АЛГОРИТМИЗАЦИЯ
И
ПРОГРАММИРОВАНИЕ

Компьютер без программ – как человек без души, без чувств, без мыслей.

С помощью программ-браузеров мы заходим в Интернет, с помощью программы для печати – пишем тексты, а программа-фотоальбом позволяет нам просматривать фотографии, сохраненные на компьютере.

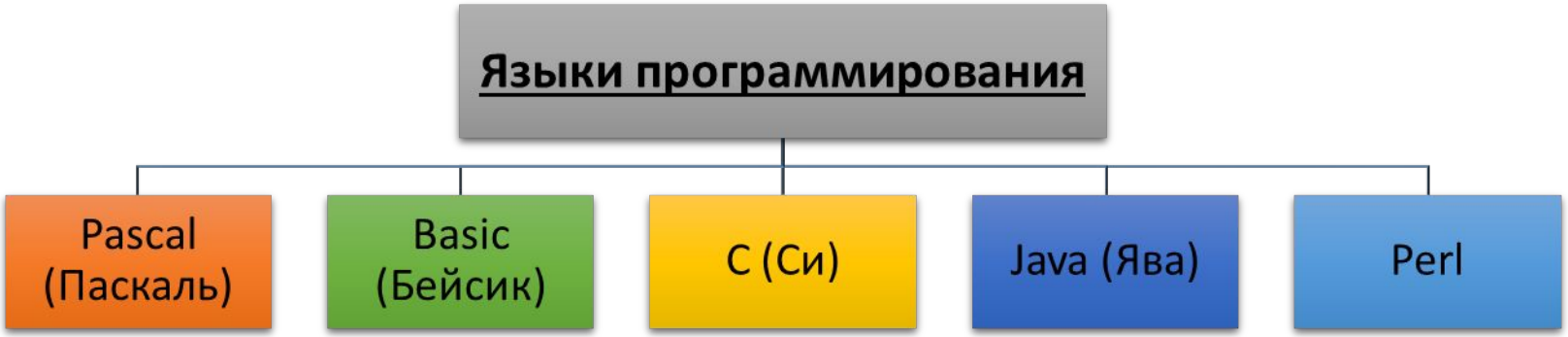
Алгоритм и программа

Для представления алгоритма в виде, понятном компьютеру, служат так называемые языки программирования.
Сначала всегда разрабатывается алгоритм действий, а потом он записывается на одном из таких языков.

Если говорить проще, то каждая нация говорит на своем языке. Для того, чтобы общаться с ее представителями, нужно выучить их язык.

С компьютером все точно также. Мы учим язык программирования и разговариваем с ним на компьютере! Точнее, это делают программисты. А потом уже данный код компьютер конвертирует в понятный для нас вид.


Человеческие языки
Русский
Английский
Китайский
Хинди
Арабский


Языки программирования
Pascal (Паскаль)
Basic (Бейсик)
C (Си)
Java (Ява)
Perl

Таким образом, чтобы создать (написать) программу необходимо знать язык программирования.
Каждый язык отличается от другого своими методами и назначениями (для обучения, для создания игр и т.д.)

Возможности языка C++

используя язык C++, можно создавать следующие решения

- настольное приложение
- Веб Серверы
- драйверы устройств
- Операционные системы
- Игры и игровые движки
- компиляторы

Среда разработки

NetBeans

Visual Studio

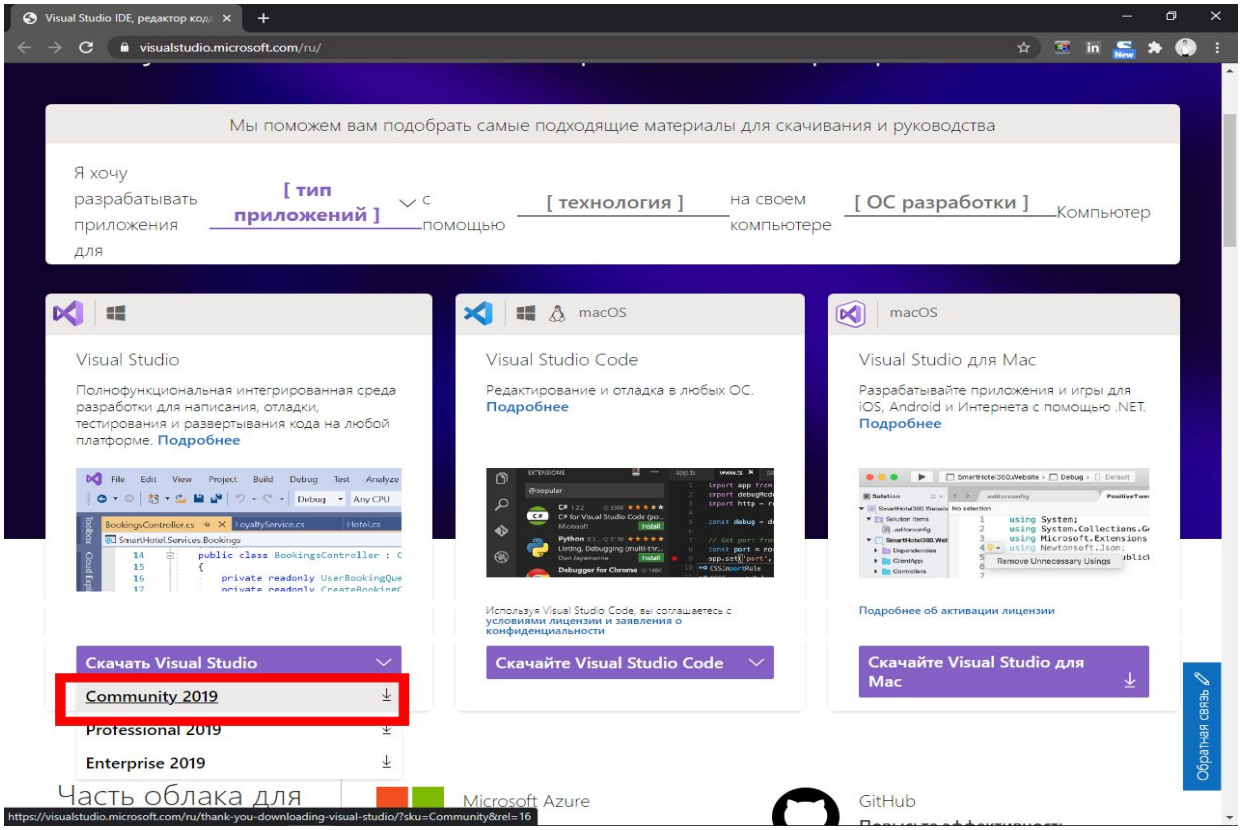
установка

Visual Studio

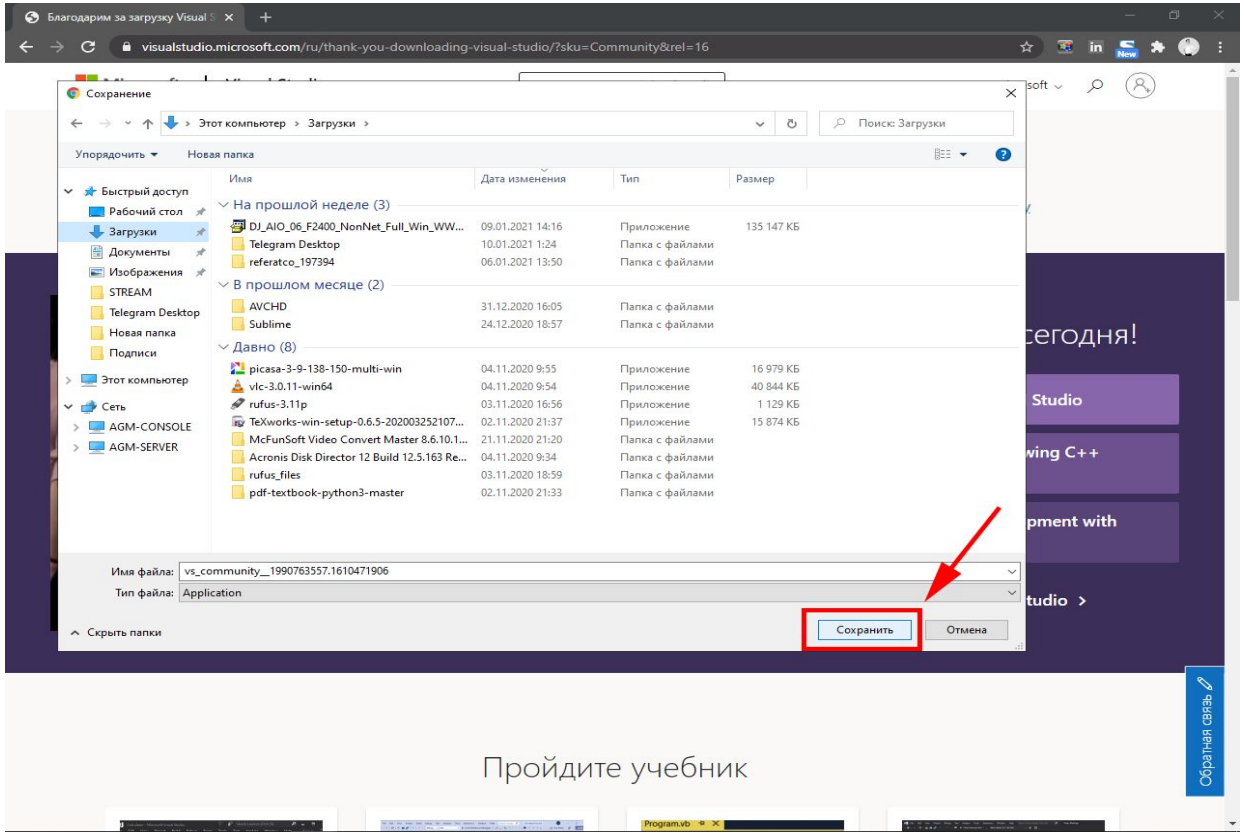
переходим по ссылке:

https://visualstudio.microsoft.com/ru/

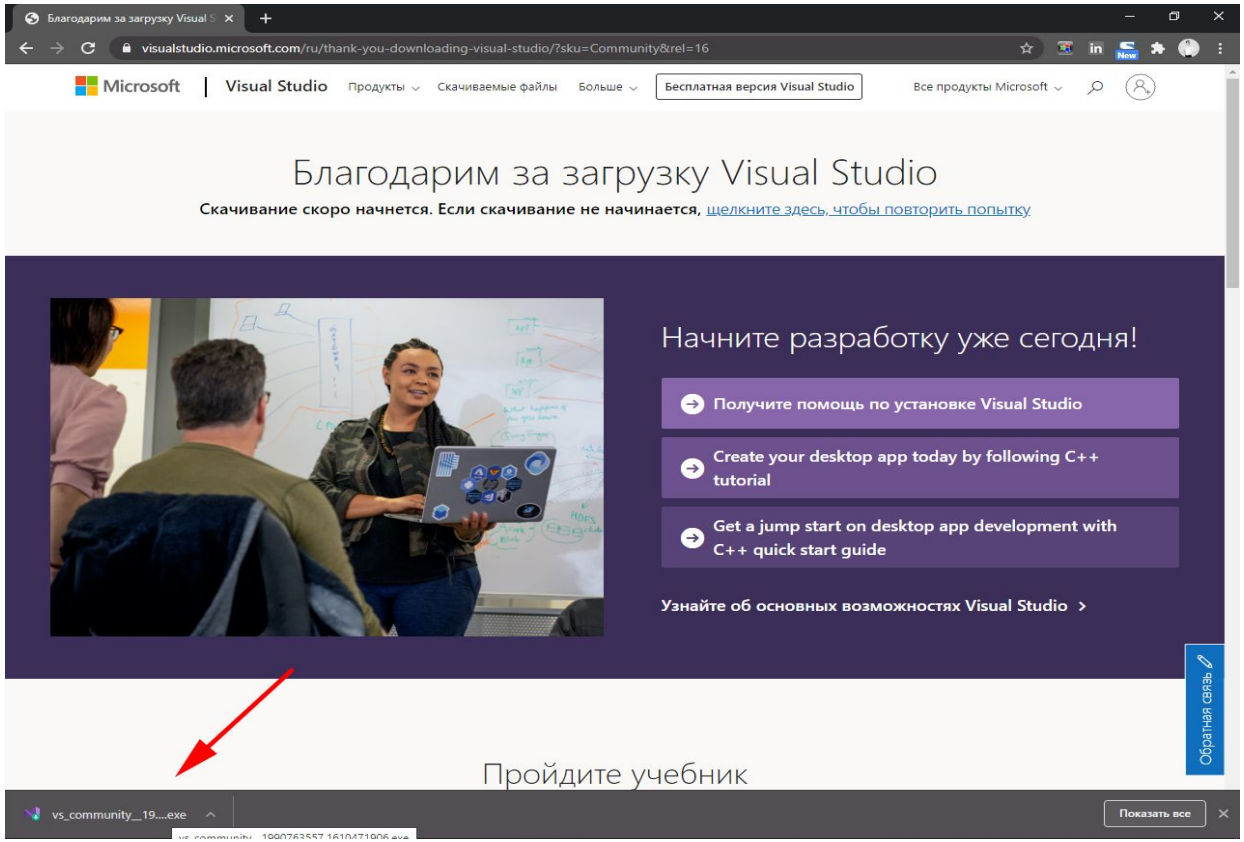
На открывшемся сайте в блоке Visual Studio выбираем «скачать Visual Studio», в выпадающем списке кликаем «Community 2019»



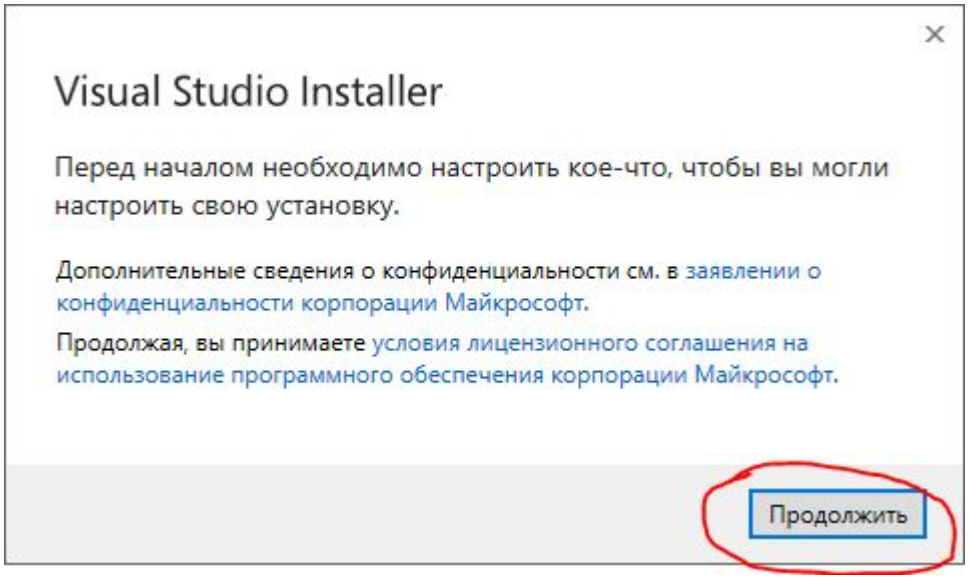
сохраняем файл в удобном для вас месте



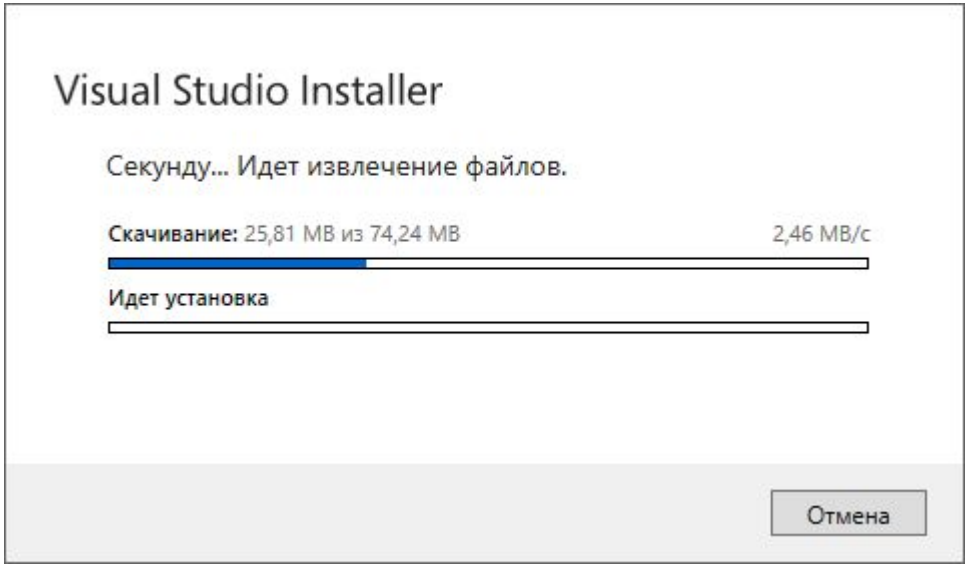
запускаем скачанный файл



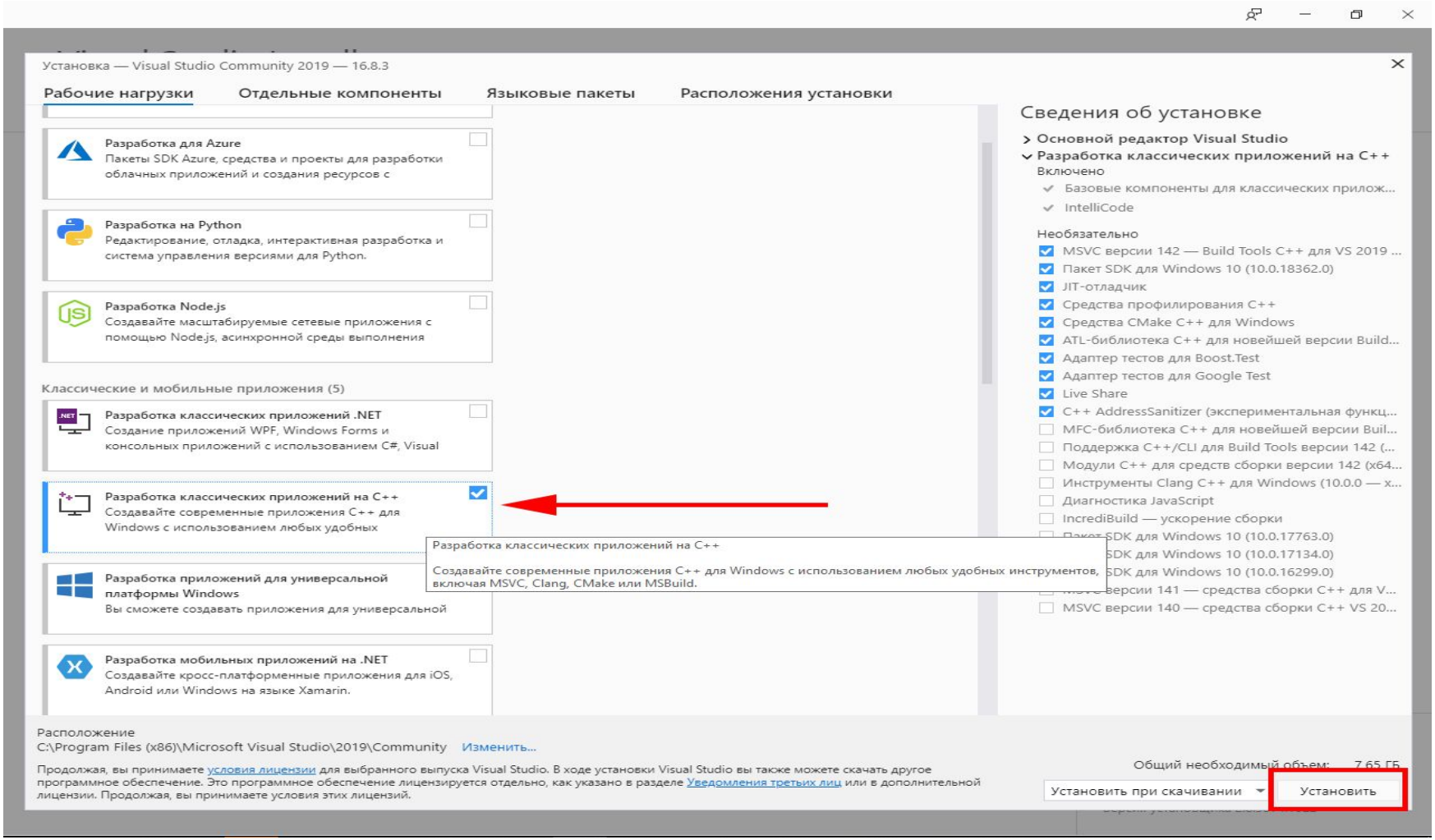
Нажмите «продолжить»



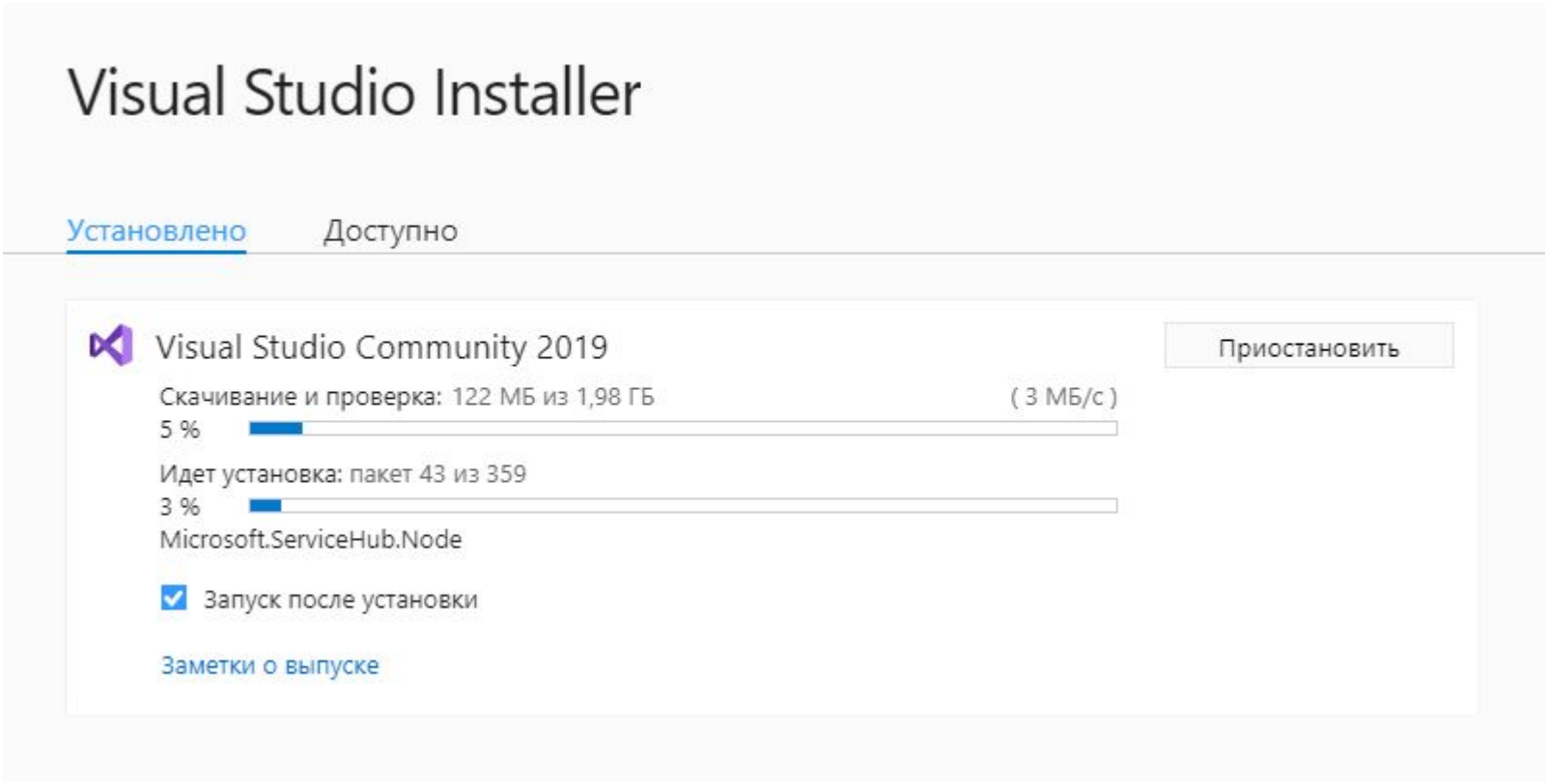
терпеливо ждем загрузки и установки необходимых компонентов



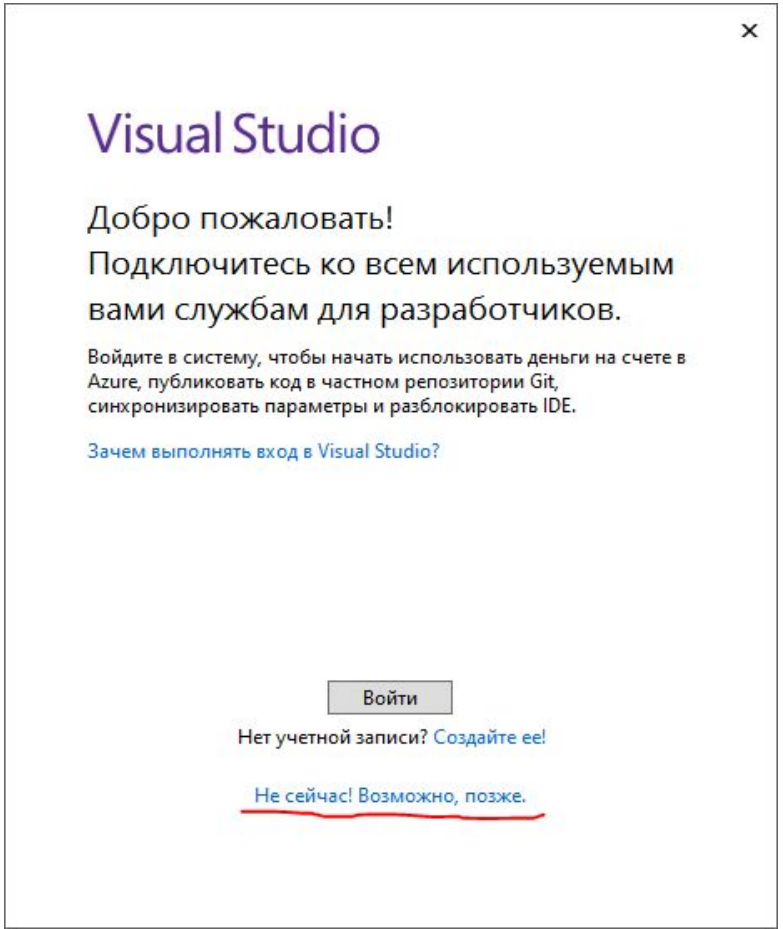
устанавливаем флажок на пункте
«Разработка классических приложений на C++»
и нажимаем «Установить»



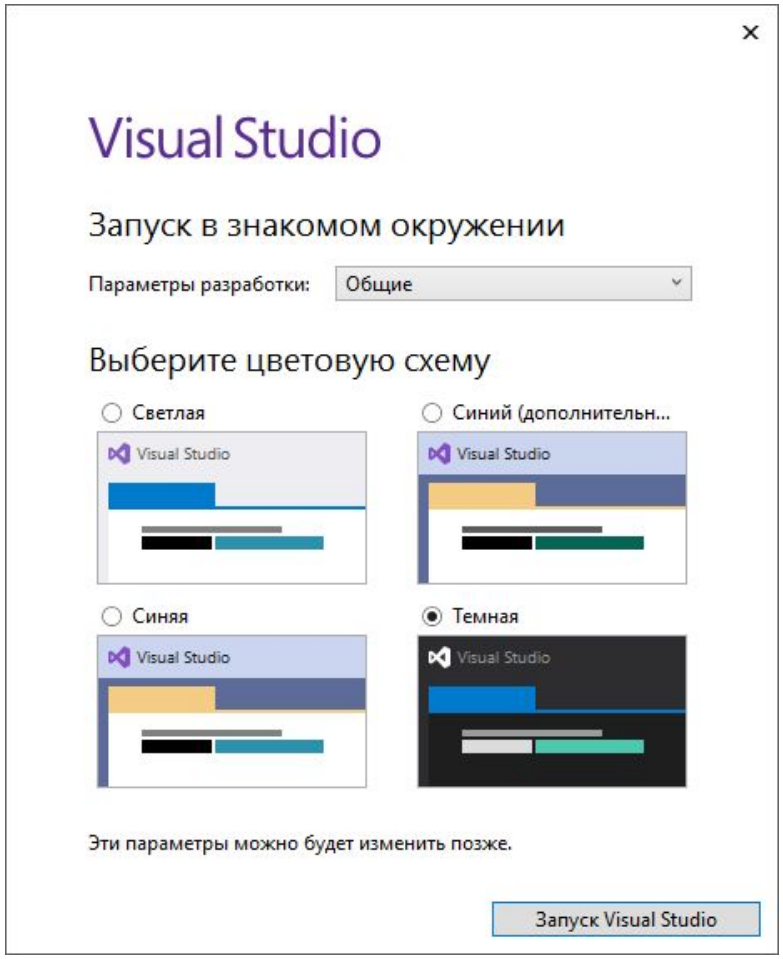
ждем.


Visual Studio Installer
Установлено
Доступно
Visual Studio Community 2019
Приостановить
Скачивание и проверка: 122 МБ из 1,98 ГБ
(3 МБ/с)
5 %
Идет установка: пакет 43 из 359
3 %
Microsoft.ServiceHub.Node
Запуск после установки
Заметки о выпуске

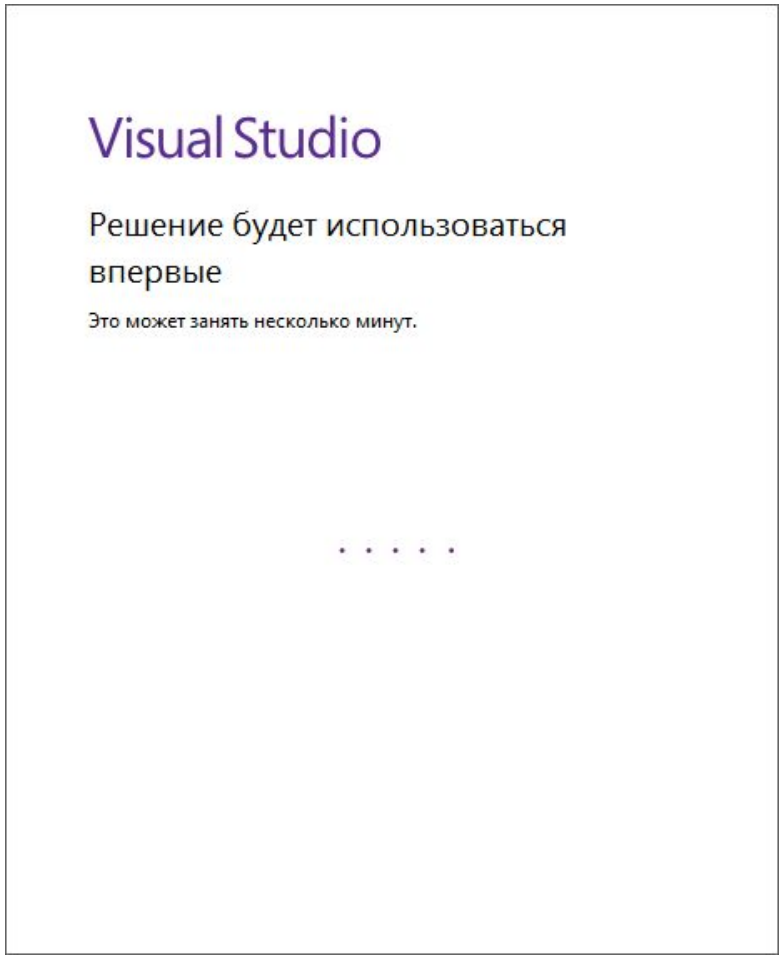
Откроется окно приветствия, если есть учетная запись Microsoft – можете авторизоваться, если нет – нажимаем «Не сейчас! Вожможно, позже.»



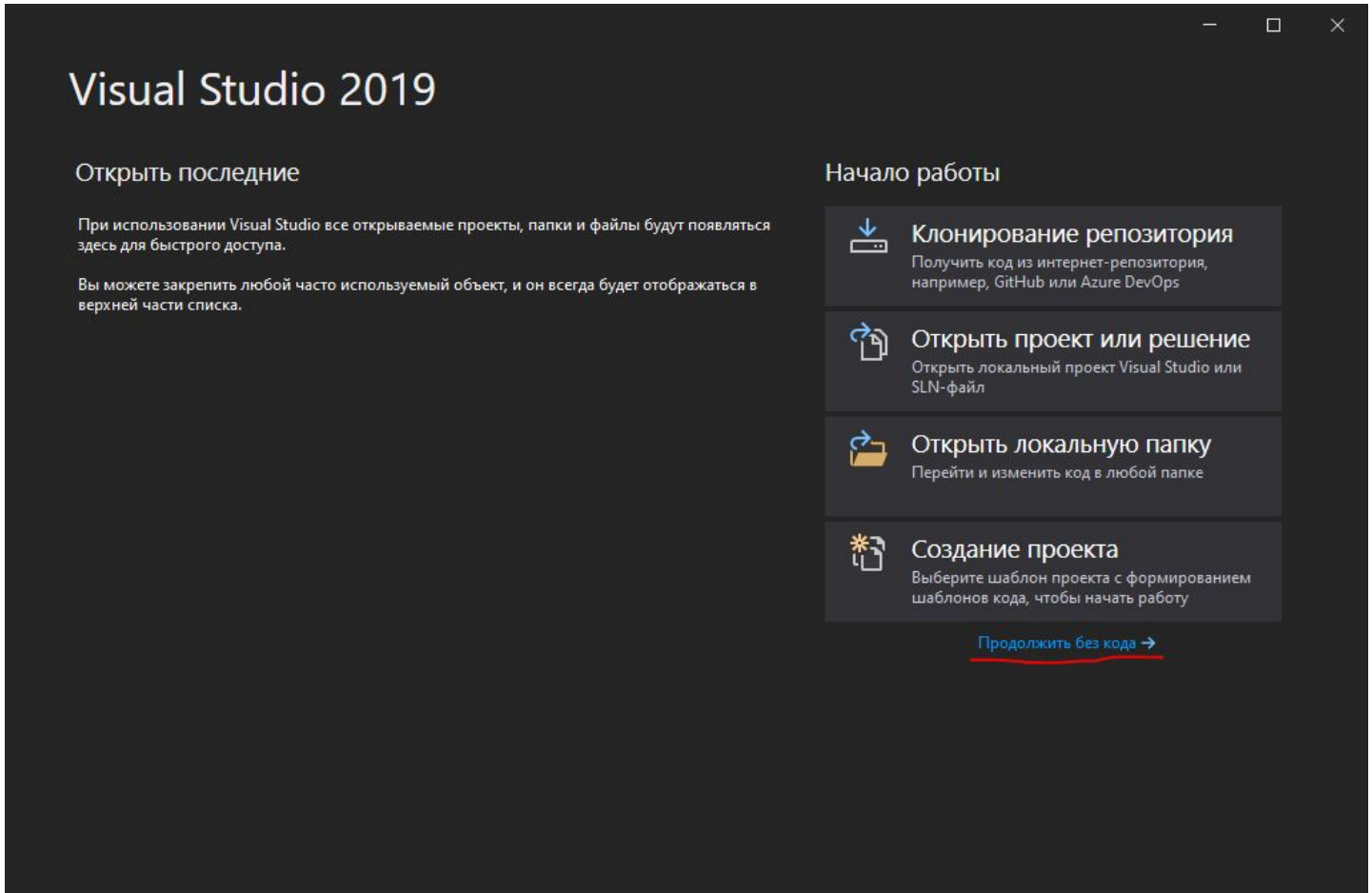
Нажимаем «Запуск Visual Studio»

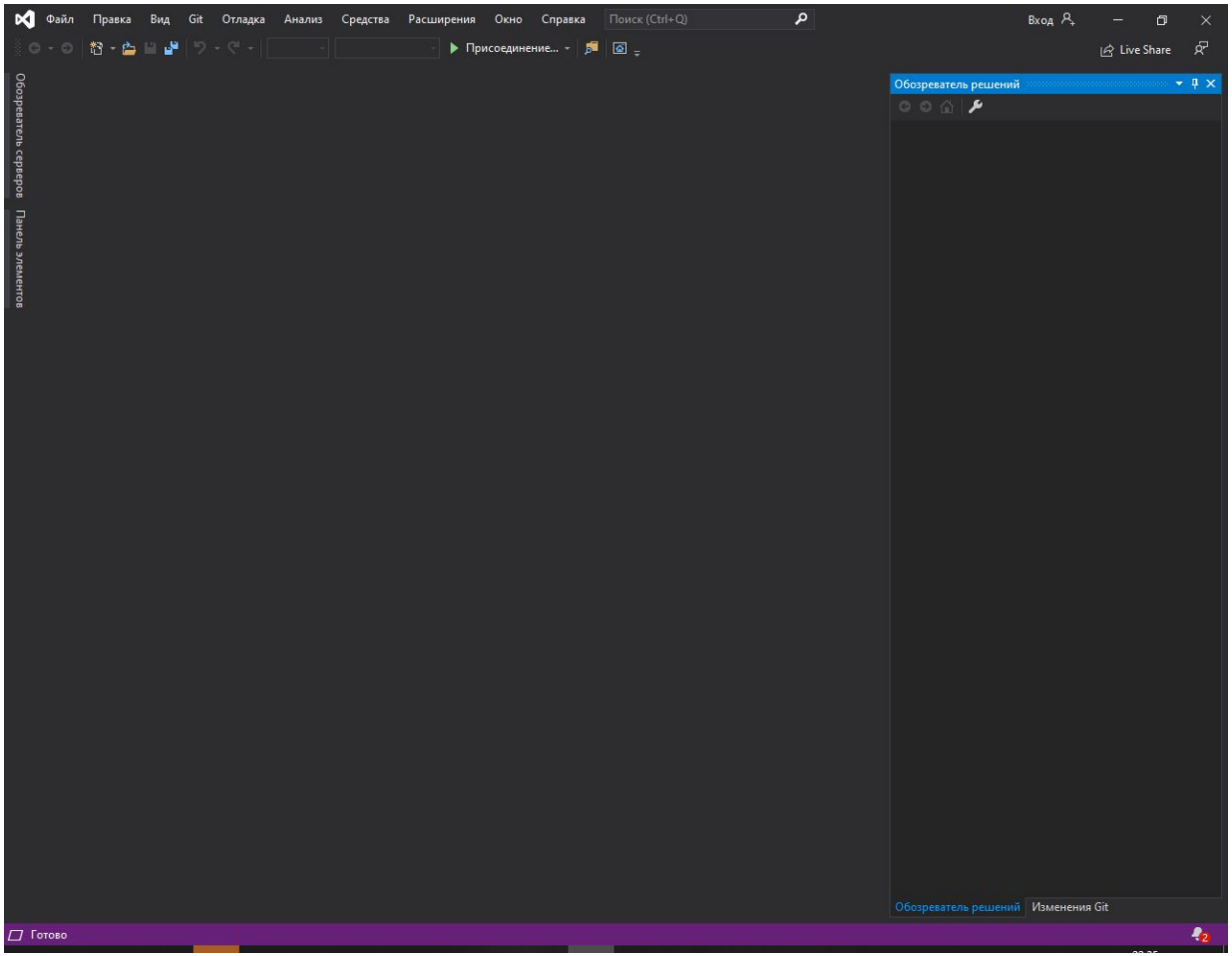


ждем.


Visual Studio
Решение будет использоваться впервые
Это может занять несколько минут.

Все готово! Теперь можешь создать свой первый проект или ознакомиться с интерфейсом, нажав «Продолжить без кода»




Файл
Правка
Вид
Git
Отладка
Анализ
Средства
Расширения
Окно
Справка
Поиск (Ctrl+Q)
Вход
Live Share
Присоединение...
Обозреватель серверов
Панель элементов
Обозреватель решений
Обозреватель решений
Изменения Git
Готово

Структура программы С++

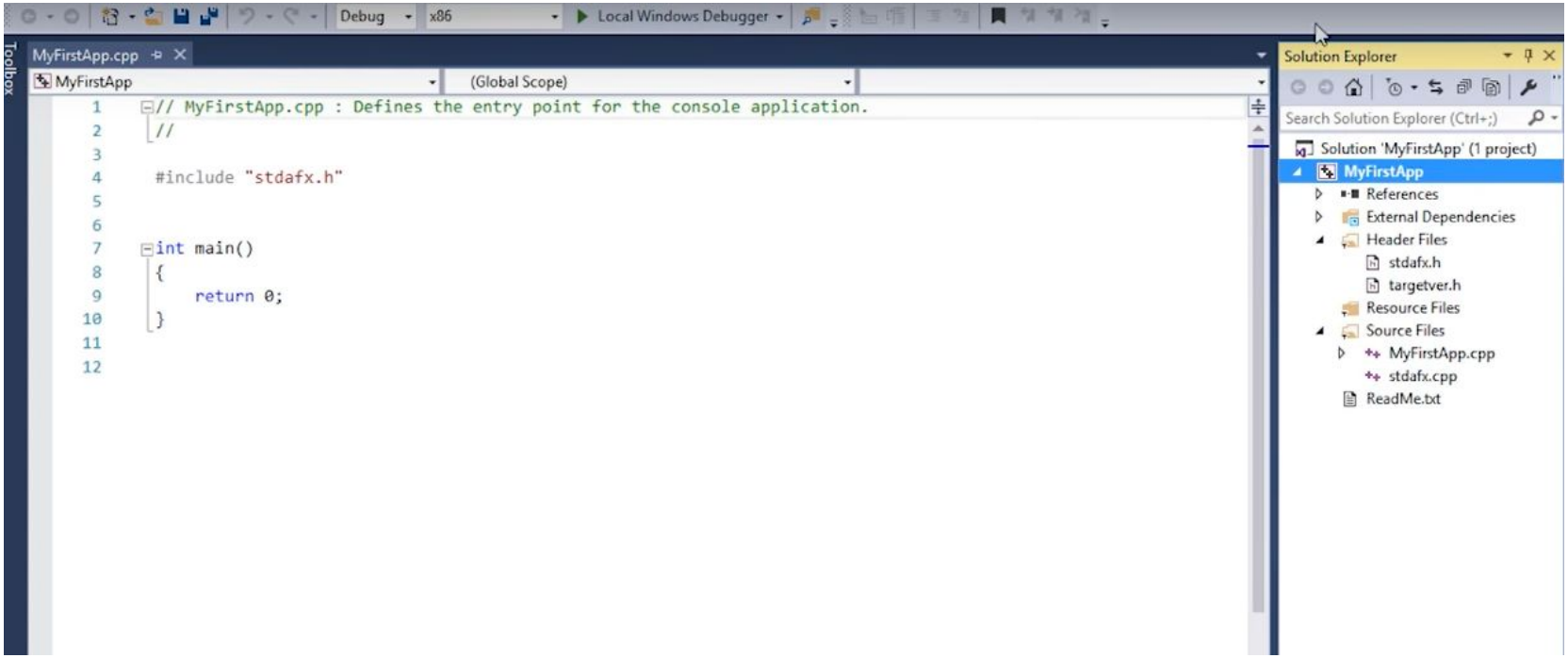
- Каждая подпрограмма имеет структуру, подобную функции main();
- Каждая программа содержит одну или несколько функций;
- Каждая функция содержит 4 основных элемента:

1. тип возвращаемого значения; Int
2. имя функции; main()
3. список параметров, -
   заключённый в круглые скобки
4. тело функции

{return 0;} эта строка значит "вернуть операционной системе в качестве сигнала об успешном завершении программы значение 0".

- **Структура программ** это разметка рабочей области (области кода) с целью определения основных блоков программ и синтаксиса.

- Структура программ несколько отличается в зависимости от среды программирования.


Debug
x86
Local Windows Debugger
Toolbox
MyFirstApp.cpp
MyFirstApp
(Global Scope)
1 // MyFirstApp.cpp : Defines the entry point for the console application.
2 //
3
4 #include "stdafx.h"
5
6
7 int main()
8 {
9 return 0;
10 }
11
12
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MyFirstApp' (1 project)
MyFirstApp
References
External Dependencies
Header Files
stdafx.h
targetver.h
Resource Files
Source Files
MyFirstApp.cpp
stdafx.cpp
ReadMe.txt

Программа на языке C++ состоит из:

1. директив препроцессора,
2. указаний компилятору,
3. объявлений переменных и/или констант,
4. объявлений и определений функций.

Препроцессор — это компьютерная программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы (например, компилятора).

- **Директива препроцессора** – это инструкция, которая включает в текст программы файл, содержащий описание множества функций, что позволяет правильно компилировать программу.

Это важно

- все директивы препроцессора начинаются со знака **#**;
- после директивы препроцессора точка с запятой **не ставится**.

Синтаксис подключения заголовочных файлов:

Директива **#include** позволяет включать в текст программы указанный файл.

Имя файла может быть указано двумя способами:

#include <some_file.h>
#include "my_file.h”

- Если файл является стандартной библиотекой и находится в папке компилятора, он заключается в угловые скобки **<>**.
- Если файл находится в текущем каталоге проекта, он указывается в кавычках **""**.

Заголовочные файлы

Стандартная Библиотека — коллекция классов и функций, написанных на базовом языке.

Основные заголовочные файлы:

- **iostream** – потоки ввода/вывода
- **fstream** – файловые потоки
- **sstream** – строковые потоки

Пространства имен (namespace)

Директива **using** открывает доступ к пространству имен (англ. namespace) **std**, в котором определяются средства стандартной библиотеки языка C++.

- **using namespace std**
- **using namespace standart** - использование имен стандартных(анг)

Пространства имен (namespace)

Пространство имен (*namespace*) — окружение, созданное для логической группировки уникальных имен.

- Необходимо чтобы избежать конфликтов имен идентификаторов.
- Функциональные особенности стандартной библиотеки объявляются внутри пространства имен **std**.

Вызов объекта: **std :: *имя объекта*;**

Организация консольного – ввода/вывода данных (т.е. в режиме чёрного экрана)

#include <iostream>; //здесь подключаем все необходимые препроцессорные директивы

using namespace stg; //директива означает.что все определённые ниже имена будут отнесся к пространству имён std

Int main() // Имя функции, которая не содержит параметров и должна возвращать значение типа *Int*

{Int a; //объявление переменную типа Int - целый тип

cout <<"введите целое числао"<<endl; //оператор вывода данных на экран

,

<< - операция помещения данных в выходной поток;

endl - манипулятор, переводит сообщение на новую сточку.

cin >>a ; //оператор ввода данных с клавиатуры,

>> - операция для извлечения данных из выходного потока, читает значения из cin и сохранения их в переменных.

cout<<a; //*оператор вывода*

return 0;} //*оператор вывода*

Пример пространства имен

std	standart

#include <iostream>
using namespace std;
 	
int main()
{
setlocale(LC_CTYPE, "rus");
int a,b,c;
 	
cout << “Введи данные\n";
cin >>a>>b>>c;
 	
system("pause");
return 0;
}
	#include <iostream>
	
	int main()
	{
	setlocale(LC_CTYPE, "rus");
	int a,b,c;
	
	std:: cout << “Введи данные\n";
	std:: cin >>a>>b>>c;
	
	system("pause");
	return 0;
	}

Функция main()

- Выполнение программы начинается со специальной стартовой функции main.
- В момент запуска программы, управление передается данной функции.
- Функция **main** обязательно должна быть определена в одном из модулей программы. Модуль, содержащий функцию main принято называть главным модулем.

Функция main()

Стандарт предусматривает два формата функции:

//без параметров

тип main( ){/* … */}

Функция main()

- //с двумя параметрами
- **тип main(int argc, char* argv[]){/* … */}**
- Если программу запускать через командную строку, то существует возможность передать какую-либо информацию этой программе.
- Параметр argc имеет тип int, и содержит количество параметров, передаваемых в функцию main.
  Причем argc всегда не меньше 1, даже когда функции main не передается никакой информации, так как первым параметром считается имя приложения.
- Параметр argv[] представляет собой массив указателей на строки. Через командную строку можно передать только данные строкового типа.

Функция main()

Функция **main** может возвращать определенное значение, или не возвращать ничего.

- Если функция не возвращает никакого значения, то она должна иметь тип **void** (такие функции иногда называют процедурами)
- Функция может возвращать значение, тип которого в большинстве случаев аналогично типу самой функции.

```
int main()
{
        ...
}
```

или

```
void main()
{
        ...
}
```

Структура функции

заголовок

```
int main(void)
```

```
{
cout << “Hellow Word \n”;
return(0);
}
```

Тело функции

Заголовок функции

тип значения которое возвращает функция

В нашем случае это int.
То есть, когда функция main закончит свою работу, она должна вернуть в программу которая её вызвала, какое-то **целое значение**.

Если не нужно чтобы программа возвращала какое-то значение, то пишем тип **void**.
Если бы функция main не должна была бы ничего возвращать, то её заголовок выглядел бы так.

void main(void)

Структура программы на C++

#include <имя библиотеки 1> #include <имя библиотеки 2>	**Заголовочные файлы (подключение библиотек)**
// прототипы функций (заголовки)	**Объявление функций**
// глобальные идентификаторы (типы, переменные и т.д.)	**Объявление глобальных идентификаторов**
int main() { *// описание переменных* *// раздел операторов* **}**	**Главная функция программы**
// реализация функций	**Реализация объявленных функций**

- Для использования библиотеки iostream в программе необходимо включить заголовочный файл
- #include <iostream>
- Операции ввода/вывода выполняются с помощью классов istream (потоковый ввод) и ostream (потоковый вывод). Третий класс, iostream, является производным от них и поддерживает двунаправленный ввод/вывод. Для удобства в библиотеке определены три стандартных объекта-потока:
- cin – объект класса istream, соответствующий стандартному вводу. В общем случае он позволяет читать данные с терминала пользователя;

- cout – объект класса ostream, соответствующий стандартному выводу. В общем случае он позволяет выводить данные на терминал пользователя;
- cerr – объект класса ostream, соответствующий стандартному выводу для ошибок. В этот поток мы направляем сообщения об ошибках программы.

- Вывод осуществляется, как правило, с помощью перегруженного оператора сдвига влево (<<), а ввод – с помощью оператора сдвига вправо (>>)

```
#include <iostream>
#include <string>
int main()
{
   string in_string;
   // вывести литерал на терминал пользователя
   cout << "Введите свое имя, пожалуйста: ";
   // прочитать ответ пользователя в in_string
   cin >>  in_string;
   if ( in_string.empty() )
      // вывести сообщение об ошибке на терминал пользователя
      cerr << "ошибка: введенная строка пуста!\n";
   else cout << "Привет, " << in_string << "!\n";
}
```

- Назначение операторов легче запомнить, если считать, что каждый "указывает" в сторону перемещения данных. Например,

>> x перемещает данные в x

<< x перемещает данные из x

Строки в языке C++ (класс string)

- **Основы работы со строками в C++**
- Строки можно объявлять и одновременно присваивать им значения:
- string S1, S2 = "Hello";
- Строка S1 будет пустой, строка S2 будет состоять из 5 символов

Объявление переменных

- **Глобальные переменные** описываются вне функций и действуют от конца описания до конца файла.
- **Локальная переменная** описывается внутри функции и действует от конца описания до конца функции.

Стандартные типы данных

- Целые типы данных – **short, int, long** и спецификаторы ***(signed,unsigned);***
- Вещественные типы – **float, double, long double;**
- Символьные типы – **char, wchar_t;**
- Логический тип – **bool**, принимающий значения (true-истина, false-ложь);

Константы (const)

a=b+2,5 //неименованная константа;

'1L' - целочисленный литерал (тип long);

"8" – целочисл.литерал (тип Int);

'f', – символьный литерал, *'\n'*-конец строки

Формат описания именованной константы:

[<класс памяти>]const <тип> <имя именован-ой константы>=<выражение>;

const int l= - 124;

const floak k1=2,345, k=1/k1

Класс памяти- это спецификатор, определяющий время жизни и область видимости данного объекта.

Переменные

Формат описания переменных:
[<класс памяти>]<тип><имя>[=<выражение> | (<выражение>)];

Пример:

```
int I,j;
double x;
```

Значение переменных должно быть определено с помощью:

1. оператора присваивания:
```
int a;  //описание переменной
int= a;  //опред.значения.переменной
```

2. оператора ввода:
```
int a;  //описание переменной
cin>>a;  // оператор ввода данных с клавиатуры
```
Определитеь значения переменной

3. инициализация – Определитеь значения переменной на этом этапе описания.
```
int i=100   //инициализация копией
int i (100);  // прямая инициализация
```

Задание 1

- Напишите программу, которая складывает два числа и отображает результат на выходе.

Задание 2

- Напишите программу, которая вводит два числа через командный интерфейс, затем складывает эти два числа и отображает результат на выходе.

Операции. Унарные операции

- Операции увеличения (декремента) и уменьшения (инкремента)

  на 1(++ и --); записываются в двух формах:

  Префиксия – операция записывается перед операндом и увеличивает свой операнд на 1 и возвращает изменённоё значений как результат

  Постфиксия – операция записывается после операнда, уменьшает свой операнд на 1 и возвр.изменённое знач.как результат.

Пример:

```
#include <iosteram>
int main()
using namespace std;
{ int x=3, y=4;
  cout <<++x<<“\t”<<--y<<endl;
  cout <<x++<<“\t”<<y--<<endl;
  cout <<x<<“\t”<<y<<endl;
  return 0;}
```

Операции отрицания (-,!)

- *(-) - унарный минус – изменяет знак операнда целого или вещественного типа на противоположный;*

- *(!) – логическое отрицание, даёт в результате значение 0(ложь), если операнд отличен от 0(истина),если равен операнд 0 (ложь);*

- *тип операнда может быть любой.*

Пример:

```
#include <iosteram>
int main()
using nanespace std;
{ int x=3, y=0;
    bool f=false, v=true;
  cout <<-x<<“\t”<<!y<<endl;
  cout <<-y<<“\t”<<!y<<endl;
  cout <<v<<“\t”<<!v<<endl;
 return 0;}
```

Бинарные операции

- *Арифметические операции: умножение.(*), деление.(/), остаток от деления.(%); слож.(+), вычит.(-)*

Рассмотрим операции деления и остаток от деления:

```
#include <iosteram>
 using nanespace std;
 int main()
  { cout <<100/24<<“\t”<<100/24<<endl;
     cout <<100/21<<“\t”<<100,0/24<<endl;
     cout <<21%3<<“\t”<<21%6<<“-21%8”<<endl;
    return 0;}
```

- *Операции отрицания (-,!) унарный минус – изменяет знак операнда целого или вещест-го типа на противоположный.*
- *Операции отношения: (<, <=, >, >=, == !=), меньше, меньше или равно, больше, больше или равно, равно, не равно, не равно соответственно).*

Результатом операций являются значения true, false.

Логические операции (&& и ||)

*И (&&) - возвращает значение истина тогда и только тогда, когда оба операнда принимают значение истина, в п*ротивном случае операция возращ.знач.ложь.

ИЛИ || - возвращает знач.истина тогда и.т.тогда, когда хотя бы один операнд принимает значение истина, в противном случае –ложь

- *логические операции выполняются слева направо;*
- *приоритет операции && выше ||.*

Пример:

```
#include <iosteram>
using namespace std;
int main()
  {  cout <<‘x\t y\t &&\t||’endl;
   cout <<“0\t 1\t”<<(0 && 1)<<‘\t’<<(0||1)endl;
   cout <<‘0\t 1\t’<<(0 && 1)<<\t’<< (0||1)endl;
   cout <<‘1\t 0\t’<<(1 && 0)<<‘\t’<<(1||0)endl;
   cout <<‘1\t 1\t’<<(1 && 1)<<‘\t’<<(1||1)endl;
   return 0;}
```

Операции присваивания

- формат операция простого присваивания (=):
- опреанд_1 = операнд_2

пример: a=b=c=100, это выражение выполняется справа налево, результатом выполнения c=100, является число 100, которое затем присвоиться переменной b, потом a.

- *Сложные операции присваивания*:
- (*=) – умножение с присв-ем,
- (/=) - деление с присв-ем
- (%=)- остаток от деления с присв-ем,
- (+=) –сложение с присв.,
- (-=) - вычит.с присв.
- *пример:* к операнду _1 прибавляется операнд_2 и результат записывается в операнд_2
- *т.е.* c = c + a, тогда компактная запись c += a

Тернарная операция

- *Условная операция (? :)*
- Формат условной операции: операнд_1 ? операнд_2 ? : операнд_3
- Операнд_1 это логическое или арифметич-ое выражение;
- Оценивается с точки зрения эквивалентности константам true и false;
- Если результат вычисления операнда_1 равен true,то результат условной операции будет значение операнда_2, иначе операнда_3;
- Тип может различаться;
- Условная операция является сокращ. формой услов-го оператора if;

Пример:

```
#include <iosteram>
  int main()
  using nanespace std;
  { int x, y,max;
    cin >>x>>y;
  (x>y)? cout <<x: cout<<y<<endl;     //1   нахождение наибольшего зна-
  max=(x>y)? x:y;                     //2   чения для двух целых чисел;
  cout<<max<<endl;
  return 0;}
```

Результат:

для x=11 и y=9
11
11