МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВПО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра компьютерных интеллектуальных технологий проектирования

КУРСОВАЯ РАБОТА

По дисциплине: “Программирование на ЯВУ”

Тема: “Объектно-ориентированное программирование”

Разработал студент А. Кудрявцев

Подпись, дата Инициалы, фамилия

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. Н. Юров\_\_\_\_\_

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

Воронеж 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Кафедра компьютерных интеллектуальных технологий проектирования

ЗАДАНИЕ

на курсовой проект

По дисциплине: \_“Программирование на ЯВУ”

Тема: “ Объектно-ориентированное программирование”

Студент группы ИВТ- Кудрявцев Андрей

Фамилия, имя, отчество

Технические условия: среда разработки - Microsoft Visual Studio 2015, Язык С++, ОС - Windows 8 Корпоративная, Процессор – Intel Core i5-6200 2.30 GHz, RAM – 8Gb DDR3 667MHz

Содержание и объем проекта (графические работы, расчеты и прочее): работа выполнена на 40 страницах и содержит 17 рисуноков

Сроки выполнения этапов

Срок защиты курсового проекта

Руководитель А. Н. Юров

Подпись, дата Инициалы, фамилия

Задание принял студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Кудрявцев\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись, дата Инициалы, фамилия

Замечания руководителя

Содержание

1.Введение ....................................................................................................................5

2.Описание среды разработки .....................................................................................6

2.1 Создание нового проекта ............................................................................7

2.2 Средства отладки........................................................................................10

3. Общие сведения об ООП .......................................................................................13

3.1 Основные идеи объектно-ориентированного подхода...........................14

3.2 Абстракция .................................................................................................16

3.3 Инкапсуляция.............................................................................................17

3.4 Наследование .............................................................................................18

3.5 Полиморфизм .............................................................................................19

4. Синтаксис языка C++ .............................................................................................20

5. Вычислительные задачи ........................................................................................24

5.1) Задача 5.40 .................................................................................................24

5.2) Задача 7.48 .................................................................................................26

5.3) Задача 9.44 .................................................................................................30

5.4) Задача 10.32 ...............................................................................................33

6. Информационное обеспечение .............................................................................36

6.1) Аппаратные средства, подходящие для запуска проекта ......................36

6.2) Инструкция по запуску проектов.............................................................36

6.3) Программные решения ............................................................................37

7. Заключение .............................................................................................................38

8.Вывод .......................................................................................................................49

9. Список литературы ................................................................................................40

1. Введение

Объектно-ориентированный подход в последнее десятилетие стал одним из наиболее интенсивно развивающихся направлений в программировании и наиболее популярным средством разработки программного обеспечения.

Начало развитию объектно-ориентированного подхода положил язык Simula 67, который был разработан в конце 60-х гг. в Норвегии. Несмотря на то, что язык намного опередил свое время, современники (программисты 60-х гг.) оказались не готовы воспринять ценности языка Simula 67, и он не выдержал конкуренции с другими языками программирования (прежде всего, с языком Fortran).

Но достоинства языка Simula 67 были замечены некоторыми программистами, и в 70-е гг. было разработано большое число экспериментальных объектно-ориентированных языков программирования. В результате исследования этих языков были разработаны современные объектно-ориентированные языки программирования: C++, Ada, Smalltalk и др.

Наиболее распространенным объектно-ориентированным языком программирования является язык C++ . Он возник на базе соединения языков С и Simula. С++ был разработан в начале 80-х Бьерном Страуструпом, сотрудником компании AT&T. Все эти годы язык интенсивно развивался, и, наконец, в августе 1998 г. был принят международный стандарт языка С++.

2. Описание среды разработки

В качестве среды разработки мною была выбрана Microsoft Visual Studio. Это линейка продуктов компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/Интегрированная_среда_разработки) программного обеспечения и ряд других инструментальных средств. Пожалуй, самая известная среда программирования, которая обладает рядом преимуществ. В первую очередь это, конечно же, поддержка многих языков программирования. В комплект входят следующие основные компоненты:

1. Visual Basic.NET - для разработки приложений на VisualBasic;

2. Visual C++ - на традиционном языке C++;

3. Visual C# - на языке C# (Microsoft);

4. Visual F# - на F# (Microsoft Developer Division).

При необходимости имеется возможность добавления пакеты для создания проектов на других языках программирования. Это языки, ориентированные на сетевую разработку (JavaScript, Python) и язык структурированных запросов SQL.

В MVS крайне удобный редактор кода, который еще на этапе ввода проверяет синтаксис, форматирует его, вставляя необходимые отступы, и применяет цветовое кодирования для выделения разных элементов. Совокупно эти приемы делают зрительное восприятие кода простым, позволяют лучше в нем ориентироваться, что значительно облегчает процесс его написания и экономит время.

Возможности отладки. Предлагаемые в VisualStudio инструменты отладки являются наилучшим средством для отслеживания загадочных ошибок и диагностирования странного поведения. Разработчик может выполнять свой код по строке за раз, устанавливать интеллектуальные точки прерывания, при желании сохраняя их для использования в будущем, и в любое время просматривать текущую информацию из памяти.

2.1 Создание нового проекта

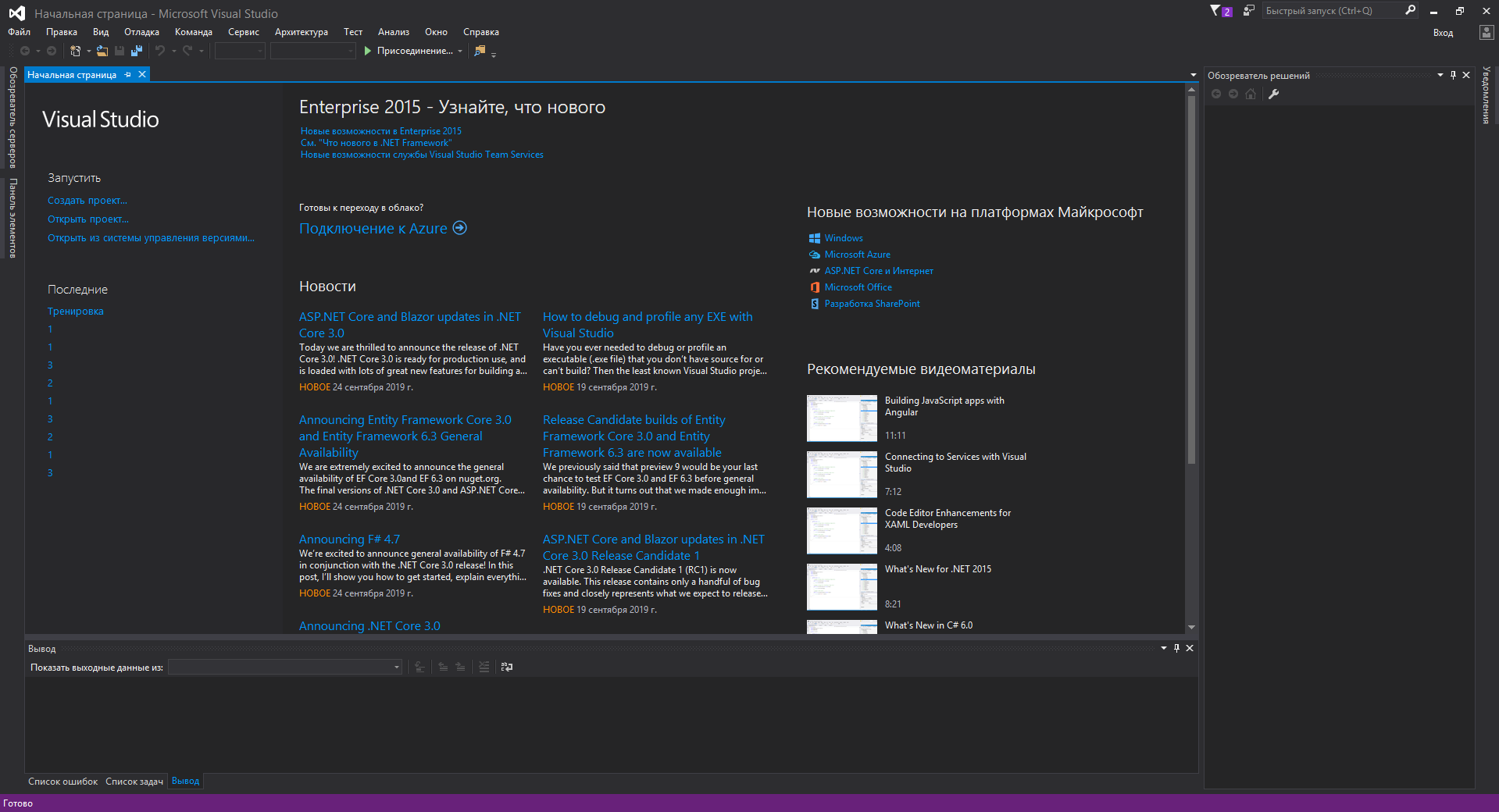


Рисунок 1. Стартовый экран VisualStudio.

Создание нового проекта осуществляется следующим образом «Файл/Создать/Проект» или просто кликнув на «Создать проект…»[рисунок 1].

В результате любого из этих действий откроется окно, в котором среди шаблонов необходимо выбрать «Visual C++/ Консольнольное приложение Win32». Затем дать имя создаваемому файлу и указать расположение.

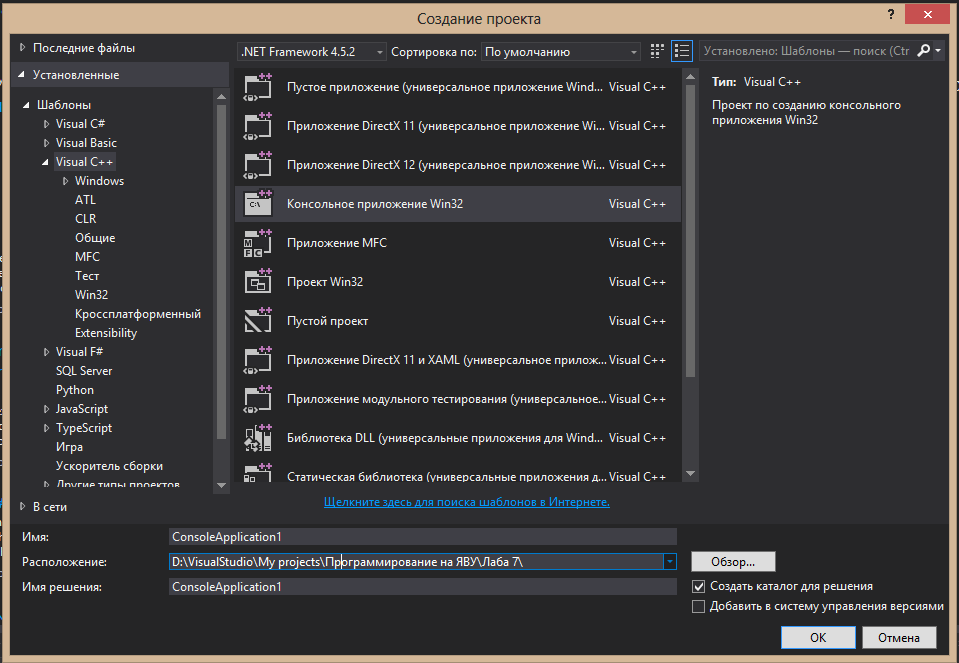


Рисунок 2. Создание нового проекта .

В новом окне нажать на «Далее», поставить галочку «Пустой проект» и снять его в «Проверки жизненного цикла разработки безопасного ПО». Нажать «Готово».

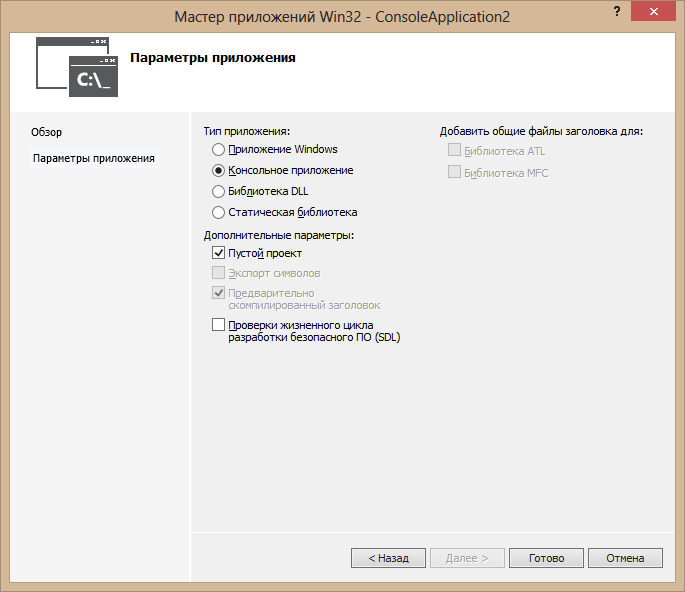


Рисунок 3. Создание нового проекта.

После этого необходимо добавить в новый проект .cpp файл, где, собственно, и будет писаться код. Для этого в обозревателе решений (по умолчанию справа) кликаем ПКМ на «Файлы ресурсов» и выбираем «Добавит/Создать элемент…».

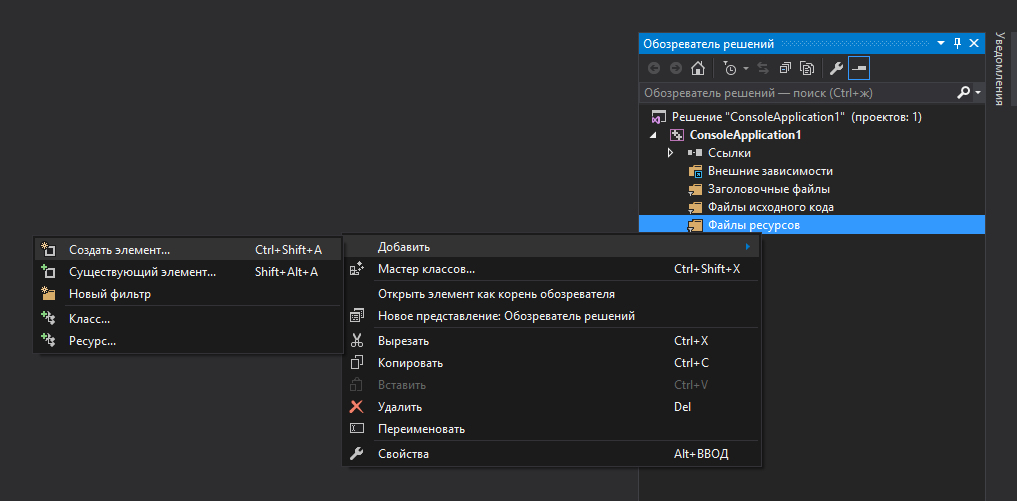


Рисунок 4. Создание .cpp файла.

В новом окне выбираем «Файл С++ (.cpp)», при необходимости меняем имя, жмем «Добавить…».

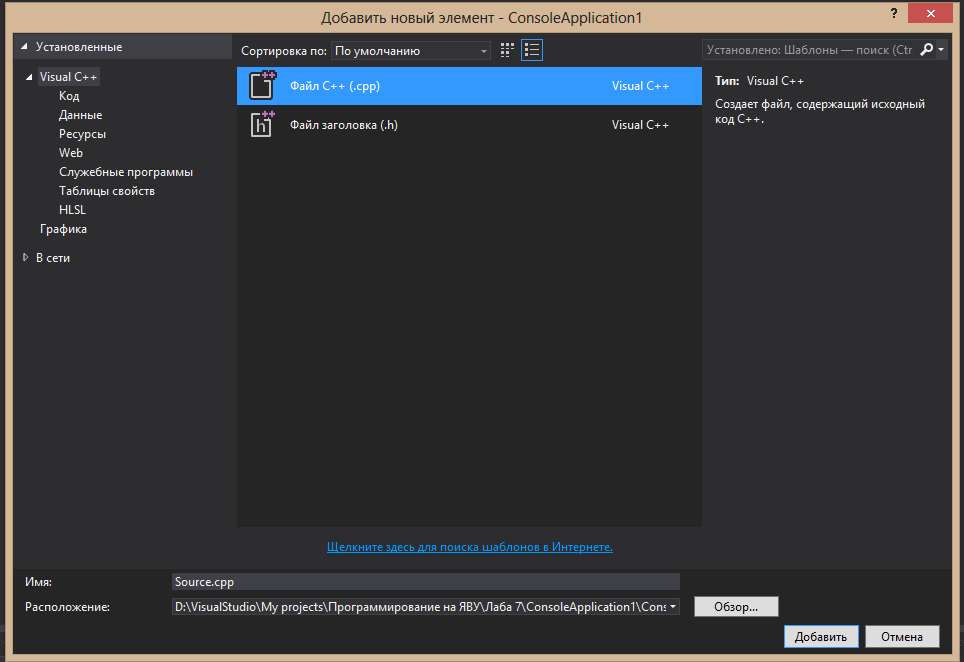


Рисунок 5. Добавление.cpp файла.

2.2 Средства отладки

Реальность такова, что никто не идеален. Независимо от уровня знаний и накопленного опыта все, так или иначе, допускают ошибки. В том числе и разработчики программного обеспечения. Независимо от обстоятельств, часто случается так, что код созданный специалистом работает совсем не так, как было задумано. Логика создаваемого и созданного проекта отличаются, и в таких случаях возникает необходимость понять, где была допущена ошибка. Для таких задач в большинстве компиляторах, и в VisualStudio в частности, существуют средства отладки.

Если говорить простым языком, то отладчик представляет собой некоторый функционал, облегчающий нахождения ошибок, позволяя последовательно и наглядно осуществить выполнение программы шаг за шагом. Для лучшего понимания обратимся к примеру. Допустим, мы имеем простую программу, которая содержит два целочисленных значения и функцию, которая меняет эти значения посредством указателей.

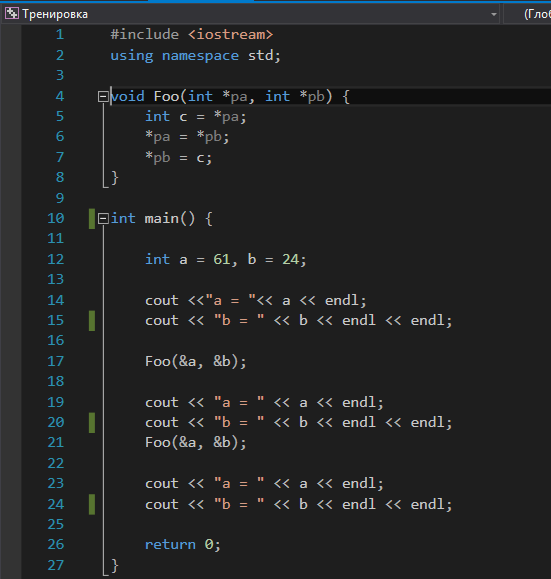


Рисунок 6. Код тренировочной программы.

Представим, что у нас возникла необходимость исследовать состояние выполняемой программы. Для этого нужно напротив интересующей нас строки кода в самой левой части рабочей области поставить точку останова, кликнув ЛКМ. Таких точек можно поставить несколько, если это требуется.

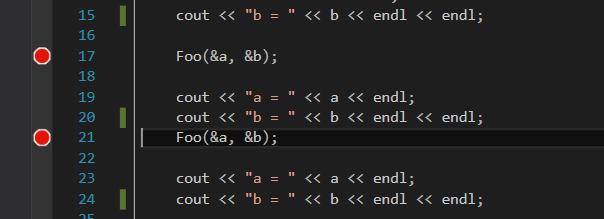


Рисунок 7. Точки останова.

Запустить работу отладчика можно несколькими способами: кликнуть на «Локальный отладчик Windows», с помощью «Отладка/Начать отладку» или воспользовавшись горячей клавишей F5. В результате таких действий компилятор начнет выполнение программы, дойдет до точки останова и сделает паузу, ожидая дальнейших команд оператора.

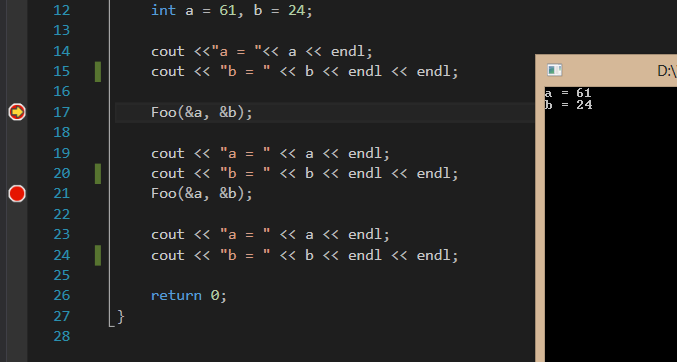


Рисунок 8. Работа отладчика.

Работа компилятора была остановлена на строке, в которой вызывается функция. Если имеется необходимость проверить ее работу необходимо воспользоваться «Отладка/Шаг с заходом» или горячей клавишей F11. Если же такой нужды нет, и мы хотим продолжить дальнейшее выполнение программы, минуя проверку функции, выбираем «Отладка/Шаг с обходом», горячая клавиша F10. Эта модель действует так же на все подобные конструкции, вроде циклов, условных операторов и т. д.

Кроме того, в VisualStudio имеется возможность вывода и закрепления на экране значения переменной, массива и т. д. для удобного отслеживания изменений. Для этого необходимо навести курсор на переменную, и на всплывшем внизу окошке кликнуть на иконку кнопки.

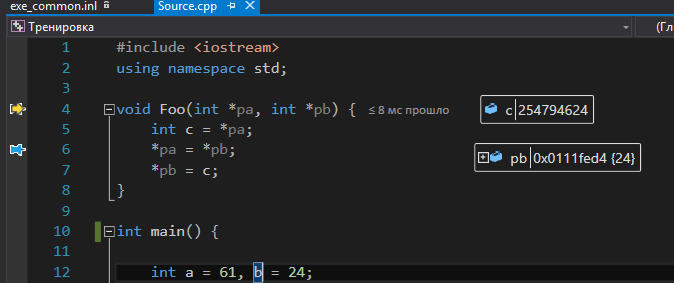


Рисунок 9. Отслеживание значения переменных.

Чтобы перейти к следующей точке остановки, можно использовать «Отладка/Перейти к следующей точке останова» или горячей клавишей F9.

Для прекращения отладки - «Отладка/остановить отладку» или комбинацией клавиш Shift+F5.

Средства отладки в VisualStudio являются основными и наиболее используемыми для проверки логики программы.

3. Общие сведения об ООП.

Объектно-ориентированный подход возник в первую очередь в ответ на растущую сложность программного обеспечения. На заре компьютерной эры возможности компьютеров были ограничены и было очень трудно написать большую программу. В 60–70-е гг. эффективность применения компьютеров резко возросла, и стало все больше создаваться прикладных программ повышенной сложности.

Наибольшее распространение в это время получило структурное проектирование по методу сверху вниз. Однако через некоторое время оказалось, что структурный подход не работает, если объем программы превышает приблизительно 100 тыс. строк. Как результат – выход проектов за рамки установленных сроков и бюджетов и, более того, их несоответствие начальным требованиям. Для решения этих проблем и стали применять объектно-ориентированный подход.

Справедливости ради отметим, что объектно-ориентированный подход является достаточно универсальным инструментом. Он может применяться и для разработки программ малой и средней сложности. Однако, именно для сложных систем использование объектно-ориентированного подхода является критичным.

3.1 Основные идеи объектно-ориентированного подхода

К основным идеям ООП относятся:

1. Программа представляет собой модель некоторого реального процесса, части реального мира; модель содержит не все признаки и свойства представляемой ею части реального мира, а только те, которые существенны для разрабатываемой программной системы;
2. Модель реального мира или его части может быть описана как совокупность взаимодействующих между собой объектов;
3. Объект описывается набором атрибутов (свойств), значения которых определяют состояние объекта, и набором операций (действий), которые может выполнять объект;
4. Взаимодействие между объектами осуществляется посылкой специальных сообщений от одного объекта к другому; сообщение, полученное объектом, может потребовать выполнения определенных действий, например, изменения состояния объекта;
5. Объекты, описанные одним и тем же набором атрибутов и способные выполнять один и тот же набор операций, представляют собой класс однотипных объектов.

С точки зрения языка программирования класс объектов можно рассматривать как тип данных, а отдельные объекты – как данные этого типа. Определение программистом собственных классов объектов должно позволить описывать конкретную задачу в терминах ее предметной области (при соответствующем выборе имен типов и имен объектов, их атрибутов и выполняемых действий).

Объектно-ориентированный подход дает следующие основные преимущества:

* Уменьшение сложности программного обеспечения;
* Повышение его надежности;
* Обеспечение возможности модификации отдельных компонент программ без изменения остальных компонент;
* Обеспечение возможности повторного использования отдельных компонент программного обеспечения.

Систематическое применение объектно-ориентированного подхода позволяет разрабатывать хорошо структурированные, надежные в эксплуатации, достаточно просто модифицируемые программные системы. Этим объясняется интерес программистов к объектно-ориентированному подходу и объектно-ориентированным языкам программирования.

3.2 Абстракция.

Люди развили чрезвычайно эффективную технологию преодоления сложности. Мы абстрагируемся от нее. Если мы не в состоянии полностью воссоздать сложный объект, то приходится игнорировать не слишком важные детали. В результате мы имеем дело с обобщенной, идеализированной моделью объекта.

Например, изучая процесс фотосинтеза у растений, мы концентрируем внимание на химических реакциях в определенных клетках листа и не обращаем внимание на остальные части – черенки, жилки и т.д.

Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов, и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.

Абстрагирование концентрирует внимание на внешних особенностях объекта и позволяет отделить самые существенные особенности поведения от несущественных. Такое разделение смысла и реализации называют барьером абстракции. Установление того или иного барьера абстракции порождает множество различных абстракций для одного и того же предмета или явления реального мира. Абстрагируясь в большей или меньшей степени от различных аспектов проявления реальности, мы находимся на разных уровнях абстракции.

3.3 Инкапсуляция

Инкапсуляция ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) encapsulation, от [лат.](https://ru.wikipedia.org/wiki/Латинский_язык) in capsula) — в [информатике](https://ru.wikipedia.org/wiki/Информатика) размещение в одном компоненте данных и методов, которые с ними работают. Также может означать скрытие внутренней реализации от других компонентов. Например, доступ к скрытой переменной может предоставляться не напрямую, а с помощью методов для чтения ([геттер](https://ru.wikipedia.org/wiki/Геттер_(программирование))) и изменения ([сеттер](https://ru.wikipedia.org/wiki/Setter)) её значения.

В ООП инкапсуляция тесно связана с принципом [абстракции данных](https://ru.wikipedia.org/wiki/Абстракция_данных) (не путать с [абстрактными типами данных](https://ru.wikipedia.org/wiki/Абстрактный_тип_данных), реализации которых предоставляют возможность инкапсуляции, но имеют иную природу). Это, в частности, приводит к другому распространённому заблуждению — рассмотрению инкапсуляции неотрывно от [сокрытия](https://ru.wikipedia.org/wiki/Сокрытие_(программирование)).

В Си++ доступно несколько спецификаторов, и они изменяют доступ к данным следующим образом:

* Публичные (public) данные — доступны всем;
* Защищенные (protected) — доступны только классу и дочерним классам;
* Приватные (private) —доступны только классу которому они принадлежат.

3.4 Наследование

Наследование — один из четырёх важнейших механизмов объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

Другими словами, класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами класса-наследника точно так же, как с объектами базового класса.

Простое наследование:

Класс, от которого произошло наследование, называется базовым или родительским (англ. base class). Классы, которые произошли от базового, называются потомками, наследниками или производными классами (англ. derived class). В некоторых языках используются абстрактные классы.

Абстрактный класс — это класс, содержащий хотя бы один абстрактный метод, он описан в программе, имеет поля, методы и не может использоваться для непосредственного создания объекта. То есть от абстрактного класса можно только наследовать. Объекты создаются только на основе производных классов, наследованных от абстрактного.

Например, абстрактным классом может быть базовый класс «сотрудник вуза», от которого наследуются классы «аспирант», «профессор» и т. д. Так как производные классы имеют общие поля и функции (например, поле «год рождения»), то эти члены класса могут быть описаны в базовом классе. В программе создаются объекты на основе классов «аспирант», «профессор», но нет смысла создавать объект на основе класса «сотрудник вуза».

3.5 Полиморфизм

Полиморфизм — возможность объектов с одинаковой спецификацией иметь различную реализацию.

Язык программирования поддерживает полиморфизм, если классы с одинаковой спецификацией могут иметь различную реализацию — например, реализация класса может быть изменена в процессе наследования.

Кратко смысл полиморфизма можно выразить фразой: «Один интерфейс, множество реализаций».

Полиморфизм — один из четырёх важнейших механизмов объектно-ориентированного программирования (наряду с абстракцией, инкапсуляцией и наследованием).

Полиморфизм позволяет писать более абстрактные программы и повысить коэффициент повторного использования кода. Общие свойства объектов объединяются в систему, которую могут называть по-разному — интерфейс, класс. Общность имеет внешнее и внутреннее выражение:

1) Внешняя общность проявляется как одинаковый набор методов с одинаковыми именами и сигнатурами (именем методов и типами аргументов и их количеством);

2) Внутренняя общность — одинаковая функциональность методов. Её можно описать интуитивно или выразить в виде строгих законов, правил, которым должны подчиняться методы. Возможность приписывать разную функциональность одному методу (функции, операции) называется перегрузкой метода (перегрузкой функций, перегрузкой операций).

4. Синтаксис языка C++

В данном разделе будет рассмотрен синтаксис языка C++, использованный лично мною для решения вычислительных задач соответствующего варианта. Сделано это будет следующим образом. Сначала приводится конструкция, затем следует ее описание или выполняемая функция.

**Директивы**

#include - предлагает компилятору включить другой исходный файл, имя которого ука­зывается после директивы. Имя файла заключается в двойные кавычки или в < >.

using - позволяет использовать все имена в пространстве имен без имени пространства имен в качестве явного квалификатора.

**Пространства имен**

namespace std – пространство имен std. декларативная область, в рамках которой определяются различные. Пространства имен используются с целью избежания конфликтов имен, которые могут возникнуть, особенно в таких случаях, когда база кода включает несколько библиотек.

**Библиотеки**

<iostream> - заголовочный файл стандартной библиотеки, содержащей операторы ввода/вывода.

<time.h> заголовочный файл стандартной библиотеки, содержащий типы и функции для работы с датой и временем.

<Windows.h> - это специфичный для Windows заголовочный файл для языков программирования C и [C++](https://en.wikipedia.org/wiki/C%2B%2B), который содержит объявления для всех функций в API [Windows](https://en.wikipedia.org/wiki/Windows_API), всех общих макросов, используемых программистами Windows, и всех типов данных, используемых различными функциями и подсистемами.

**Функции и параметры**

main() – глобальная функция, являющаяся точкой входа в программу.

setlocale() - позволяет настраивать локальные, например, российские, параметры в программе. Используется для локализации текста. Первый параметр отвечает за определение области действия функции, второй представляет собой название локали.

setlocale(LC\_ALL, «ru») – комбинация значений,  говорит о том, что применяется настройка для всей программы (т.к. используется LC\_ALL) в виде установки русского языка (т.к. используется “russian”).

rand() – генерирует числа в диапазоне от 0 до RAND\_MAX. RAND\_MAX – это константа определенная в библиотеке <cstdlib>.

srand() - устанавливает передаваемое пользователем значение в качестве стартового. функцию следует вызывать только один раз: в начале программы (обычно в верхней части функции main()).

time() - возвращает текущее календарное значение времени в секундах.

endl – вставляет символ перехода на новую строку и очищает буфер.

SetConsoleCP(1251) - задаем кодировку для ввода символов с клавиатуры в консоль.

SetConsoleOutputCP(1251) - задаем кодировку для вывода символов на экран

gets\_s() - считывает строку из стандартного входного потока  и сохраняет ее в буфер.

Типы данных:

int - представляет целое число. Занимает 4 байта (32 бита). Диапазон предельных значений от −2 147 483 648 до 2 147 483 647.

double - представляет вещественное число двойной точности с плавающей точкой в диапазоне +/- 1.7E-308 до 1.7E+308. В памяти занимает 8 байт (64 бита)

char – символьная переменная, занимает 1 байт. Однако, вместо конвертации значения типа char в целое число, оно интерпретируется как символ ASCII.

const int – целочисленная константа, особенность которой заключается в том, что ее значение не может быть изменено по ходу программы.

unsigned int - представляет положительное целое число. Занимает 4 байта (32 бита), и может принимать значения от 0 до 4 294 967 295.

Операторы:

= - оператор присваивания значения.

% - оператор, позволяющий вычислить остаток от деления.

>= - оператор сравнения (больше или равно).

+ - математический оператор (сложение).

i++ - математический оператор, инкремент в постфиксной форме, увеличивает значение переменной i на 1.

cout – оператор вывода.

cin – оператор ввода.

for – параметрический цикл (с заданным числом повторений).

Общая форма записи

for (Инициализация; Условие; Модификация){  
   БлокОпераций;  
}

a[b] – обращение к элементу массива.

if else - условный оператор, конструкция ветвления.

Общая форма записи

If (условие) {  
   БлокОпераций1;  
}

else {  
   БлокОпераций2;  
}

while – цикл, используемый в тех случаях, когда неизвестно, сколько итераций должно произойти.

while (Условие) {

Тело цикла;

}

return – оператор возврата значения.

**Вычислительные задачи.**

**Задача 5.40**

Условие: Даны натуральное число n и вещественные a1,a2,…,an.Определить сумму всех вещественных чисел.

Решение:

Словесное описание алгоритма:

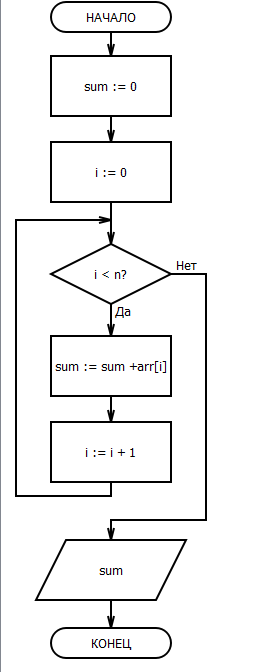
1.Инициализируем переменную n и статический массив вещественных чисел размером n.

2. Инициализируем переменную sum нулем, она нужна для хранения суммы.

3. В цикле от 0 до n (не включая n) прибавляем к sum каждый элемент массива.

4. Выводим результат на экран.

Блок-схема алгоритма:

(Рис. 10)

Программная реализация алгоритма на языке C++:

#include<iostream>

using namespace std;

int main()

{

const int n = 4;

float arr[] = {1.43f,3.234f,534.23f,99.01f};

float sum = 0;

for(int i=0;i<n;i++)

{

sum += arr[i];

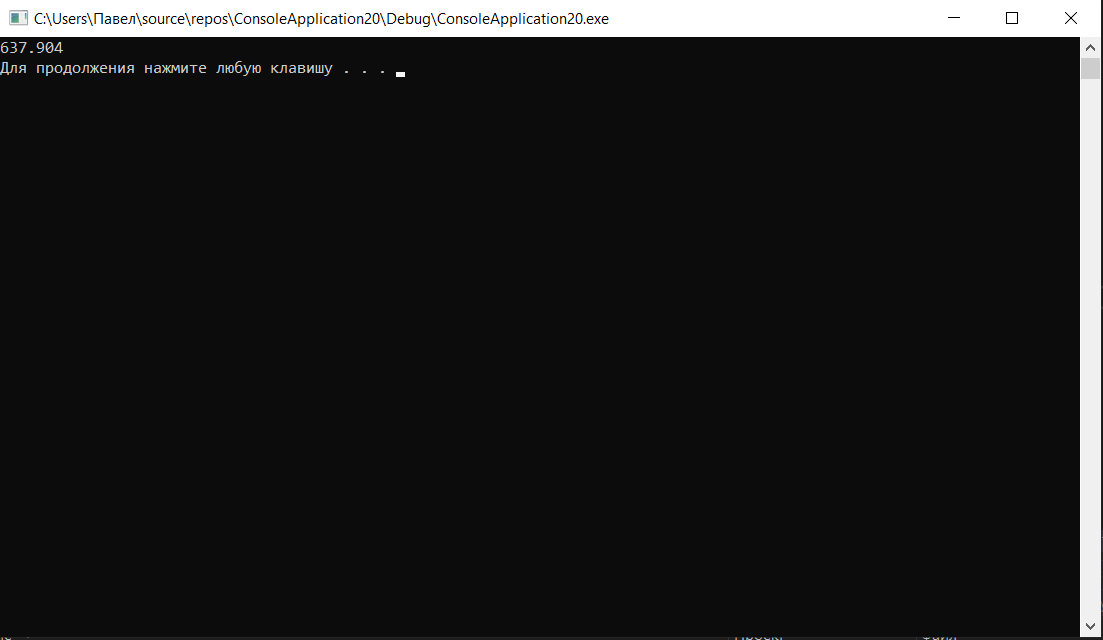
}

cout<<sum;

return 0;

}

Листинг программы:

(Рис 11)

**Задача 7.48**

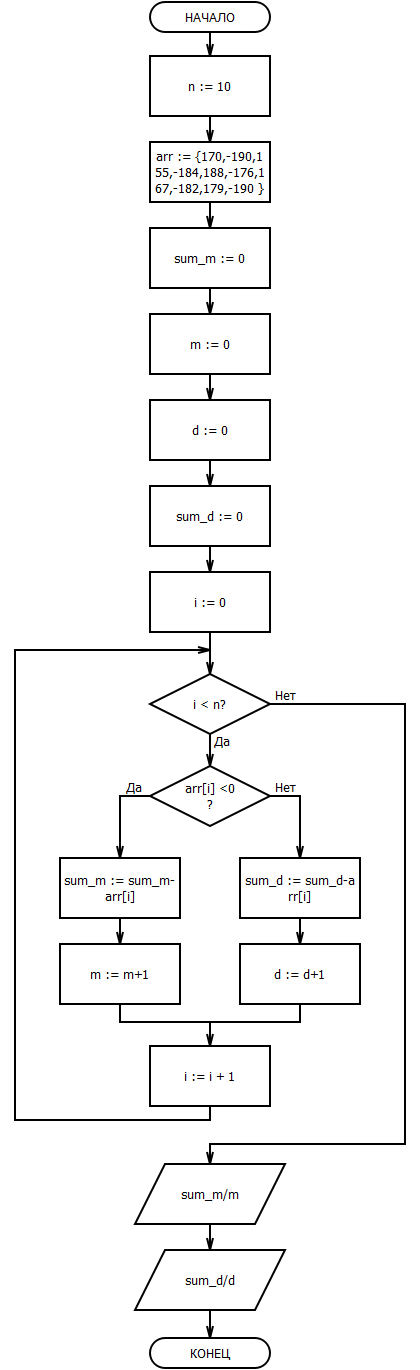
Условие: Известен рост каждого ученика класса. Рост мальчиков условно задан отрицательными числами. Определить средний рост мальчиков и средний рост девочек.

Решение:

Словесное описание алгоритма:

1. Инициализируем статический целочисленный массив, который хранит рост всех учеников и переменную n, значение которой равно числу учеников.
2. Инициализируем нулями переменные, которые будут хранить сумму роста мальчиков, их количество, сумму роста девочек, их количество.
3. В цикле от 0 до n, проверяем, если рост меньше нуля, то умножаем его на -1 и суммируем к росту мальчиков и к их количеству добавляем единицу.
4. Если рост больше 0, то добавляем это значение к росту девочек, а их количество увеличиваем на единицу.
5. Выводим на экран средний рост мальчиков (отношение суммы роста к количеству мальчиков).
6. Выводим средний рост девочек (по аналогии с мальчиками).

Блок-схема алгоритма:

(Рис. 12)

Программная реализация алгоритма:

#include<iostream>

using namespace std;

int main()

{

const int n = 10;

int arr[10] = { 170,-190,155,-184,188,-176,167,-182,179,-190 };

double sum\_m = 0;

int m = 0;

double sum\_d = 0;

int d = 0;

for (int i = 0; i < n; i++)

if (arr[i] < 0)

{

sum\_m += (-1) \* arr[i];

m++;

}

else

{

sum\_d += arr[i];

d++;

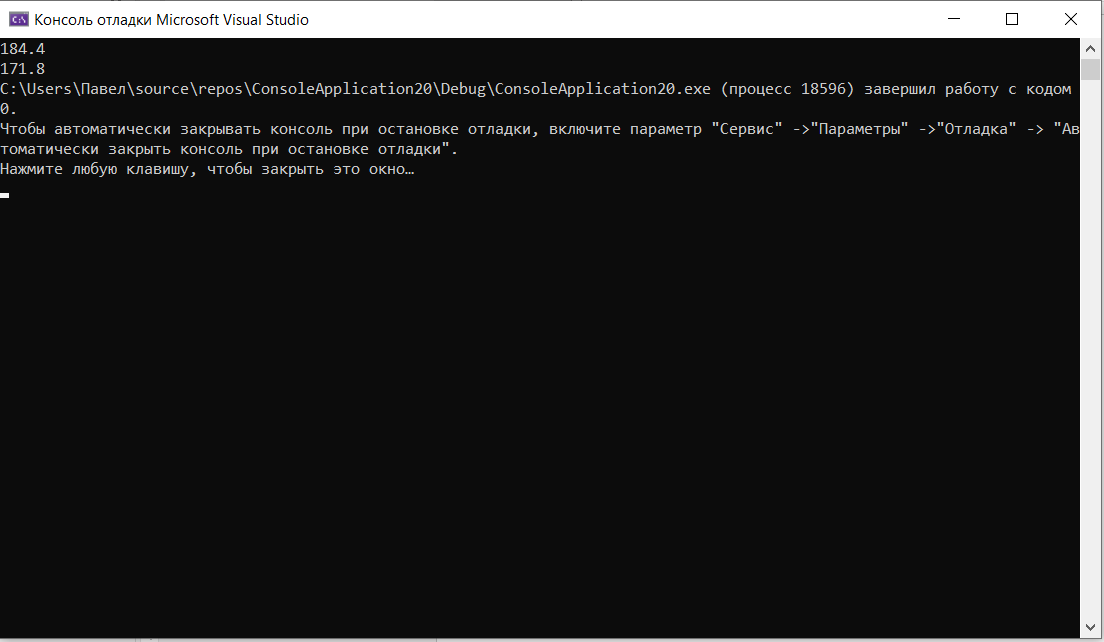
}

cout << sum\_m / m << endl << sum\_d / d;

return 0;

}

Листинг программы:

(Рис. 13)

**Задача 9.44**

Условие: Дано слово s. Получить слово t, получаемое путем прочтения слова s начиная с его конца.

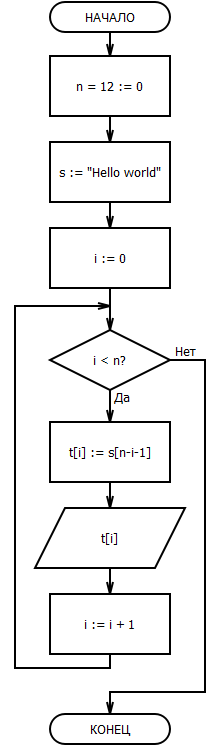
Решение:

Словестное описание алгоритма:

1. Инициализируем массив символов, из которых состоит слово и переменную, значение которой равно длине слова.
2. Объявляем массив символов для хранения нового слова размером исходного массива.
3. В цикле, пробегаясь по всем элементам исходного массива, записываем их в новый с конца.
4. Выводим новое слово на экран.

Блок-схема алгоритма:

(Рис. 14)



Программная реализация алгоритма:

#include<iostream>

using namespace std;

int main()

{

const int n = 12;

char s[n] = "Hello world";

char t[n];

for (int i = 0; i < n; i++)

{

t[i] = s[n - i - 1];

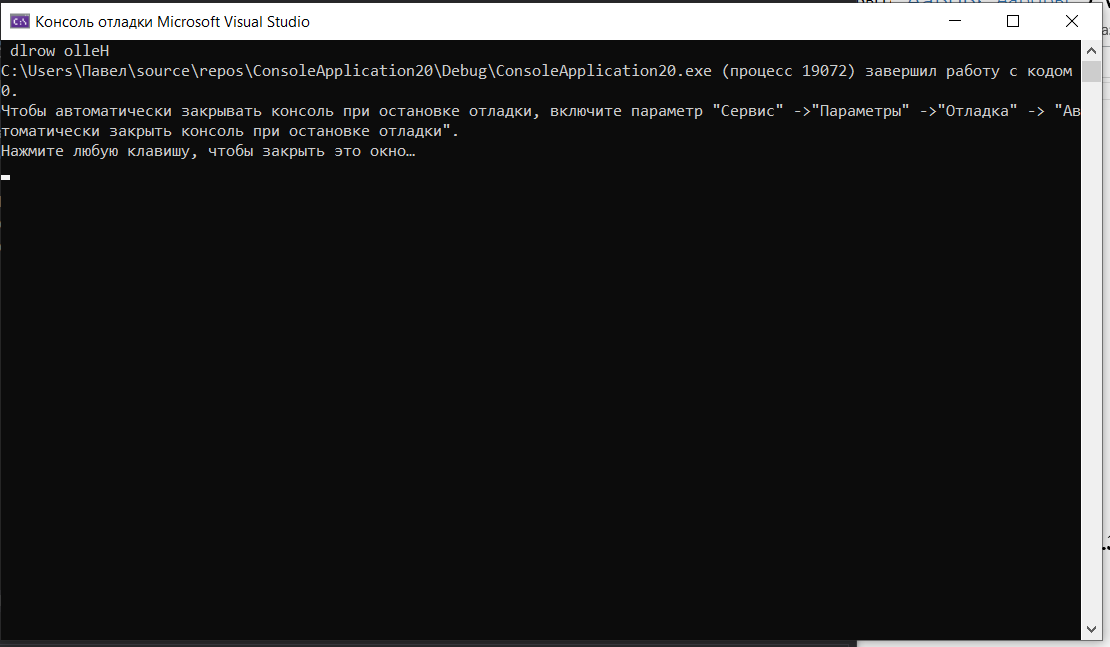
cout << t[i];

}

return 0;

}

Листинг программы:

(Рис. 15)

**Задача 10.32**

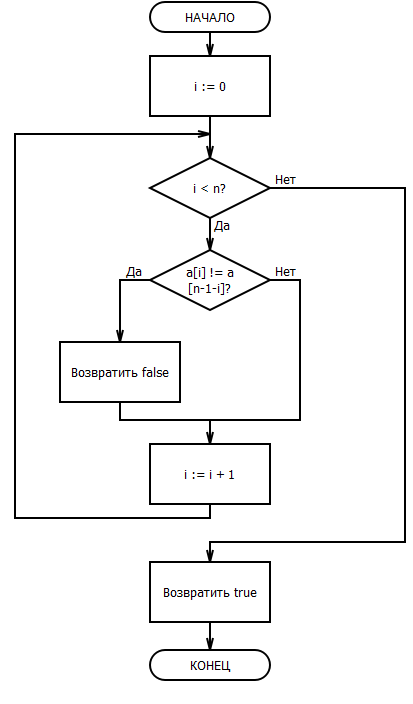
Условие: Даны три слова. Выяснить, является ли хоть одно из них палиндромом ("перевертышем"), т. е. таким, которое читается одинаково слева направо и справа налево. (Определить функцию, позволяющую распознавать слова палиндромы.)

Решение:

Словесное описание алгоритма:

1. Определим функцию, принимающую в качестве аргументов константный указатель на массив символов и его размер, и возвращающую результат сравнения.
2. Очевидно, для того чтобы слово являлось палиндромом, при проходе массива в обе стороны соответствующие символы будут равны.
3. При первом нарушении равенства прерываем функцию и возвращаем false;
4. Если условие пройдено успешно всеми элементами массива, то возвращается true;

Блок-схема алгоритма:

(Рис. 16)

Программная реализация алгоритма:

#include<iostream>

using namespace std;

bool Palindrome(const char\* a,int n)

{

for (int i = 0; i < n/2; i++) if (a[i] != a[n - 1 - i]) return false;

return true;

}

int main()

{

char a[4] = "LoL";

char b[9] = "Darkness";

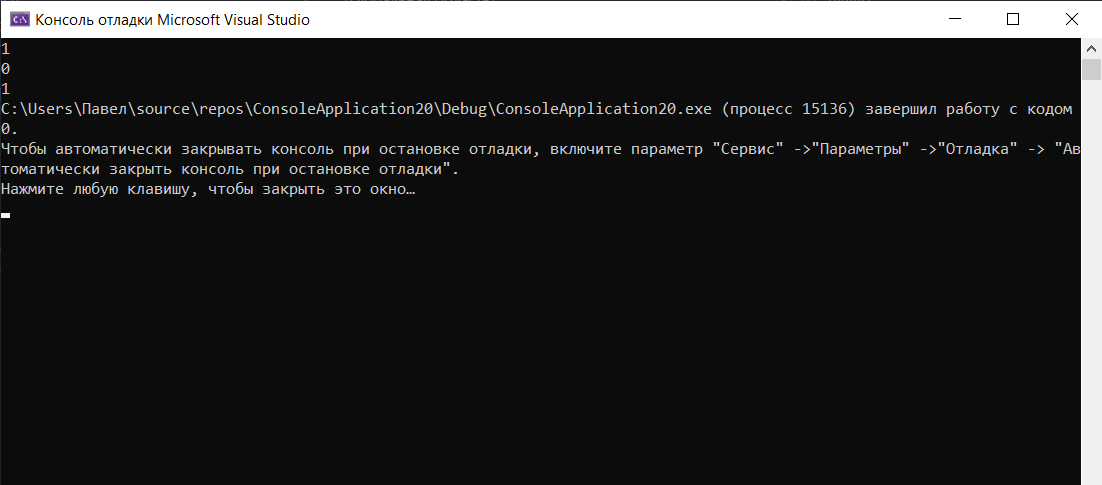
char c[12] = "aaaabbbaaaa";

cout << Palindrome(a, 3) << endl << Palindrome(b, 8) << endl << Palindrome(c, 11);

return 0;

}

Листинг программы:

(Рис. 17)

6. Информационное обеспечение

6.1 Аппаратные средства, подходящие для запуска проекта

* ОС: Windows Server 2008 R2 с пакетом обновления 1 (SP1), Windows Server 2012, Windows Server 2012 R2, Windows 7 с пакетом обновления 1 (SP1), Windows 8, Windows 8.1, Windows 10.
* Процессор с тактовой частотой 1,6 ГГц или большей.
* 1 ГБ ОЗУ (1,5 ГБ при выполнении в виртуальной машине).
* 4 ГБ доступного пространства на жестком диске.
* Жесткий диск 5400 об/мин.
* Видеоадаптер с поддержкой DirectX 9 и разрешением экрана 1024x768 или выше.

6.2 Инструкция по запуску проектов

Одним из возможных и самым привычным способом запуска проекта является его запуск с помощью проводника. Переходим в папку с адресом «D:\VisualStudio\My projects\Программирование на ЯВУ\Курсовая». Здесь находятся все решения задач курсовой работы. Запуск всех проектов аналогичен, поэтому рассмотрим его на примере задачи 5.52. Заходим в соответствующий каталог, далее в папку «Debag». В ней находится файл с именем «Задача 5.52.exe». Запуск осуществляется двойным кликом.

6.3 Программные решения

Значительная часть работы при написании курсовой – это работа с текстом и его редактирование. Microsoft Word – один из самых популярных текстовых процессоров, предназначенный для создания, просмотра и редактирования [текстовых документов](https://ru.wikipedia.org/wiki/Текстовый_файл), с локальным применением простейших форм [таблично](https://ru.wikipedia.org/wiki/Таблица) - [матричных](https://ru.wikipedia.org/wiki/Матрица_(математика)) алгоритмов. Выпускается [корпорацией Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в составе [пакета](https://ru.wikipedia.org/wiki/Офисный_пакет) [Microsoft Office](https://ru.wikipedia.org/wiki/Microsoft_Office). Microsoft Word крайне прост в освоении, имеет приятный и интуитивно понятный интерфейс, а так же более чем достаточным функционалом для выполнения своей задачи.

В качестве среды разработки мною была использована Microsoft Visual Studio версии 2015 года. В этом пункте просто упомянем ее: останавливаться подробно на ее описании нет необходимости, так как это было сделано выше в соответствующем разделе.

Для создания блок - схем я использовал онлайн – конструктор «Flowchart Maker & Online Diagram Software». Очень удобный софт, позволяющий создавать не только блок – схемы, но и другие графические элементы и конструкции для широкого списка учебных дисциплин. Вся работа осуществляется в браузере, после чего имеется возможность скачать файл в формате drawio для последующего использования или открытия в том же конструкторе и корректирования. Кроме этого возможно сохранить в файл в стандартных для графических изображений форматах png и jpg.

Для создания скриншотов мною было использовано приложение «Ножницы», включенное в Windows. Оно представляет собой инструмент захвата экрана и позволяет снимок произвольной области с последующим его сохранением в виде изображения.

1. **Заключение**

Курсовая работа традиционно состоит из двух частей. Первая часть теоретическая. Ее цель – ознакомление студента с необходимым теоретическим материалом и его последующее изложение. В частности, в данном курсовом проекте мною была рассмотрена вся информация, которая касается программного обеспечения и которой я должен владеть на данном этапе обучения: что такое ПО, какие существуют концепции разработки ПО, каких видов оно бывает и где используется.

Кроме того, мною было приведено краткое описание среды разработки, используемой для решения своих задач. Сюда входят общие сведения, описание сильных, рассмотрение основных действий: создание проекта, сборка проекта, работа с отладчиком и т. д.

Я ознакомился с таким понятием как объектно-ориентированное программирование, с его концепциями и их разновидностями. Особое внимание было уделено рисованию блок - схем.

Отдельно следует выделить третий раздел, который содержит в себе синтаксис языка C++, использованный мною и подробное его описание.

Вторая часть – практическая - и она представляет собой непосредственное решение поставленных задач и корректное его оформление, которое я сделал в соответствии с планом: условие, наглядное представление алгоритма решения с помощью блок – схем, листинг программы с подробным описанием и скриншот, отображающий результат работы программы.

1. **Вывод**

Данная курсовая работа – это отличный способ изучения учебной дисциплины. В ходе ее написания мною были обретены как новые полезные навыки, так и отработаны старые.

В частности, я получил подробное представление о процессе программирования и создании проектов, обрёл необходимый опыт в такой среде разработки как Microsoft Visual Studio. Необходимость составить блок – схему, отображающую принцип решения задачи стало поводом для повторения школьной программы и тренировки данного навыка. Мною были освоены и отработаны базовые навыки программирования на языке C++.

Результатом выполненной курсовой работы стало обретение мною необходимого теоретического материала и опыта разработки конкретных задач, что в совокупности является хорошей крепкой базой для дальнейшего обучения.

1. Список литературы.

Инкапсуляция – Википедия. Свободная энциклопедия [Электронный ресурс]/URL: <https://ru.wikipedia.org/wiki/Инкапсуляция_(программирование)>

2.C++ для начинающих - proginfo.ru [Электронный ресурс]/URL: <https://proginfo.ru/>

3. CyberForum.ru [Электронный ресурс]/URL: <http://www.cyberforum.ru/>

4.Fandom [Электронный ресурс]/URL: <https://gos.fandom.com/wiki/Основные_принципы_ООП>

5. SimpleCode [Электронный ресурс]/URL: <https://www.youtube.com/channel/UCtLKO1Cb2GVNrbU7Fi0pM0w>