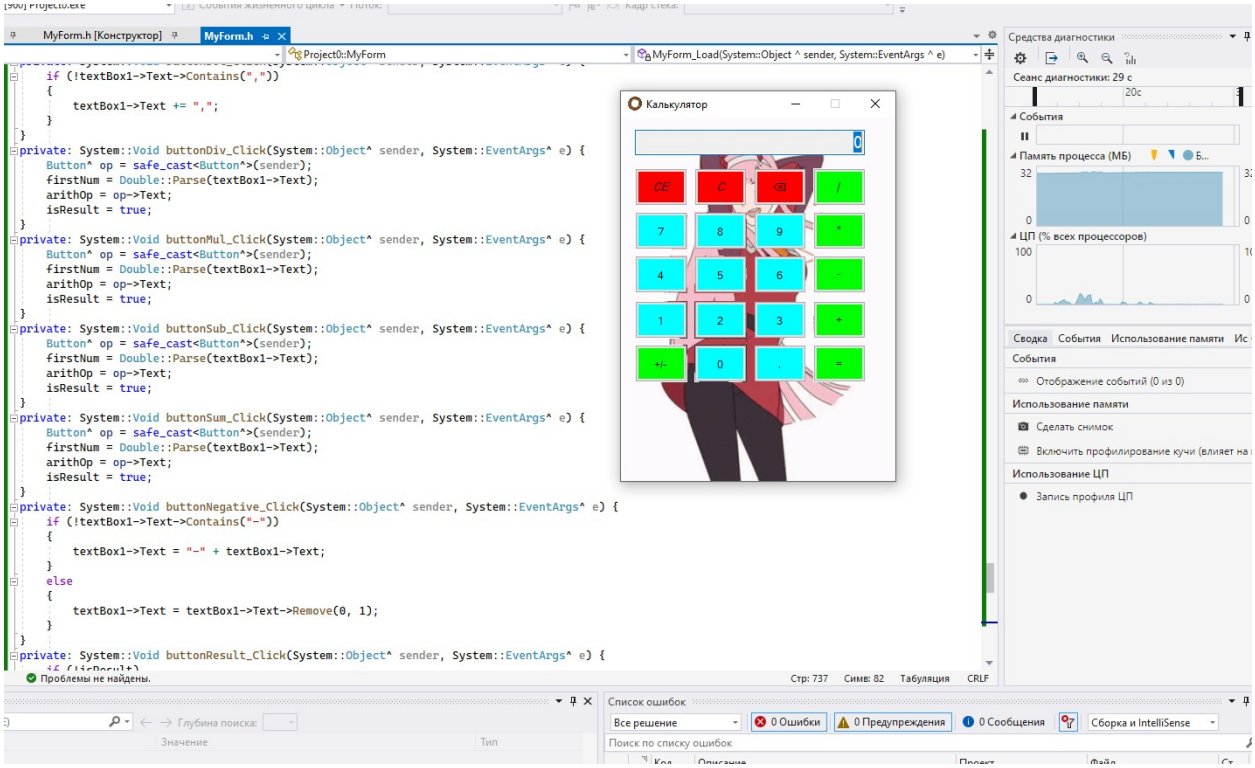



```
int main()
{
   setlocale(LC_ALL, "Russian"); // Добавление русского языка
   int N;
   int* A, * B;
   int sub = 0, mult = 1;

   cout << "Введите размерность массивов: ";
   cin >> N;

   A = new int[N];
   B = new int[N];

   cout << "Инициализация массива A (введите " << N << " элементов): ";
   for (int i = 0; i < N; i++)
   {
      cin >> A[i];
   }

   cout << "Инициализация массива B (введите " << N << " элементов): ";
   for (int i = 0; i < N; i++)
   {
      cin >> B[i];
   }
```

```
    // Подсчет суммы и произведения всех элементов массива
    for (int i = 0; i < N; i++)
    {
        sub = sub + A[i] + B[i];
        mult = mult * A[i] * B[i];
    }

    cout << "Сумма всех элементов массивов: " << sub << endl;
    cout << "Произведение всех элементов массивов: " << mult << endl;

    return 0;
}
```

3.2. Задачи по прикладному программированию.

Задача1. Разработать оконное приложение «калькулятор» в Microsoft visual studio 2022 C++



Фрагмент кода программы (без учета кода оконной формы, а также без основной части кода оконного приложения «калькулятор»)

ЗАКЛЮЧЕНИЕ

В ходе прохождения учебной практики я закрепил знания, полученные в процессе лекционных и практических занятий, а также лабораторных работ в рамках ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем». Приобрел умения и опыт в практической работе с пакетами прикладных программ.

Кроме вышеперечисленного перед началом учебной практики состоялся инструктаж по технике безопасности, правилам поведения и пожарной безопасности.

В процессе выполнения практики были выполнены следующие задачи:

1. Ознакомился сущностью и социальной значимостью будущей профессии;
2. Разобрался с разработкой программного продукта на основе готовых спецификаций на уровне модуля;
3. Мной были закреплены знания, полученные во время выполнения практических заданий, связанных с оптимизацией программного кода;
4. Набрался опыта в тестирование и в отладке программных модулей с использованием специализированных программных средств.

ВЫВОД

Итогом прохождения мной практики является следующий вывод: разрабатывать эффективные модули программного обеспечения для компьютерных систем, можно исключительно проверяя их на практике и последовательно тестируя в конкретной востребованной области.