Программа, которая печатает Hello, world! на языке С. Потом разберем подробно что делает каждая строка.

#include <stdio.h> // использовать стандартные функции ввода-вывода

int main() // здесь начинается программа

{

printf("Hello, world!\n"); // печать строки Hello, world!

return 0; // конец программы

}

**Программирование** - это запись решения задачи на специальном формальном языке, который понимает компьютер, например Си. Вы переводите решение задачи с русского или языка, на котором вы думаете, на язык Си.

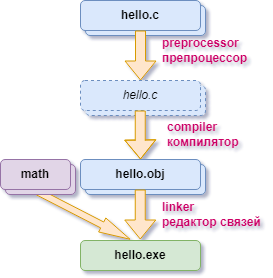
**Ключевые слова языка Си**

auto continue else for long signed switch void

break default enum goto register sizeof typedef while

case do extern if return static union

char double float int short struct unsigned



Рассмотрим, как из программы на языке Си, которая написана в файле hello.c, получается файл hello.exe.

Файл hello.exe можно запустить и он напечатает Hello, world!

### Preprocessor (препроцессор)

**Препроцессор** (preprocessor) в файле hello.c выполняет свои директивы (команды).

* Команды препроцессора - это поиск текста и замена его на другой текст.
* #include <stdio.h> - команда препроцессора include и ее аргумент <stdio.h>.
  + Эта строка заменяется на содержимое файла stdio.h
* Команды препроцессора начинаются с #.
* Полученный текст можно читать (с трудом). Результат не хранят и обрабатывают дальше.

### Compiler (компилятор)

* **Компилятор** (compiler) переводит программу в **объектный модуль** (человеку трудно понять его текст). Обычно это файл с расширением .o на UNIX и .obj на Windows. Иногда данные не сохраняют в файл, а обрабатывают дальше.

Большая программа может быть написана в нескольких .c файлах. Из одного .c получают один объектный модуль.

### Linker (редактор связей)

* **Редактор связей** (linker) связывает в единый **исполняемый модуль** все объектные модули и необходимые библиотеки.

Исполняемый модуль содержит машинные команды. Набор машинных команд зависит от архитектуры компьютера и операционной системы. hello.exe созданный в командной строке Unix не будет исполняться в командной строке Windows и наоборот.

**Абсолютный путь**

* В Linux начинается с /
  + Примеры абсолютного пути в Linux:

/home/taty/conline/1int/hello.c

/mnt/c/work/stud

* В Windows с \ или с имени диска
  + Примеры абсолютного пути в Windows:

C:\work\stud\conline\1int\hello.c

\work\stud\conline\1int\hello.c

**Относительный путь**

Задается относительно текущей директории (мы в ней находимся).

* .. - родительская директория (на 1 директорию вверх)
* . - текущая директория

Примеры относительного пути в Linux:

hello.c

work

conline/1int/hello.c

./a.out

..

../../img

Примеры относительного пути в Windows:

hello.c

work

conline\1int\hello.c

.\a.exe

..

..\..\img

## Команды

| **Команда** | **Что делает** | **Мнемоника** | **Сокращение от** |
| --- | --- | --- | --- |
| pwd | печатает путь к текущей директории | где я? | **p**rint **w**orking **d**irectory |
| ls | печатает содержимое текущей директории | что тут? | **l**i**s**t |
| cd директория | переходит в указанную директорию | иди на | **c**hange **d**irectory |
| cat файл | выводит на экран содержимое файла | покажи |  |

Рекомендуем на первой странице вашей тетради записывать команды Linux

// начало программы - началось

#include <stdio.h>

int main()

{

// начало программы - закончилось

// тут пишем код

printf("Hello, world!"); // печать строки Hello, world!

// конец программы - началось

return 0;

}

// конец программы – закончилось

// выполнение программы всегда начинается с функции main { // начало тела функции main

Разберем по строкам

#include <stdio.h>

Вставляем в код необходимую информацию для подключения стандартной библиотеки ввода-вывода. В этой библиотеке реализована функция printf, которую мы используем позже.\

Int main()

{

какой-то код

}

Выполнение программы в языке Си всегда начинается с функции main.

Стандартная функция printf печатает строковую константу "Hello, world!", строковые константы пишем в двойных кавычках.

return 0;

Код возврата из функции. Код возврата из функции main - это код возврата из всей программы. Не обязательный элемент. Но в курсе его писать обязательно, потому что без return 0; проверяющая система может посчитать, что ваша программа решена не верно. Так как проверка программ начинается к кода возврата, если он не 0, то считается, что программа закончилась с ошибкой и проверять её дальше бессмысленно.

Обратите внимание, в программе три разных типа скобок:

* <> в первой строке,
* () у функций main и printf
* {} один раз

**#include <stdio.h>**

* #include <stdio.h> - включить в программу информацию для чтения и печати данных.
  + # - с этого знака "диез" начинаются все команды препроцессора
  + команда препроцессора include (англ. "включить")
  + скобки <>
  + включаем мы файл, в котором описан **st**an**d**ard **i**nput/**o**utput (стандартный ввод/вывод)
  + не забываем, расширение этого файла .h

Сделаем ошибки:

* забудем #
* неверно напишем include
* неверно напишем stdio.h
* напишем другие скобки вместо <>

Заметьте, если написать без пробелов #include<stdio.h>, то выражение компилируется без ошибок и работает правильно.

**int main() { }**

* Программа всегда начинает выполнение с функции **main** (англ. "главная").
* Перед названием функции пишем int от слова **int**eger.
* Дальше ()
* Дальше {}. Внутри этих скобок мы будем менять код на этом уроке.

Сделаем ошибки:

* неправильно напишем main
* Забудем обе или одну из ()
* Забудем {
* Забудем }
* Напишем без пробела intmain

Заметьте, количество пробелов или переводов на новую строку в этой конструкции не приводит к ошибкам.

**printf("Hello, world!");**

* **print+f**ormat - печать по формату, стандартная функция языка С для вывода на печать,
* printf функция, ее аргументы надо писать в (), как в математике,
* текст пишем в "
* в конце пишем ;

В русском языке в конце каждого предложения ставим точку. В языке Си в конце каждой мысли ставим ;

Заметьте, что ; мы забыли на одной строке, а компилятор показывает, что ошибка на следующей, нужен ; **перед** указанным кодом.

**return 0;**

* **return** (англ. "вернуть") - ключевое слово языка Си, запомните его.
* Дальше любое целое число,
* ; в конце мысли.

Заметьте, можно написать вместо 0 любое целое число и программа будет по-прежнему компилироваться и работать.

| **Escape последовательность** | **Значение** |
| --- | --- |
| \n | новая строка (курсор на первую позицию следующей строки) |
| \r | возврат каретки (курсор на первую позицию этой строки) |
| \t | табуляция |
| \' | ' |
| \" | " |
| \\ | \ |

#include <stdio.h> // использовать стандартные функции ввода-вывода

int main() // начало программы

{

printf("Hello!""\n"); // печать Hello!

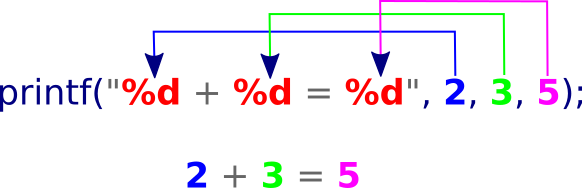
printf("Good bye!"); // печать Good bye!

return 0; // конец программы

}

Hello!

Good bye!



#include <stdio.h>

int main()

{

// тут пишем код, который вычисляет 2+3

printf("2 + 3 = %d\n", 2+3); // 2 + 3 = 5

printf("%d + %d = %d\n", 2, 3, 2+3); // 2 + 3 = 5

return 0;

}

| **format placeholder** | **заменяется на** |
| --- | --- |
| %d | целое десятичное число |
| %f | дробное число (десятичная дробь) |

printf("Ускорение свободного падения g = %f\n", 9.81); // Ускорение свободного падения g = 9.81

Электричка идет 2 часа 20 минут. Сколько это в минутах? Сколько это в секундах? Напишем программу:

#include <stdio.h>

int main()

{

// тут пишем код

printf("Время в пути в минутах %d минут\n", 2\*60 + 20);

printf("Время в пути в секундах %d секунд\n", (2\*60 + 20)\*60);

return 0;

}

Она напечатает:

Время в пути в минутах 140 минут

Время в пути в секундах 8400 секунд

Сначала нужно **объявить переменную**, потом можно ее использовать.

* Тип **int** от английского слова integer (целое число).
* Тип **float** от floating point number (число с плавающей точкой) -

int x; // объявили переменную с именем х типа int

float g; // объявили переменную с именем g типа float

Можно объявить сразу несколько переменных **одного типа**:

int x, y, n; // тип и переменные через запятую

#include <stdio.h>

int main()

{

// объявим переменные min и sec

int min; // время в минутах

int sec; // время в секундах

// используем переменные для вычислений

min = 2 \* 60 + 20; // в переменную min записали число 140

sec = min \* 60; // прочитали из переменной min число 140, вычислили 140\*60 и 8400 записали в переменную sec

// используем переменные для печати результатов

// вместо %d подставили число, которое хранится в переменной min

printf("Время в пути в минутах %d минут\n", min); // вместо %d подставили число, которое хранится в переменной sec

printf("Время в пути в секундах %d секунд\n", sec);

return 0;

}

int x, y; // сначала объявим переменные

scanf("%d", &x); // целое число записать в переменную х

scanf("%d", &y); // целое число записать в переменную y

или так:

scanf("%d%d", &x, &y); // 2 целых числа записать в переменные x и y

Эксперимент длится h часа m минут. Сколько это в минутах? Сколько это в секундах?

* Дано: два целых числа через пробел (часы и минуты).
* Найти: два целых числа, по одному числу на строке. Всего минут. Всего секунд.

#include <stdio.h>

int main()

{

// объявим переменные

int h; // часы

int m; // минуты

// прочитаем входные данные

scanf("%d", &h);

scanf("%d", &m);

// объявим переменные для результатов и посчитаем результат

// по формуле 60h + m,это сколько минут

int minres = h \* 60 + m;

printf("%d\n", minres);

// секунд в 60 раз больше

int secres = minres \* 60;

printf("%d\n", secres);

return 0;

}

* **float** - тип для работы с вещественными числами, например, с десятичной точкой 3.14.
* **%f** - формат для чтения и печати вещественных чисел (значений переменных типа float).
* / - оператор деления в языке Си.

В примере вычисляют и печатают индекс массы тела человека (ИМТ) по формуле Кетле�=�/ℎ2*k*=*m*/*h*2где �*m* - масса в кг, ℎ*h* - рост в метрах.

#include <stdio.h>

int main()

{

float m; // масса, кг

float h; // рост, м

float imt; // индекс массы тела (формула Кетле)

// читаем входные данные

scanf("%f%f", &m, &h);

// вычисления

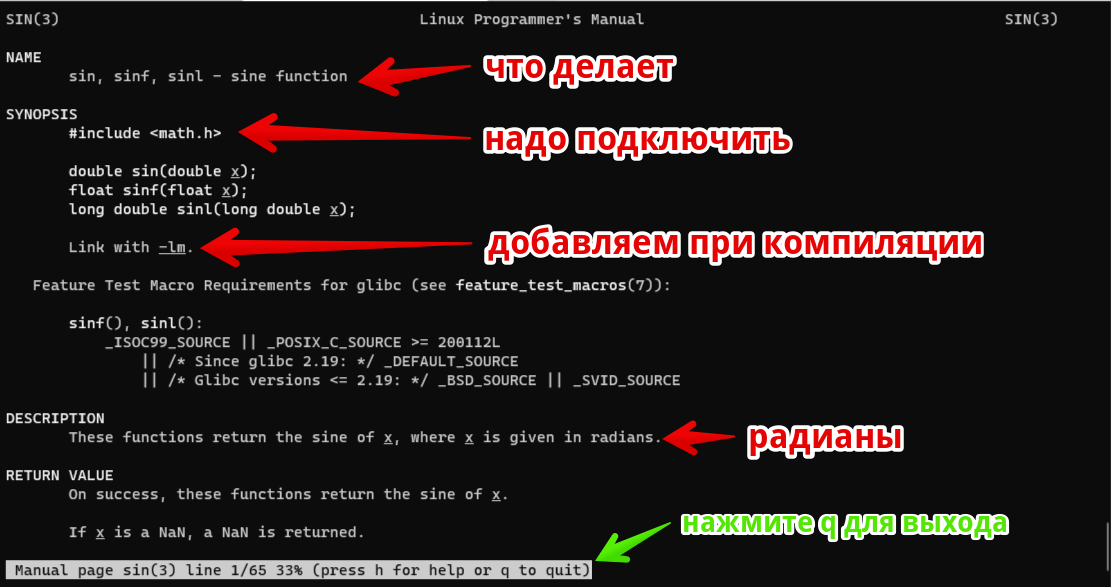
imt = m / (h \* h);

printf("%f\n", imt);

return 0;

}

**ФУНКЦИИ**



функции **sqrt**. Она вычисляет квадратный корень из числа.

**if (условие) оператор // выполняется только когда условие истинно**

int main()

{

int n, // пассажиров на берегу

k, // пассажиров за один рейс может перевезти лодка

reisov, // количество рейсов

ostalos; // пассажиров в неполной лодке

scanf("%d%d", &n, &k); // читаем n и k

reisov = n / k; // количество рейсов, когда лодка полная

ostalos = n % k; // сколько человек осталось после этого на берегу

if (ostalos > 0) { // если на берегу остались люди

reisov += 1; // добавить еще 1 рейс с неполной лодкой

printf("Нужен дополнительный рейс\n");

}

printf ("%d\n", reisov); // напечатать количество рейсов

return 0;

}

**Части if**

* **if** - ключевое слово.
* *условие* пишем в круглых скобках ( )
* команды пишем с отступом относительно ключевого слова if;
* отступ в 1 табуляцию для каждого дополнительного уровня вложенности;
* отступ одинаковый для одинакового уровня вложенности;
* если нужно написать несколько *оператор* , то объединяем их в один **блочный оператор**, то есть пишем все операторы внутри фигурных скобок { и }
* единственный *оператор* можно написать без { }, а можно поставить вокруг него { }
* первое время пишите { } всегда, это поможет избежать ошибок, когда вы добавляете к операторам еще код, например, отладочную печать.
* закрывающую фигурную скобку пишите под if (на прежнем уровне отступов)

| **В математике** | **В языке С** | **По-русски** |
| --- | --- | --- |
| > | > | больше |
| ≥ | >= | больше или равно |
| < | < | меньше |
| ≤ | <= | меньше или равно |
| = | == | равно |
| ≠ | != | не равно |

## if .. else

if (x % 2 == 0)

printf("четное\n");

else

printf("нечетное\n");

**\n**    - новая строка  
**\t**    - горизонтальная табуляция. Сдвигает выводимые данные вправо на ближайшую позицию табуляции. Обычно используется для выравнивания текста внутри строки.  
**\'**    - вывод символа '  
**\"**    - вывод символа "  
**\\**    - вывод символа \  
**\?**    - вывода символа

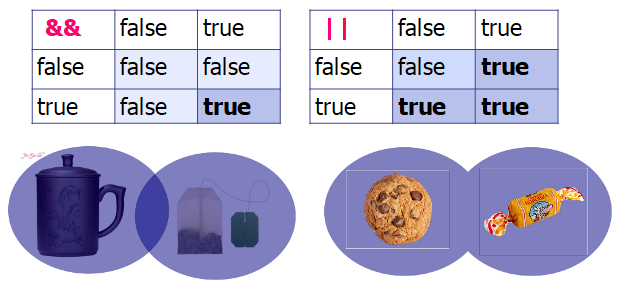
Для использования математических функций нужно подключить заголовочный файл **math.h**. В ней определено много различных функций, но мы пока рассмотрим следующие:

Некоторые математические функции

**fabs(x)** модуль числа x  
**sqrt(x)** квадратный корень из числа x  
**sin(x)** синус числа x (х в радианах)  
**cos(x)** косинус числа x (х в радианах)  
**pow(x, y)** вычисление xy  
**exp(x)** вычисление ex  
**log(x)** натуральный логарифм числа x  
**log10(x)** десятичный логарифм числа x

* && - логическое **И** (AND, конъюнкция)
* || - логическое **ИЛИ** (OR, дизъюнкция)
* ! - логическое отрицание (NOT)

**Внимание: Два && и два || - это логические И и ИЛИ**. Один & и | - это **побитовые** И и ИЛИ



* **!** - отрицание имеет самый высокий приоритет, как унарный минус.
* **&&** - как умножение.
* **||** - как сложение.

Чтобы программисты не теряли ни инициализацию управляющей переменной цикла i = 0, ни ее изменение в цикле i++, придумали цикл **for**.

for(выр1; выр2; выр3)

тело цикла

* *выр1* - инициализация, выполняется **один** раз в начале цикла.
* *выр2* - условие *продолжения* цикла.
* *выр3* - выполняется каждый раз после тела цикла, *перед* проверкой условия продолжения цикла.

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