**Ворожцов Артём Викторович**
*Кандидат физико-математических наук, преподаватель кафедры информатики Московского физико-технического института (МФТИ), тренер сборной команды МФТИ по программированию.*

**Королёв Дмитрий Николаевич**
*преподаватель информатики, лицей №1511 при Московском инженерно-физическом институте (МИФИ)*

**Винокуров Никита Алексеевич**
*преподаватель кафедры информатики Московского физико-технического института (МФТИ)*

# Язык Си в примерах

## Вики – подход в образовании

Журнал «Потенциал» открыл раздел на сайте «Викиучебника» (http://ru.wikibooks.org). В этом разделе планируется публиковать избранные статьи из печатного номера, а также много других статей.

Викиучебник – это братский сайт Википедии (http://wikipedia.org) – коллективно создаваемой интерент-энциклопедии. Любой желающий может поучаствовать в написании и редактировании статей этих сайтов. Викиучебник содержит различные обучающие материалы и материалы справочного характера. Сегодня там можно найти учебники по объектно-ориентированному программированию на Си++ и основам функционального программирования, учебник «Теория музыки для математиков» и много других (не только естественно-научного направления) статей.

Идея «Вики» (wiki) заключается в предоставлении возможности каждому пользователю интернета редактировать и добавлять страницы на сайте[1]. Эта простая идея, оказывается, содержит в себе большой потенциал. Она позволяет организовываться сообществам единомышленников и коллективно создавать информационные ресурсы в интернете. Многие не верили и не допускали возможности, что идея «Вики» найдёт широкое применение в интернете. В первую очередь, с трудом верилось, что коллектив авторов, раздираемый внутренними противоречиями и отличающимися друг от друга мнениями, может создать что-то целостное и качественное, не имея никакой материальной мотивации.

[1] Одним из первых общедоступных вики-сайтов был сайт http://wiki.tck.tk – сайт, посвящённый языку Tcl (об этом языке будет одна из наших следующих статей). Сегодня вики-разделы есть на огромном количестве сайтов. Такие разделы есть на институтских сайтах, на сайтах, посвящённых различным технологиям.

Но, как показало время, самоорганизация множества коллектива авторов возможна. И это просто замечательно! Человечество созрело до такого сотояния, когда вандализм и вредительство не в моде. Человечество объединилось в едином порыве уменьшения количества информационного хаоса и беспорядка. Именно вики-подход позволяет выработать действительно объективные и полные энциклопедические статьи и точные, красиво оформленные учебные материалы, не содержащие серьёзных ошибок и неточных формулировок.

Более того, сегодня уже ясно, что написать настолько отточенный и адекватный учебный материал, какой можно найти в вики-учебниках, одному человеку практически невозможно.

В настоящий момент английская версия Википедии содержит более 1000000 статей, а русская – 61 000 статей. Количество статей в русской версии постоянно увеличивается – в день появляется порядка 1000 новых статей. Викиучебник развивается не настолько быстро, но скорость роста постоянно увеличивается. Сегодня в Викиучебнике можно найти книги:

- Кулинарная книга
- Объектно-ориентированное программирование
  - * Си++
  - * Smalltalk в примерах
- Основы функционального программирования
  - * Функциональные парсеры
- Язык Си в примерах
- Московская олимпиада по информатике
- Математика случая
- Теория музыки для математиков

и много других.

Теперь у сайта нашего журнала также есть вики-раздел – этот раздел является частью Викиучебника и читатели могут писать отзывы к статьям журнала, исправлять неточности в них и даже сами писать статьи, которые будут рассмотрены редакцией журнала и могут войти в печатную версию.

Коллектив авторов журнала Винокуров Н.А., Ворожцов А.В. (преподаватели кафедры информатики Московского физико-технического института), и Королёв Д.С. (тренер команд Московского инженерно-физического института по программированию) начали разрабатывать на сайте Викиучебника раздел «Язык Си в примерах».

Целью данного учебника является:

- предоставление простых наглядных материалов, демонстрирующих базовые конструкции языка Си и обучающих новичка правильному стилю;
- создание коллекции кодов решения простых типичных алгоритмических задач на языке Си.

Предполагается, что материал будет касаться достаточно простых вещей, для которых существует небольшое количество канонических, устоявшихся решений.

Каждый пример включает в себя:

- формулировку простой, ясной и достаточно живой программистской задачи;
- короткий, почти не требующий комментариев, код;
- пояснение особенно сложных для начинающего программиста моментов.

Авторы постараются снабдить каждый пример указанием его сложности и ссылками на примеры, которые рекомендуется предварительно изучить.

Содержание учебника «Язык Си в примерах»

На первых шагах изучения нового языка программирования преподаватель вынужден давать искусственные упрощённые задачи. Этот учебник также может содержать ряд таких искусственных задач. Несмотря на то, что данный учебник представляет собой вводный курс, авторы стремились уменьшить количество таких искусственных задач и давать (иногда в ознакомительном стиле) примеры реальных жизненных задач, а также рассказывать о сопутствующих языку Си технологиях (bison, make utils, ...)

Здесь мы опубликовали первые три статьи из учебника «Язык Си в примерах».



## Компиляция программ

Программа на языке Си – это один или несколько текстовых файлов, которые также называются исходными. Исполнить исходные файлы нельзя, их необходимо *скомпилировать*, т.е. создать исполняемый файл, содержащий в себе инструкции процессора и пригодный для запуска на компьютере.

Процесс преобразования исходных файлов в исполняемый файл называется *компиляцией*. Если ваша программа состоит из одного исходного файла hello.c, то для его компиляции компилятором GNU C достаточно выполнить команду:

```
bash$ gcc hello.c -o hello
```

В результате получится файл hello, имя которого мы указали в опции -o. Этот файл является исполняемым и его можно запускать (**exe**cute) при помощи команды:

```
bash$ ./hello
```

Пара символов ./ перед hello означает «искать исполняемый файл `hello` в текущей директории». Строчка

```
bash$ gcc xxx.c yyy.c -o zzz -I./common -I.. -lm
```

соответствует команде: "скомпилировать файлы xxx.c и yyy.c в программу zzz; заголовочные файлы находятся в директориях ./common и ..; подключить библиотеку libm".

Библиотека libm (подключаемая с помощью опции `-lm`) содержит откомпилированные математические функции, которые объявляются в заголовочном файле `math.h`. Если вы используете функции из этой библиотеки (такие как `log, sin, cos, exp`), то не забывайте подключать её при компиляции.

Подробную информацию об опциях компилятора gcc можно получить, если набрать

```
bash$ man gcc
```

или

```
bash$ info gcc
```

## Простейшая программа «Hello World»

Первая программа, которую мы рассмотрим, — это «Hello World» — программа, которая выведет на экран строчку «Hello, World!» и закончит своё выполнение.

```
#include <stdio.h>
int main (void) {
```

```
    printf ("Hello, World!\n");
    return 0;
}
```

Посмотрим на неё внимательно. Первая строчка

```
#include<stdio.h>
```

означает «включи файл `stdio.h`». В этом файле определяются функции, связанные с вводом и выводом данных.

Аббревиатура STDIO означает «STanDard Input/Output Library». Буква «h» после точки означает «header», то есть заголовочный файл. В заголовочных файлах описано, какие функции предоставляет соответствующая им библиотека[2].

Далее идёт функция main. Она начинается с объявления

```
int main(void)
```

что значит: «функция с именем main, которая возвращает целое число (число типа int от англ. *integer number*) и у которой нет аргументов (`void`)»

Слово `void` можно переводить как `íè÷òî`. Далее открываются фигурные скобки и идёт описание этой функции, в конце фигурные скобки закрываются. Функция `main` — эта главная функция вашей программы, именно она начинает выполняться, когда ваша программа запускается.

Между фигурными скобками находится *тело функции*, в котором описана последовательность действий, производимых данной функцией — логика функции. Наша функция производит одно единственное действие:

```
printf ("Hello, world!\n");
```

Это действие, в свою очередь, есть вызов функции `printf` из библиотеки `stdio`. В результате выполнения этой функции, на экране печатается текст `Hello, world`!. Обратите внимание на комбинацию `"\n"` — она задаёт специальный символ, который в действительности является командой текстовому терминалу: «перейти на следующую строку». Таких специальных символов несколько, все они записываются с помощью символа \ (символ backslash).

Затем идёт команда return 0;, которая завершает выполнение функции и возвращает значение 0. Функция main должна возвращать 0, если выполнение прошло успешно.

## Учимся складывать

Разнообразные вычисления — моделирование, решение алгебраических и дифференциальных уравнений — это то, для чего и создавались первые компьютеры. Давайте и мы научимся использовать компьютер для вычислений. Начнём со сложения двух чисел.

В нашей программе будут две целочисленные переменные: a и b. Это две ячейки памяти, в которых могут храниться целые числа из определённого диапазона значений (в 32-битной архитектуре от $-2^{31}$ до $2^{31}-1$).

---

[2] В действительности, `#include<...>` есть директива препроцессора, то есть команда, которая выполняется до начала компиляции файла. Смысл этой директивы очень прост и заключается в том, чтобы на место, где указана эта директива, вставить содержимое файла, имя которого указано в угловых скобках. Обычно заголовочные файлы содержат только прототипы функций, то есть просто список функций с указанием аргументов и типа возвращаемого значения.

Переменные объявляются в начале тела фнуции `main` — после открывающей фигурной скобки. Объявление начинается со слова, обозначающего тип переменных, имена которых перечисляются через запятую после обозначения типа.

```
int a, b;
```

В языке Си есть несколько типов данных. Они делятся на две группы: целые типы и типы с плавающей точкой. К первому типу относятся

```
char, short, int, long, unsigned char, unsigned int, unsigned long.
```

Ко второму —

```
float, double.
```

Вот текст программы, складывающей два введённых целых числа:

```
#include <stdio.h>
int main (void) {
    int a, b;
    printf ("Введите два числа: ");
    scanf ("%d%d", &a, &b);
    printf ("%d\n", a + b);
    return 0;
}
```

Функция `scanf`, так же как и `printf`, определена в библиотеке `stdio`. Эта функция считывает данные, которые пользователь (тот, кто запустит вашу программу) вводит с клавиатуры. Слово scan означает «считывать данные», а `print` — «печатать данные». Буква «f» в конце соответствует первой букве английского слова «formated», то есть `scanf` и `printf` есть функции для форматированного ввода и вывода данных.

Первый аргумент у функции `scanf` — это `"%d%d"` (то, что стоит между открывающей скобкой и первой запятой). Первый аргумент является описанием формата входных данных, то есть описание типа данных, которые (как мы ожидаем) введёт пользователь. В этой программе мы ожидаем, что пользователь введёт два целых числа.

Символ % служебный, с него начинается описание формата. Обычно после него идёт один или два символа, определяющих тип входных данных. Формат "%d" соответствует целому числу в десятичной системе счисления (decimal integer). Если вы напишете "%x", то функция будет ожидать ввода целого числа, записанного в шестнадцатиричной системе счисления.

Подробнее об спецификациях форматах ввода/вывода можно прочитать в документации (для Unix систем):

```
bash$ man 3 printf
bash$ man 3 scanf
```

Первый аргумент команды man есть номер раздела документации. Помощь по языкам Си/Си++ находится в третьем разделе.

Следует отметить, что для каждого типа данных существует несколько форматов, и наоборот, для разных типов можно использовать один и тот же формат[3].

---

[3] Переменные типа char, short, int можно печатать, используя формат "%d". Также этот формат можно использовать для печати значений указателей (номера ячейки памяти).

Приведённая программа умеет складывать только целые числа. Если вы хотите складывать действительные числа, то эту программу нужно несколько модифицировать. Ниже приведена программа, которая считывает два действительных числа и выводит результат четырёх арифметических операций: сложения, вычитания, умножения и деления. Причём, программа выводит результаты вычислений два раза — сначала в обычном виде, а потом со специальным форматированием. Формат `"%10.3lf"` соответствует выводу числа типа `double`, при котором под запись числа выделяется ровно 10 позиций (если это возможно), а после запятой пишется ровно три знака. Выравнивание происходит по правому краю.

```
/* Программа "Арифметические операции с числами с плавающей точкой" */
#include <stdio.h>
int main () {
   double a, b;
   printf ("Введите два числа: ");
   while(scanf ("%lf%lf", &a, &b) == 2 ) {
      printf ("%lf %lf %lf %lf\n", a + b, a - b, a * b, a / b );
      printf ("a + b=%10.3lf\n a - b=%10.3lf\n a * b=%10.3lf\n a / b=%10.3lf\n",
              a + b, a - b, a * b, a / b );
   }
   return 0;
}
```

В этой программе мы встречаемся с оператором `while`. Конструкция

```
while ( A ) B;
```

означает буквально следующее:

«Пока выполнено условие A делать B.»

или, другими словами,

«Выполнять в цикле B, проверяя перед каждой итерацией, что выполнено условие A.»

В нашем случае A есть

scanf("%lf%lf", &a, &b) == 2.

Что соответствует логическому выражению:

«пользователь ввёл два действительных числа, и они удачно считаны в переменные a и b»

Таким образом, эта программа будет считывать пары чисел и выводить результаты арифметических операций, пока пользователь не введёт что-нибудь непохожее на число.

Цикл `while` закончится тогда, когда функция `scanf` не сможет успешно считать два числа.

Заметьте, что после каждой команды стоит точка с запятой. Одна из самых популярных синтаксических ошибок начинающих программистов – это не ставить точку с запятой в конце команды.