Введение в программирование на языке Си

Лекция 1

Языки программирования

- Язык программирования — это формализованная система записи алгоритмов, предназначенная для передачи компьютеру информации о том, какие действия он должен выполнять. Язык должен быть понятен как человеку-программисту, так и машине после соответствующей обработки.
- Языки программирования принято делить на несколько уровней:
- **машинные языки**, непосредственно соответствующие двоичным командам процессора;
- **языки низкого уровня** — ассемблеры, где команды описываются с использованием мнемоник, но сохраняется прямая связь с машинными инструкциями;
- **языки высокого уровня**, предоставляющие более абстрактные конструкции: операторы ветвления, циклы, функции, структуры данных.

Версии языка СИ

•**K&R C (1978)** – первая описанная версия, без строгой типизации, закреплена в книге Кернигана и Ритчи.
•**ANSI C / C89 (1989)** – стандартизация, строгая проверка типов, прототипы функций, стандартные библиотеки.
•**C90 (1990)** – международный стандарт (ISO), эквивалент ANSI C.
•**C95 (1995)** – мелкие улучшения, поддержка широких символов (Unicode через wchar_t).
•**C99 (1999)** – новое поколение: long long, массивы переменной длины, комплексные числа, //-комментарии.
•**C11 (2011)** – многопоточность, атомарные операции, расширенный Unicode, безопасные функции.
•**C17 (2017)** – исправление ошибок и уточнения, без новых возможностей.
•**C23 (2023)** – современный стандарт: удобная работа со строками, расширенный Unicode, повышение безопасности.

Особенности языка Си

- **Эффективность** – близок к машинному коду, программы работают очень быстро.
- **Переносимость** – код можно запускать на разных архитектурах с минимальными изменениями.
- **Простая структура** – небольшой набор ключевых слов (около 30).
- **Гибкость** – допускает как высокоуровневое, так и низкоуровневое программирование.
- **Указатели** – прямой доступ к памяти, возможность работы с адресами.
- **Модульность** – программы состоят из функций и файлов, легко разбивать проект на части.
- **Богатый набор операторов** – арифметика, логика, битовые операции.
- **Базовый язык для многих других** – C++, C#, Java, Objective-C, даже Python заимствует синтаксис.

Недостатки языка Си

- **Сложность для новичков** – требует понимания памяти, типов и указателей.
- **Нет встроенной поддержки ООП** – только процедурный стиль.
- **Низкий уровень защиты** – легко сделать ошибки с памятью (утечки, выход за границы массива).
- **Мало средств безопасности** – нет автоматической проверки типов массивов или строк.
- **Трудности в больших проектах** – по сравнению с современными языками, меньше инструментов для управления сложностью.
- **Стандартная библиотека ограничена** – базовые функции ввода-вывода, строки и математика, но нет «готовых» коллекций (например, списков, словарей).

Алгоритм. Программа. Исполнитель

- Алгоритм – концептуальное описание шагов, которые необходимо выполнить для решения той или иной задачи, в то время как программа – это запись алгоритма на специальном языке (программирования), который понятен исполнителю. Исполнитель – нечто, что умеет шаг за шагом выполнять программу, написанную на понятному ему языке.

Пример 1

- Алгоритм нахождения минимального элемента в заданной последовательности чисел:

1. Запомним значение первого элемента последовательности как минимальное.

2. Перебираем все элементы последовательности, выполняя для каждого:

а) если значение текущего элемента меньше минимального, то примем его за минимальное.

7 5 6 9 8 4 2 3 1

Пример 2

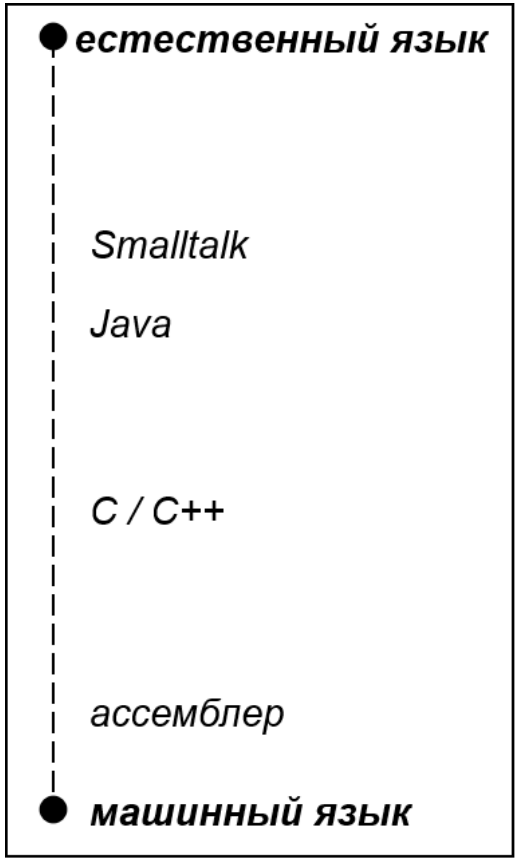

Задача «Отсортировать массив чисел»
Алгоритм сортировки вставками
Алгоритм пузырьковой сортировки
Алгоритм сортировки слиянием
...
Алгоритм быстрой сортировки
Код на языке C/C++
Код на языке Python
Код на языке Java
Код на языке C#
...
Код на языке C/C++
Код на языке Python
Код на языке Java
Код на языке C#
...
Код на языке C/C++
Код на языке Python
Код на языке Java
Код на языке C#
...
Код на языке C/C++
Код на языке Python
Код на языке Java
Код на языке C#
...

Уровень языка

– это его позиция в шкале «компьютер-человек».

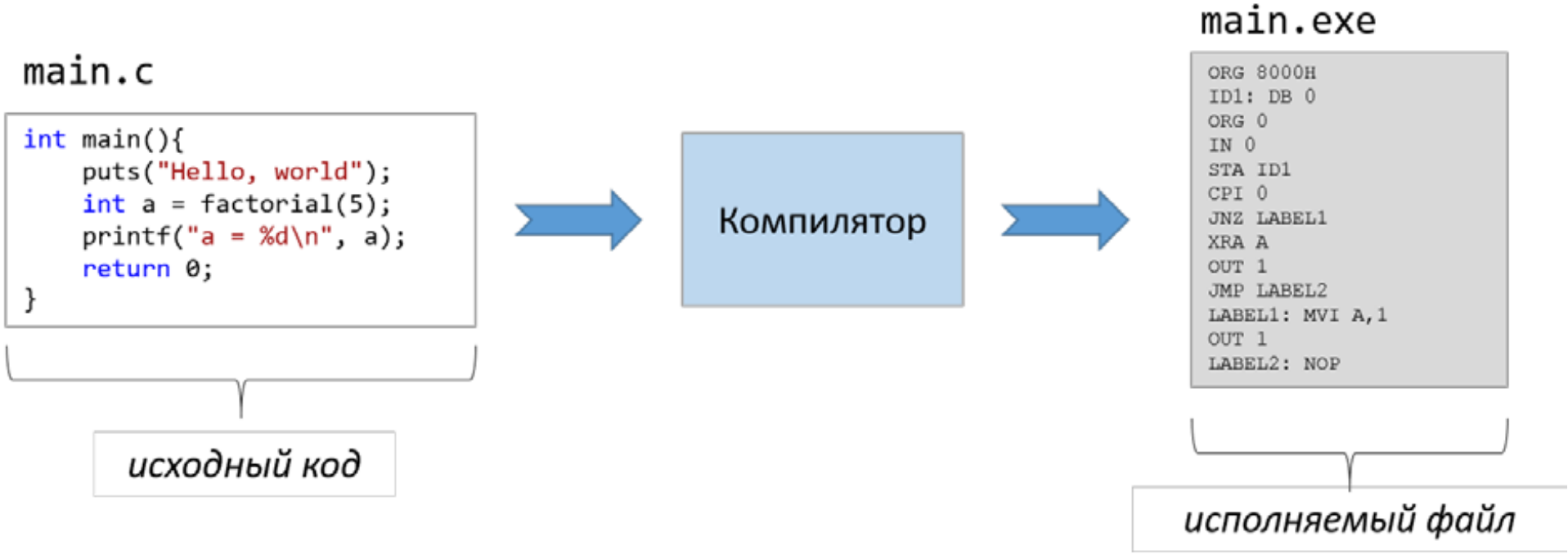
Чем ниже уровень – тем более он понятен компьютеру и менее понятен человеку, и наоборот.

Программы, написанные на языках высокого уровня, похожи на тексты на естественном (чаще всего английском) языке. Например, смысл строки кода «`if (a > b) then min = b`» несложно уловить даже не программисту.



Компилятор

— это программа, которая переводит исходный код, написанный на языке программирования высокого уровня, в машинный код или другой низкоуровневый язык, понятный компьютеру.



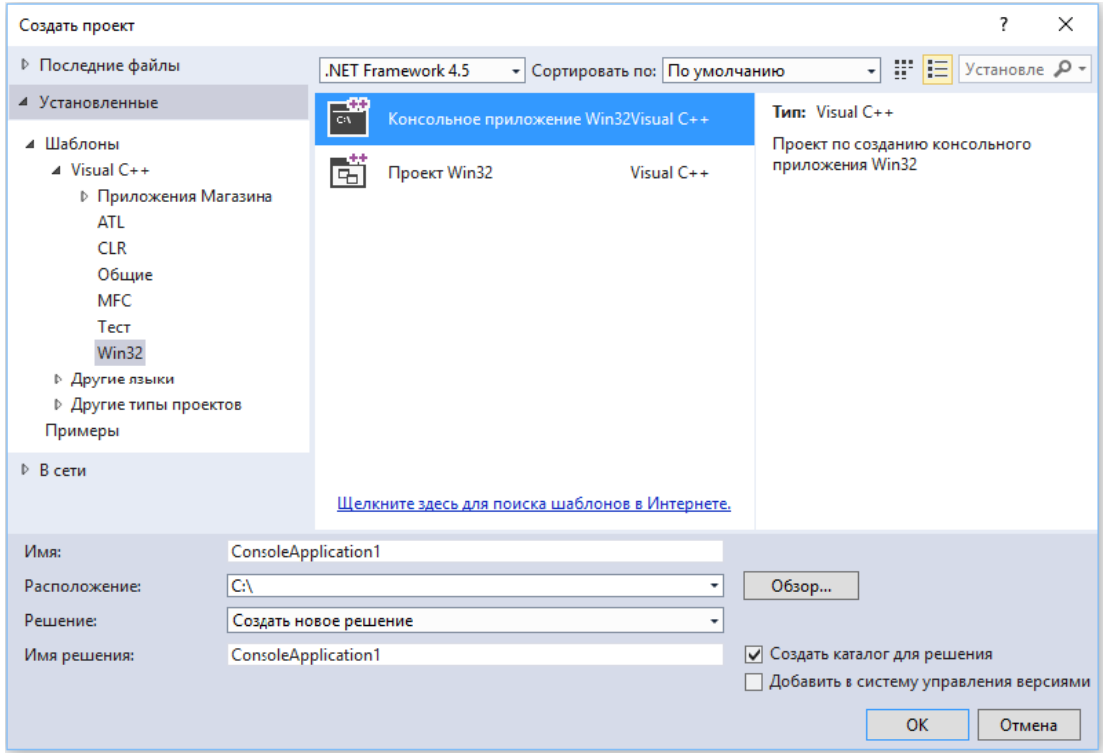
Работа в IDE

- **IDE (Integrated Development Environment)** — это *интегрированная среда разработки*, то есть программный комплекс, в котором программисту удобно писать, отлаживать и запускать код.
- Современный IDE состоит из:

1. интеллектуального текстового редактора с функциями структурирования и подсветки кода, автодополнения и т.д.,
2. встроенной справки,
3. компилятора,
4. линковщика,
5. библиотек кода,
6. средств отладки,
7. средств автоматизированной сборки приложения,
8. средств для интеграции с системами управления версиями кода,
9. средств для совместной разработки,
10. средств для взаимодействия с базами данных,
11. всевозможных утилит, упрощающие разработку и отладку кода

Создание проекта

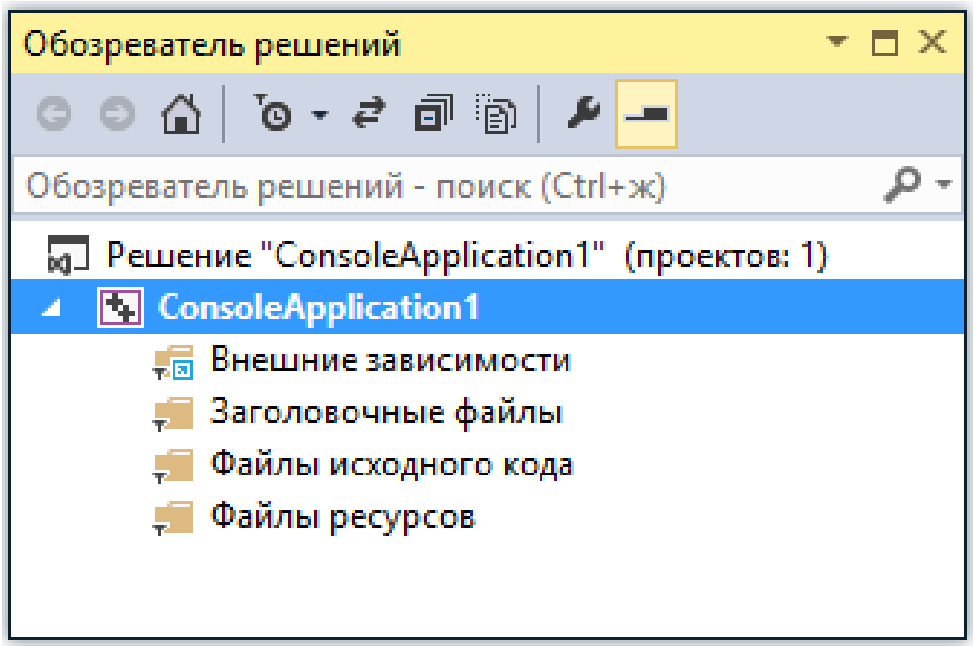
- Нажимаем на ссылку «Создать проект…»
- Шаблон Visual C++, подшаблон Win32, тип «Консольное приложение Win32»
- Указываем место, где будет создан проект, имя проекта и имя решения
- Выбрать пустой проект для консольного приложения



Обозреватель решений

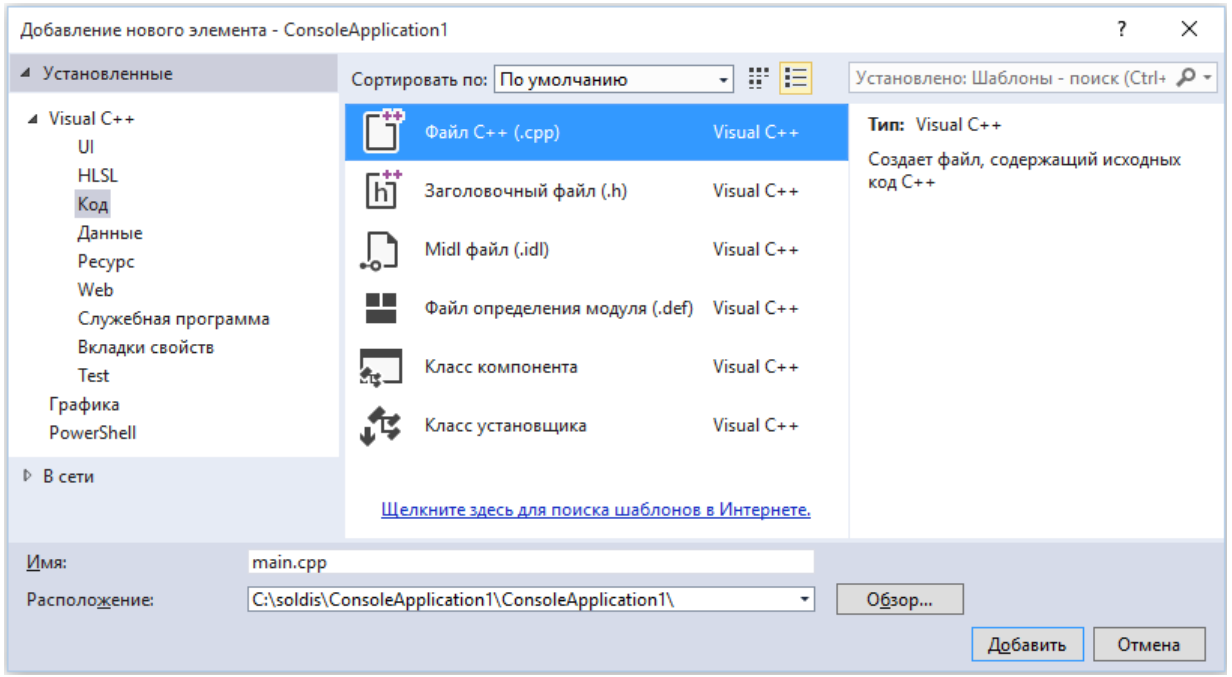
- Здесь изображена логическая структура нашего проекта– на верхнем уровне представлено решение, в нем перечислены проекты, в которых идут папки для заголовочных файлов, файлов с исходным кодом и файлами ресурсов.

- Если обозреватель решений не виден, его можно включить в меню «Вид»



Шаблон файлов

- Нажимаем правой кнопкой на папку «Файлы исходного кода»
- Выбираем пункт «Создать новый элемент».
- Открывается окно с набором шаблонов файлов, которые мы можем создать/добавить в проект – от элементов графического интерфейса до файлов ресурса. Нам надо выбрать Код –> Файл C++
- Указываем его имя
- Нажимаем «Добавить»

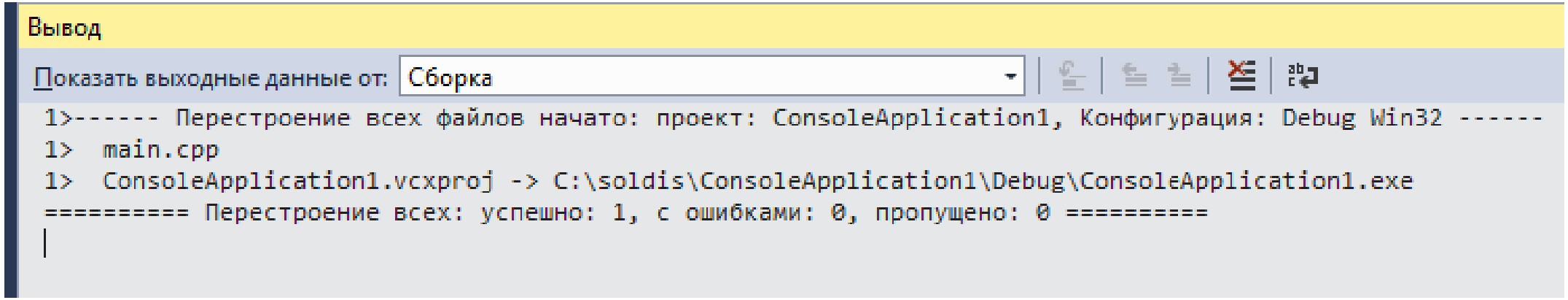


Пример кода

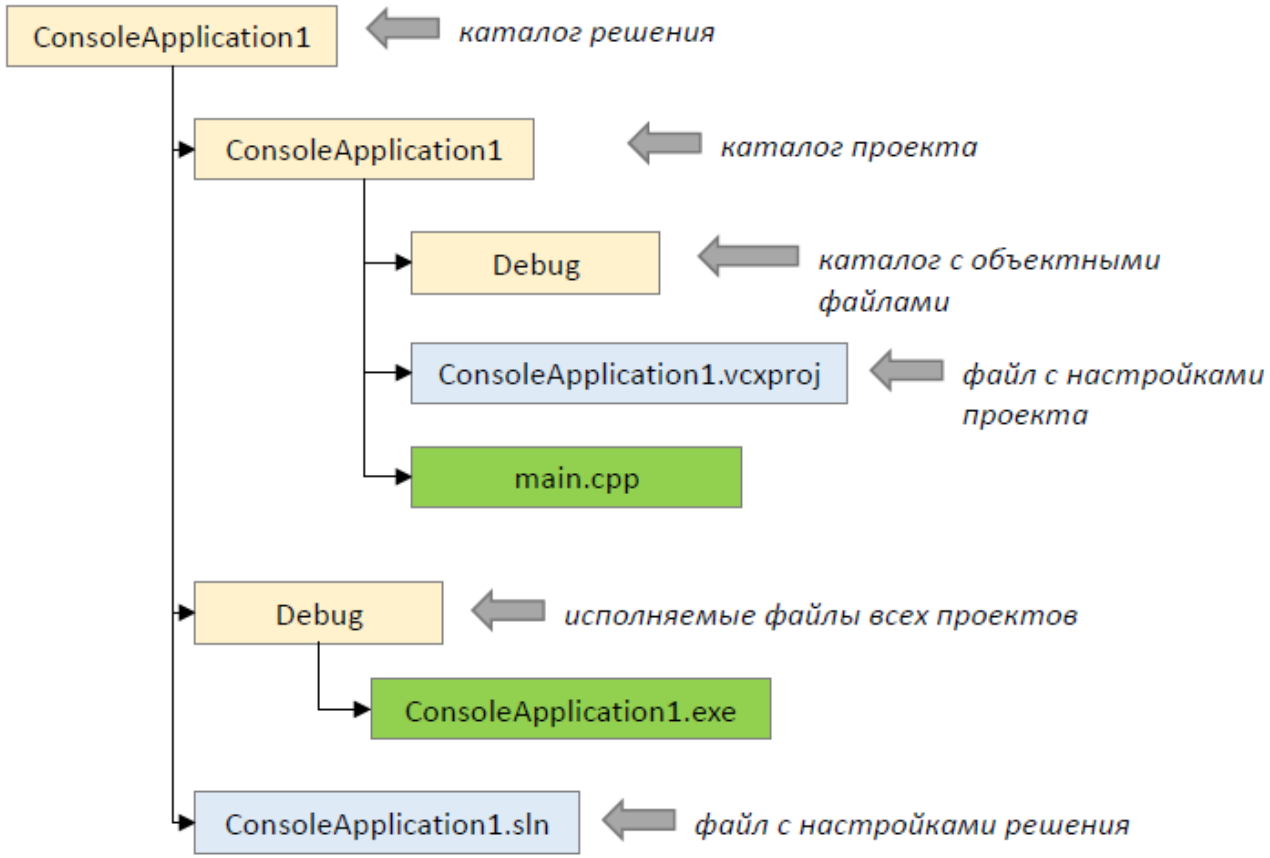
```c
#include <stdio.h>

int main(){
     puts("Hello, world");
     int a = 2;
     printf("a = %d\n", a);
     return 0;
}
```

Компиляция

- Для этого можно выбрать в меню «Сборка» пункт «Собрать решение» или просто нажать на клавиатуре <F7>. В окне «Вывод» снизу экрана мы видим сначала ход, а затем и результат компиляции

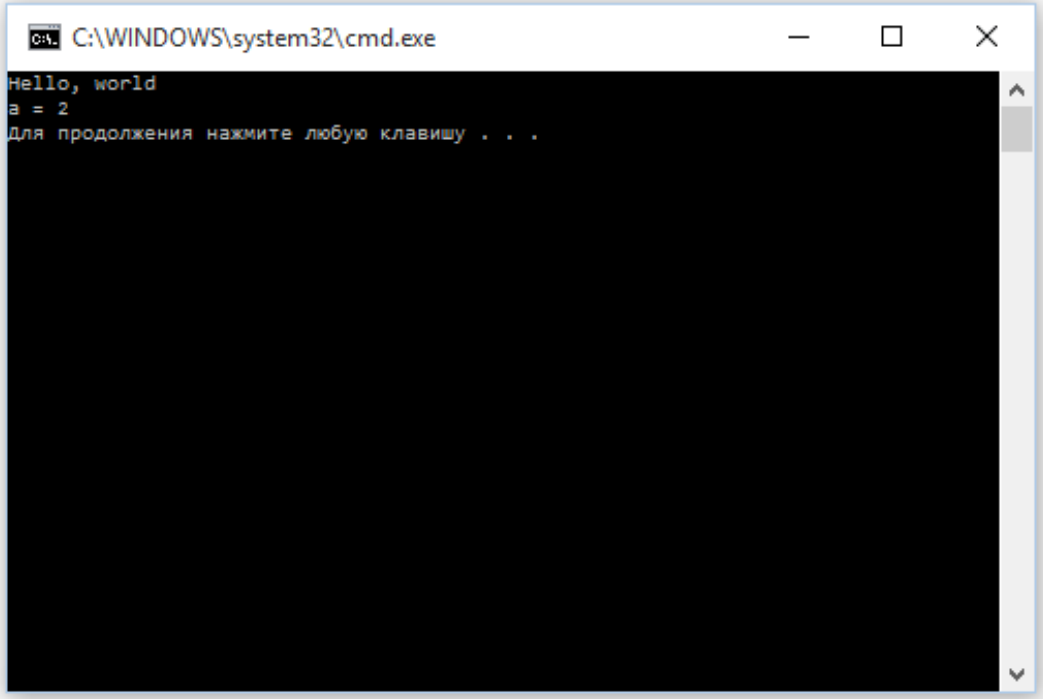


Содержимое каталога решения



Запуск приложения

- Мы можем это сделать, нажав кнопку <F5>, однако в этом случае приложение запустится в новом консольном окне, отработает и исчезнет, не дав нам увидеть результат своей работы. Чтобы задержать консольное окно, приложение надо запускать с помощью комбинации клавиш <Ctrl>-<F5>. В этом случае после выполнения программы консольное окно будет держаться на экране до тех пор, пока мы не нажмем любую кнопку на клавиатуре



Отладка

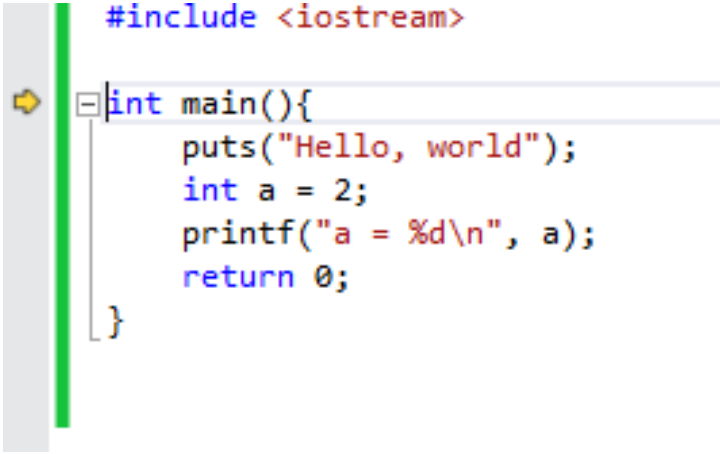
- Обратим внимание, что еще до компиляции студия выделила красной волнистой чертой те фрагменты кода, которые вызывают у нее вопросы. Это происходит благодаря тому, что студия делает подробный анализ кода уже тогда, когда вы его только набираете в текстовом редакторе, практически осуществляя его «пробную» компиляцию на лету. Это позволяет заметить ошибки еще до процесса сборки решения, который для сложных программных продуктов может длиться часами.

```
#include <iostream>

int main(){
    puts("Hello, world");
    int a = 2;
    printf(a = %d\n", a);
    return 0;
}
```

```
1>------ Сборка начата: проект: ConsoleApplication1, Конфигурация: Debug Win32 ---
1>  main.cpp
1>…\main.cpp(6): error C2017: недопустимая escape-последовательность
1>…\main.cpp(6): error C2065: d: необъявленный идентификатор
1>…\main.cpp(6): error C2001: newline в константе
1>…\main.cpp(6): error C2146: синтаксическая ошибка: отсутствие ")" перед
идентификатором "n"
========== Сборка: успешно: 0, с ошибками: 1, без изменений: 0, пропущено: 0 ========
```

Пошаговое выполнение программы

- <F10> – приложение запускается открывается и «зависает» консольное окно, а в текстовом редакторе студии напротив одной из строк появляется желтая стрелка
- Нажать <F10>, чтобы сделать следующий шаг
- Нажать <F5> , если мы хотим закончить пошаговое выполнение
- Если желтая стрелка указывает на строку, содержащую вызов функции, то, чтобы зайти «внутрь», нужно нажать < F11>



Точка остановы

- Если код слишком большой и «шагать» до нужного места, которое хочется посмотреть повнимательнее, слишком долго, то можно воспользоваться механизмом *точек останова*. Для этого устанавливаем курсор на нужную строку, с которой мы хотели бы начать отладку, и нажимаем <F9>. Напротив этой строки появится красный маркер. Дальше нажимаем <F5> – приложение запускается, начинает работать и как только оно дойдет до помеченной красным кружком строки, оно перейдет в режим пошаговой отладки. Точек останова можно ставить произвольное количество. Чтобы снять точ- ку останова, нужно на соответствующей строке снова нажать <F9>.

Написание простейшей программы

```
1. #include <stdio.h>
2.
3. int main(){
4. /* Выводим на экран строку */
5. printf("Hello, World!");
6.
7. return 0;
8. }
```

- Строка 1 содержит команду препроцессора, подключающую заголовочный файл стандартной библиотеки Си, который дает нашей программе возможность печатать сообщения на экран. stdio.h – библиотека для работы с вводом/выводом информации (standard input/output – стандартный ввод/вывод).
- Строки 3-8 содержат определение главной функции программы. Функция языка Си, как и любого другого структурированного языка программирования, это именованный фрагмент кода, который можно вызывать по его имени

Написание простейшей программы

```
1. #include <stdio.h>
2.
3. int main(){
4. /* Выводим на экран строку */
5. printf("Hello, World!");
6.
7. return 0;
8. }
```

- Оператор– это отдельная инструкция, заканчивающаяся точкой с запятой.
- Точка с запятой – обязательный разделитель.
- На строке 5 указан оператор вызова библиотечной функции печати. Вызов функции в языке Си выглядит следующим образом: сначала пишется имя функции, а затем в круглых скобках через запятую перечисляются передаваемые в функцию аргументы. Функция printf (от англ. print formatted, форматированный вывод) печатает на экран переданную ей строчку. Строки в языке Си выделяются двойными кавычками. Круглые скобки – обязательный синтаксис вызова функции.

Написание простейшей программы

```
1. #include <stdio.h>
2.
3. int main(){
4. /* Выводим на экран строку */
5. printf("Hello, World!");
6.
7. return 0;
8. }
```

- 7-я строка – это оператор возврата из функции. Под «возвратом» понимается завершение работы данной функции, а так как main – это главная функция, то данный оператор завершает выполнение всей нашей программы.
- Строка 4 – это пример многострочного комментария. Эта строчка игнорируется компилятором. Она необходима программистам, чтобы понимать, что делает программа.