

Министерство науки и высшего образования Российской Федерации
Казанский национальный исследовательский технический университет –
КАИ им. А.Н. Туполева

Институт компьютерных технологий и защиты информации
Отделение СПО ИКТЗИ «Колледж информационных технологий»

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам

Казань 2021

Составитель преподаватель СПО ИКТЗИ Мингалиев Заид Зульфатович

Методические указания к лабораторным работам по дисциплине «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЕ» предназначены для студентов направления подготовки 09.02.07 «Информационные системы и программирование»

ОГЛАВЛЕНИЕ

ПРОЦЕСС СДАЧИ ВЫПОЛНЕННОЙ РАБОТЫ	4
ЛАБОРАТОРНАЯ РАБОТА №1.	5
Знакомство с интегрированной средой разработки. Создание простейшего консольного приложения.	

ПРОЦЕСС СДАЧИ ВЫПОЛНЕННОЙ РАБОТЫ

По итогам выполнения работы студент:

1. демонстрирует преподавателю правильно работающие программы;
2. демонстрирует приобретённые знания и навыки отвечает на пару небольших вопросов преподавателя по составленной программе, возможностям её доработки;
3. демонстрирует отчет по выполненной лабораторной работе.

Итоговая оценка складывается из оценок по трем указанным составляющим.

Шаблон оформления отчета представлен в приложении 1. Требования к формированию отчета представлены в приложении 2.

ЛАБОРАТОРНАЯ РАБОТА №1.

Знакомство с интегрированной средой разработки. Создание простейшего консольного приложения.

ЦЕЛЬ РАБОТЫ

Приобрести умения и практические навыки для работы с интегрированной средой разработки Visual Studio IDE и управления потоками ввода/вывода при составлении консольных программ.

ХОД РАБОТЫ

1) Знакомство с системой программирования Microsoft Visual Studio .NET.

Microsoft Visual Studio – это интегрированная среда разработка (Integrated Development Environment (IDE)) для создания, документирования, запуска и отладки программ, написанных на языках платформы .NET Framework (C#, C++, Visual Basic и другие).

Среда разработки Visual Studio включает средства управления проектами, редактор исходного текста, конструкторы пользовательского интерфейса, компиляторы, компоновщики, инструменты, документацию и отладчики. Она позволяет создавать кроссплатформенные приложения для 32-, 64- и 86-разрядных Windows-платформ. Одно из важнейших усовершенствований – возможность работы с разными языками программирования в единой среде разработки.

Скачать установщик среды разработки можно скачать на официальном сайте Microsoft <https://visualstudio.microsoft.com/ru/vs/>.

.NET Framework – это кроссплатформенная среда выполнения приложений.

.NET Framework состоит из двух частей. Первая часть включает в себя набор заранее написанного кода (официально именуемого SDK, Dev Packs или «Пакеты разработчика»). Вторая часть включает в себя программу, которая может интерпретировать код .NET Framework в команды для операционной системы. Эта часть, которую называют «средой выполнения», позволяет запускать программы, написанные с использованием .NET Framework.

Если бы не .NET, пользователям пришлось бы устанавливать среду исполнения для программ на каждом языке. То есть чтобы запустить приложение на Visual Basic, нужно скачать среду выполнения для Visual Basic. Если же программа написана на C#, то придётся скачивать среду и для неё.

Для программистов это важно, потому что даёт возможность развивать одну среду, которая используется сразу для четырёх языков. Иначе обычным разработчикам приходилось бы ждать, пока выйдет новая версия библиотек для их языка. Менее популярные языки, вроде F#, получали бы обновление намного позже C#.

Принцип работы достаточно простой, хотя выглядит запутанным. В основном — из-за схожих названий: CLR, CLI и CIL. Для начала посмотрите на Рисунок 1.1:

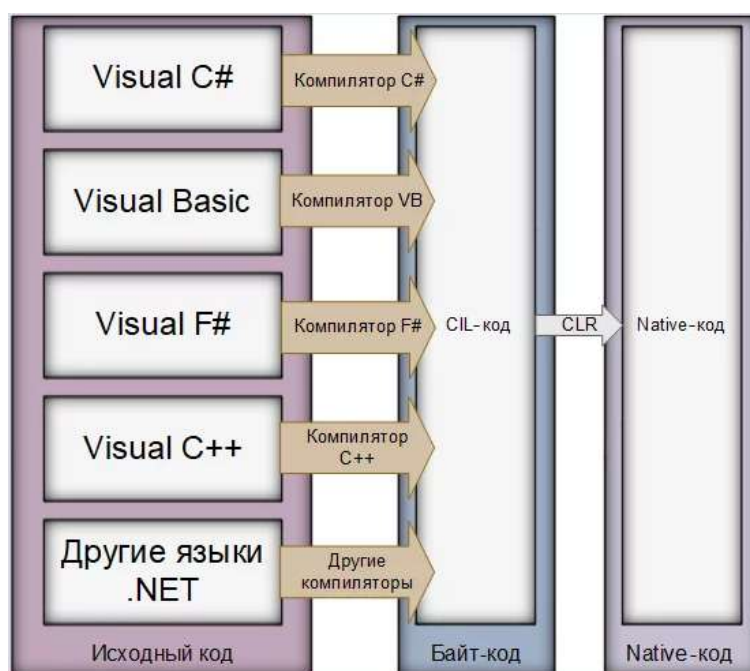


Рисунок 1.1 – Общезыковая инфраструктура

Это CLI (англ. Common Language Infrastructure — общезыковая инфраструктура) — спецификация общезыковой инфраструктуры. Она определяет, как работает .NET (а также другие похожие фреймворки вроде Mono и DotGNU).

В CLI у каждого языка есть свой компилятор. Но программы компилируются не в нативный код (исполняемый), а в промежуточный байт-

код CIL (англ. Common Intermediate Language — общий промежуточный язык). Этот язык является языком низшего уровня, похожего по синтаксису на язык ассемблер.

Когда вы запускаете программу, написанную на одном из языков семейства .NET, её байт-код передаётся дальше по цепи в общезыковую исполняющую среду CLR (Common Language Runtime). Там этот байт-код компилируется в нативный и уже начинает выполняться.

В состав платформы входят также следующие компоненты:

- Base Class Library – библиотека классов, используемая при создании пользовательских приложений. В её состав входят средства ввода-вывода, сетевого взаимодействия, работа со строками, коллекциями объектов и т.д.;
- ADO .NET и XML – средства обеспечения доступа к системам управления базами данных и обработки структурированной информации, представленной на языке XML;
- ASP .NET – средства разработки приложений, размещаемых в сети Интернет и реализованных в виде сервис-ориентированных архитектур. Сюда же входят средства организации веб-интерфейса пользователя;
- Windows Forms – средства разработки графического интерфейса в приложениях. Здесь расположены элементы управления, размещаемые в пользовательских формах, обеспечивающих пользователю удобство работы с прикладными программами.

2) Разработка простого консольного приложения

Большинство типов приложений в Visual Studio может быть создано с использованием мастер-сценария, исполняемого средой разработки для создания структуры и заготовок кода приложений. Существует большое число мастеров, применяемых для создания практически любых типов приложений.

В меню «Файл» выберите пункт «Создать», далее в раскрывающемся списке выберите пункт «Новый проект». Появится окно, в котором будут

перечислены все известные мастера, которые среда Microsoft Visual Studio .NET предоставляет для создания приложений различных типов (Рисунок 1.2).

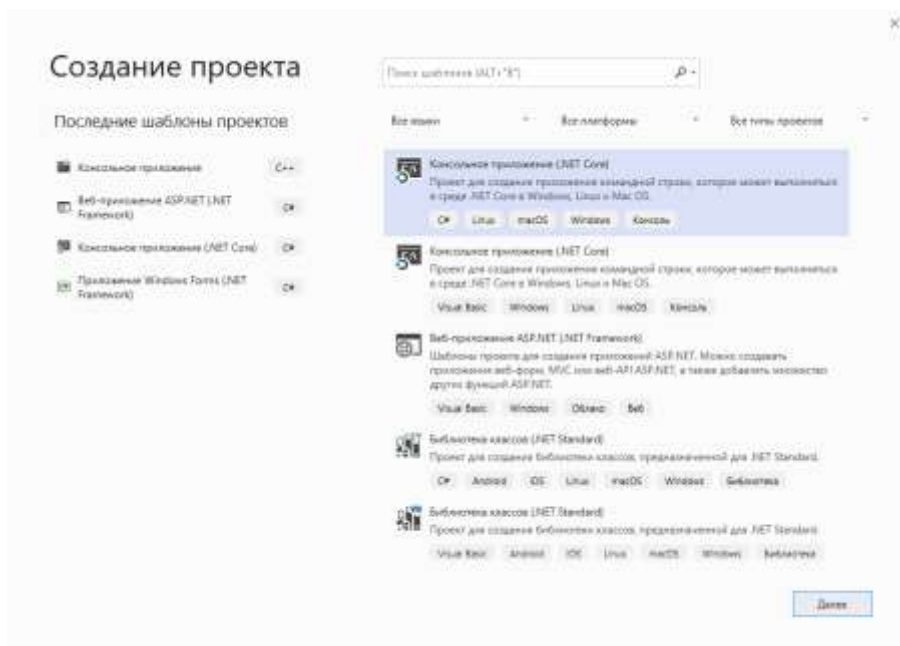


Рисунок 1.2 – Окно создания проекта

Для создания консольного приложения на языке программирования C# следует выбрать категорию мастеров «Консольные приложения C# (.Net Core)». Затем следует указать имя проекта и нажать ОК (Рисунок 1.3).

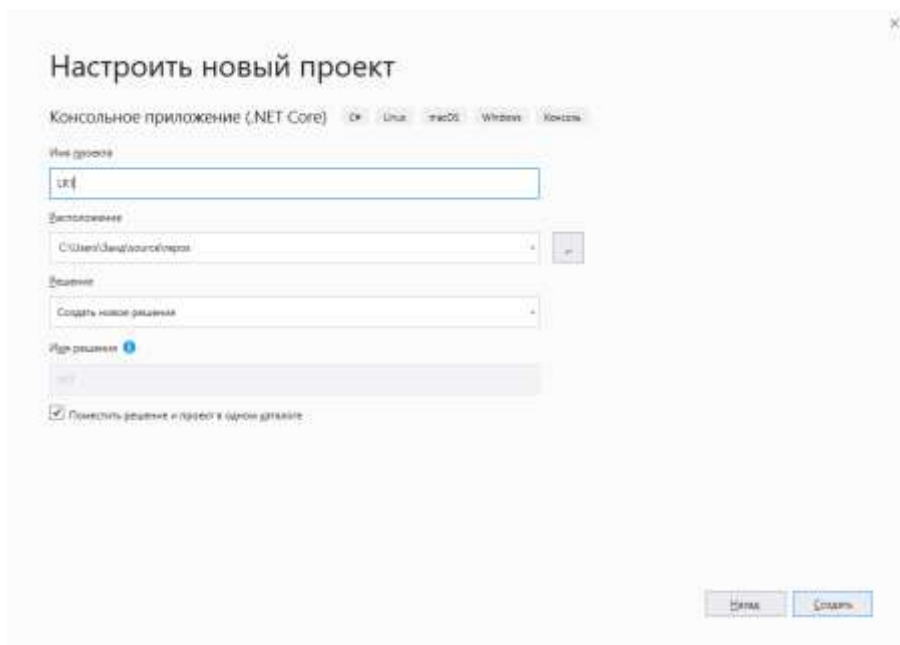


Рисунок 1.3 – Окно создания проекта

После завершения работы мастер создает заготовку кода, приведенную на рисунке 1.4.

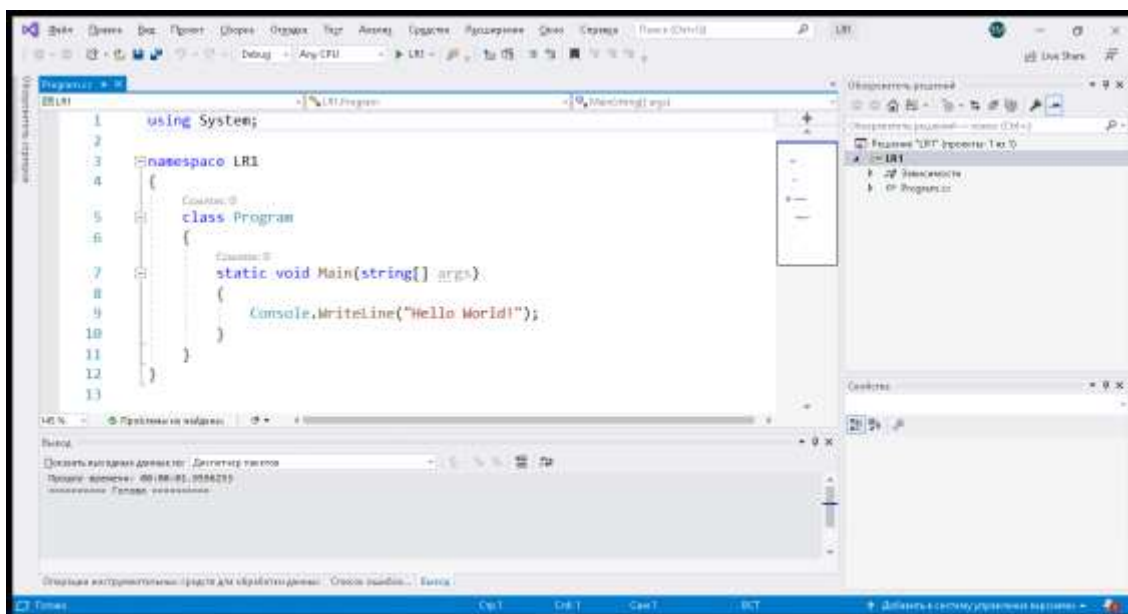


Рисунок 1.4 – Рабочее окно Visual Studio

Как видно из примера (Рисунок 1.5), приведенный код выполняет вывод на консоль стандартного приветственного сообщения «Hello World!», содержит объявление используемых системных библиотек, а также заготовку класса Program со статическим методом Main, являющимся точкой входа в приложения.

```

1      using System;
2
3      namespace LR1
4      {
5          class Program
6          {
7              static void Main(string[] args)
8              {
9                  Console.WriteLine("Hello World!");
10             }
11         }
12     }

```

Рисунок 1.5 – Заготовка кода

Расширим этот пример, добавив к выводу строки с приветственным сообщением имени, сохраненную в инициализированную переменную (Рисунок 1.6).

```

1      using System;
2
3      namespace LR1
4      {
5          class Program
6          {
7              static void Main(string[] args)
8              {
9                  string name = "Zaid";
10                 Console.WriteLine($"Hello World {name}");
11             }
12         }
13     }

```

Рисунок 1.6 – Модифицированная программа

В данном примере мы создали переменную `name` типа `string` (строка 9). Тип данных `string` используется для хранения данных строкового типа. В параметр метода консольного вывода данных (строка 10) мы добавили вывод значения переменной `name` к строке «Hello World». Для встраивания отдельных значений в выводимую на консоль строку используются фигурные скобки, в которые заключается встраиваемое значение. Это можем значение переменной (`{name}`) или более сложное выражение (например, операция сложения `{4 + 7}`). А перед всей строкой ставится знак доллара `$`.

Для того чтобы собрать приложение, следует выбрать пункт меню «Сборка», а в нем команду «Собрать решение». После построения программы компилятор сообщит о завершении или укажет ошибки в тексте программы.

Для запуска приложения следует выбрать пункт меню «Отладка», а в нем команду «Запуск без отладки». После чего на экране, в случае отсутствия ошибок, вы увидите результат выполнения программы (Рисунок 1.7).

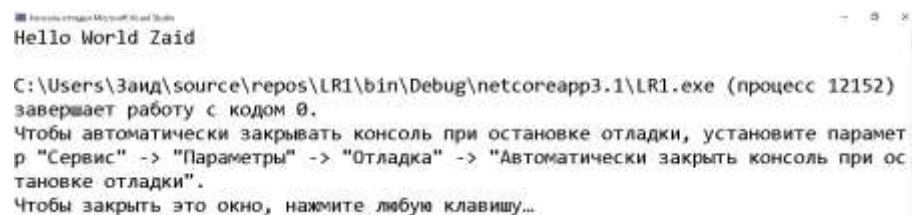


Рисунок 1.7 - Результат выполнения программы

Для создания консольного приложения на языке программирования C++

в окне создания проекта выберите пункт «Консольное приложение C++» (Рисунок 1.8).

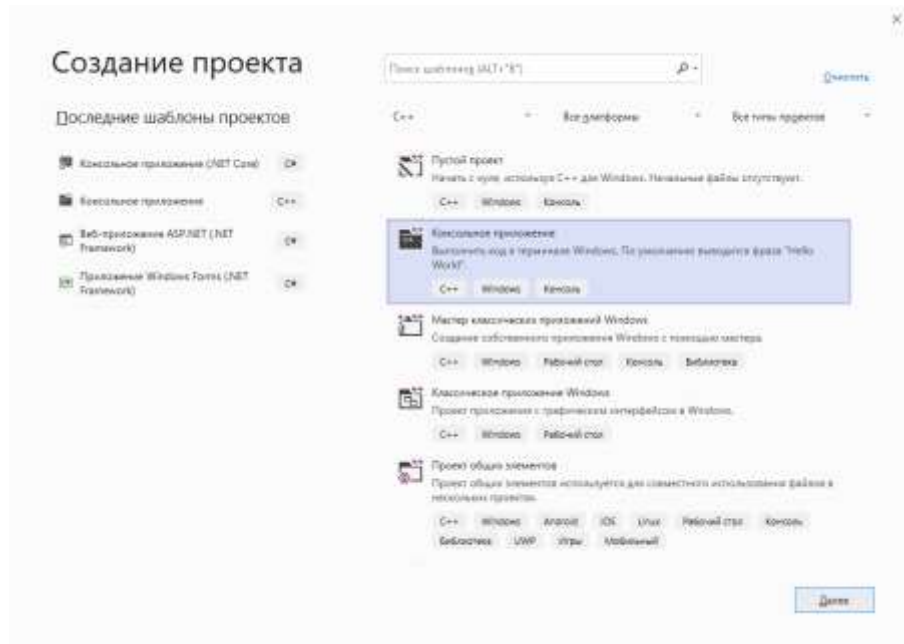


Рисунок 1.8 – Окно создания проекта

Код приложения, эквивалентный созданному ранее, представлен на рисунке 1.9.

```

1  // LP1.2.cpp : Этот файл содержит функцию "main". Здесь
2  //
3
4  #include <iostream>
5
6  int main()
7  {
8      std::string name = "Zaid";
9      std::cout << "Hello World " << name << std::endl;
10 }
```

Рисунок 1.9 – Программа вывода приветственного сообщения на C++

3) Задание на лабораторную работу

Обе части лабораторной работы выполняются на языках программирования C++ и C#.

Часть 1.

Написать простейшую консольную программу, которая выводит приветственное сообщение, введенное с клавиатуры пользователем.

Часть 2.

В соответствии с вариантом необходимо написать простейшую консольную программу, в которой помимо главной функции будет функция по варианту.

Варианты	Индивидуальное задание
1	Вычисление суммы двух вещественных чисел.
2	Вычисление разности двух вещественных чисел.
3	Вычисление произведения двух вещественных чисел.
4	Вычисление частного двух вещественных чисел.
5	Вычисление суммы квадратов двух вещественных чисел
6	Вычисление разности квадратов двух вещественных чисел
7	Вычисление произведения квадратов двух вещественных чисел
8	Вычисление частного квадратов двух вещественных чисел
9	Вычисление квадрата суммы двух вещественных чисел
10	Вычисление квадрата разности двух вещественных чисел
11	Вычисление квадрата произведения двух вещественных чисел
12	Вычисление квадрата частного двух вещественных чисел
13	Вычисление суммы трех вещественных чисел.
14	Вычисление разности трех вещественных чисел.
15	Вычисление произведения трех вещественных чисел.
16	Вычисление частного трех вещественных чисел.
17	Вычисление суммы квадратов трех вещественных чисел
18	Вычисление разности квадратов трех вещественных чисел
19	Вычисление произведения квадратов трех вещественных чисел
20	Вычисление частного квадратов трех вещественных чисел
21	Вычисление квадрата суммы трех вещественных чисел
22	Вычисление квадрата разности трех вещественных чисел
23	Вычисление квадрата произведения трех вещественных чисел
24	Вычисление квадрата частного трех вещественных чисел

25	Возведение первого введенного числа в степень второго введенного числа
26	Вычисление периметра треугольника по введенным длинам трех различных сторон
27	Вычисление периметра прямоугольника по введенной длине и ширине
28	Вычисление площади треугольника по трем известным сторонам
29	Вычисление площади прямоугольника по введенной длине и ширине
30	Вычисление площади окружности
31	Вычисление площади квадрата
32	Вычисление объема параллелепипеда по трем известным сторонам

4) Контрольные вопросы

1. Какие основные элементы входят в состав среды разработки Visual Studio?
2. В чем преимущество использования фреймворка .NET?
3. Перечислите основные части программы, написанной на языке программирования C++?
4. Перечислите основные части программы, написанной на языке программирования C#?
5. Какие операторы используются для реализации консольного ввода/вывода данных?
6. Что такое переменная?