1. Введение в язык программирования C

Владимир Верстов

Б. Керниган, Д. Ритчи. Язык программирования C

Литература

Б. Керниган, Д. Ритчи. Язык программирования С

Интересные факты о книге:

- Самый известный пример из книги - программа “Hello, world!”. С 1978 года большинство книг по языкам программирования продолжают традицию начинать с этой программы
- Стиль форматирования кода из книги используется в исходном коде операционной системы Unix и ядра Linux
- До 1989 года книга де факто являлась стандартом для языка С
- Книга считается классическим примером технической литературы

Hello, World!

```
Вывести слова
  Hello, world!
```

Hello, World!

```c
#include <stdio.h>

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

Компиляция и исполнение

- Компиляция и создание исполняемого файла **a.out**

```
iMac-Vladimir:Desktop vverstov$ cc hello.c
```

- Исполнение **a.out**

```
iMac-Vladimir:Desktop vverstov$ ./a.out
Hello, World!
iMac-Vladimir:Desktop vverstov$
```

- Компиляция и создание исполняемого файла **hello**

```
iMac-Vladimir:Desktop vverstov$ cc hello.c -o hello
```

- Исполнение **hello**

```
iMac-Vladimir:Desktop vverstov$ ./hello
Hello, World!
iMac-Vladimir:Desktop vverstov$
```

Hello, World!

1. Подключение стандартной библиотеки ввода и вывода **stdio**

2. Определение функции **main**

3. Функция **main** вызывает функцию **printf** из библиотеки **stdio** для печати последовательности символов;

4. возврат кода завершения программы в операционную систему

```c
#include <stdio.h>

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

Программы на C

- Состоят из **функций** и **переменных**
- Функции содержат **операторы**
- **Оператор** - команда для выполнения определенной вычислительной операции
- В **переменных** хранятся числа и другие данные, которые используются в вычислительных операциях
- **main** - особая функция, с нее начинает выполнятся программа. Любая программа на C **обязательно содержит** только **одну функцию main**.

Строковая константа

- … - это последовательность символов в двойных кавычках
- Пример: `"Hello, World!\n"`
- **\n** - условное обозначение символа перехода на новую строку
- **\n** - один символ, **управляющая последовательность** (Escape Sequence)
- **Управляющие последовательности** используются для отображения непечатных символов

Переменные и арифметика

```
 0  -17
 20  -6
 40   4
 60  15
 80  26
 100 37
 120 48
 140 60
 160 71
 180 82
 200 93
 220 104
 240 115
 260 126
 280 137
 300 148
```

```c
#include <stdio.h>

int main() {
    int fahr, celsius;
    int lower, upper, step;
    lower = 0;
    upper = 300;
    step = 20;
    fahr = lower;
    while (fahr <= upper) {
        celsius = 5 * (fahr - 32) / 9;
        printf("%3d\t%3d\n", fahr, celsius);
        fahr = fahr + step;
    }
}
```

Объявление переменных

- В языке C все переменные нужно объявить до их использования
- В **объявлении** описываются свойства переменных
- **Объявление** состоит из наименования типа и списка переменных
- Пример:

```
int fahr, celsius;
int lower, upper, step;
```

Типы данных

- **int** - целое число
- **float** - число с плавающей точкой
- **double** - число с плавающей точкой с двойной точностью

Цикл while

- Тело цикла выполняется до тех пор, пока условие истинно

```
while (fahr <= upper) {
    celsius = 5 * (fahr - 32) / 9;
    printf("%3d\t%3d\n", fahr, celsius);
    fahr = fahr + step;
}
```

Формула и присвоение

- Результат целое число - дробная часть отбрасывается

```
celsius = 5 * (fahr - 32) / 9;
```

- Результат вещественное число - float

```
celsius = (5.0 / 9.0) * (fahr - 32);
```

Цикл for

```
  0 -17.8
 20  -6.7
 40   4.4
 60  15.6
 80  26.7
100  37.8
120  48.9
140  60.0
160  71.1
180  82.2
200  93.3
220 104.4
240 115.6
260 126.7
280 137.8
300 148.9
```

```c
#include <stdio.h>

int main() {
    int fahr;
    float celsius;

    for (fahr = 0; fahr <= 300; fahr = fahr + 20) {
        celsius = (5.0 / 9.0) * (fahr - 32);
        printf("%3d %6.1f\n", fahr, celsius);
    }
}
```

Форматированный вывод

- %d - вывести аргумент как десятичное целое число
- %3d - вывести аргумент как десятичное целое в поле с шириной не менее 3 символа
- %6.1f - вывести аргумент как вещественное число в поле шириной не менее 6 символов и с 1 знаком после запятой
- %.1f - вывести аргумент как вещественное число с 1 знаком после запятой

Примеры спецификации printf

- %o - восьмеричное целое число
- %x - шестнадцитиричное число
- %c - символ
- %s - строка
- %% - знак процента

Именованные константы

```c
#include <stdio.h>

#define MAX 300
#define STEP 20
#define MIN 0

int main() {
    int fahr;
    float celsius;

    for (fahr = MIN; fahr <= MAX; fahr = fahr + STEP) {
        celsius = (5.0 / 9.0) * (fahr - 32);
        printf("%3d %6.1f\n", fahr, celsius);
    }
}
```

Директива #define

- #define определяет имя и константу

```
#define ИМЯ текст для подстановки
```

- Имя - последовательность букв и цифр
- Имя принято писать прописными буквами
- Текст для подстановки последовательность любых символов

Символьный ввод и вывод

- Текстовый ввод-вывод, независимо от его источника, выполняется над потоками символов
- Поток символов - это последовательность символов, разбитых на строки
- Каждая строка заканчивается специальным символом конца строки
- Строка может быть пустой или состоять из некоторого количества символов
- Функция **getchar** - считывает следующий символ текстового потока
- Функция **putchar** - печатает 1 символ в текстовый поток

Копирование ввода на вывод

```
чтение символа
while (символ не является признаком конца файла)
  вывод только что прочитанного символа
  чтение символа
```

Копирование ввода на вывод

```c
#include <stdio.h>

int main() {
    int c;
    c = getchar();
    while (c != EOF) {
        putchar(c);
        c = getchar();
    }
}
```

EOF - End of File

- **getchar** - считывает символ
- **putchar** - печатает символ
- **EOF** - символ конца файла (**E**nd **o**f **F**ile)
  - решает проблему определения конца ввода
  - константа из **stdio.h**
  - Обычно **EOF = -1**
- Тип переменной **c** - целое число, чтобы переменная могла гарантированно вместить **EOF**

Копирование ввода на вывод

```c
#include <stdio.h>

int main() {
    int c;
    while ((c = getchar()) != EOF)
        putchar(c);
}
```

Количество символов

```c
#include <stdio.h>

int main() {
    long nc;
    nc = 0;
    while (getchar() != EOF)
        ++nc;
    printf("%ld\n", nc);
}
```

Инкремент

- Инкремент - увеличение числа на единицу
- **++nc;** - инструкция на языке C для инкремента
- **nc = nc + 1;** - формула аналогична инкременту
- **++** - оператор инкремента

Количество строк

```c
#include <stdio.h>

int main() {
    int c, nl;
    nl = 0;
    while ((c = getchar()) != EOF) {
        if (c == '\n') {
            ++nl;
        }
    }
    printf("%d\n", nl);
}
```

Символьная константа

- Символьная константа - символ, записанный в одинарных кавычках, который представляет числовое значение, равное коду символа в таблице кодировки системы

- Пример: `'\n'`

- Типовая ошибка: `"\n"`

Количество слов

```c
#include <stdio.h>

#define IN 1
#define OUT 0

main() {
    int c, nl, nw, nc, state;
    state = OUT;
    nl = nw = nc = 0;
    while ((c = getchar()) != EOF) {
        ++nc;
        if (c == '\n') {
            ++nl;
        }
        if (c == " " || c == '\n' || c == '\t') {
            state = OUT;
        } else if (state == OUT) {
            state = IN;
            ++nw;
        }
    }
    printf("%d %d %d\n", nl, nw, nc);
}
```

Количество слов

- **||** - оператор **или**
- **&&** - оператор **и**
- **nl = nw = nc = 0;** эквивалентно **nl = (nw = (nc = 0));** - все значения равны 0
- Конструкция **if-else**. **else** - альтернативные действия, если условие **if** ложно

```
if (выражение)
  инструкция1
else
  инструкция2
```

Количество цифр,
символов разделителей
и остальных символов

```c
#include <stdio.h>

int main() {
    int c, i, nwhite, nother;
    int ndigit[10];
    nwhite = nother = 0;
    for (i = 0; i < 10; ++i)
        ndigit[i] = 0;
    while ((c = getchar()) != EOF)
        if (c >= '0' && c <= '9')
            ++ndigit[c - '0'];
        else if (c == ' ' || c == '\n' || c == '\t')
            ++nwhite;
        else
            ++nother;
    printf("digits =");
    for (i = 0; i < 10; ++i)
        printf(" %d", ndigit[i]);
    printf(", white space = %d, other = %d\n", nwhite, nother);
}
```

Массивы

- **int ndigit[10];** - объявление переменной-массива из 10 целых чисел

- Элементы массива нумеруются с нуля: **ndigit[0], ndigit[1], … , ndigit[9]**

- **Индекс массива** - любое целое число или целочисленное выражение

Символы и кодировка

- **char** в языке C - целое число (integer)
- **Кодировка** (Encoding) - преобразование числа в символ
- В **таблице кодировки** цифры идут друг за другом от 0 до 9 (см. ВИКИ)

```
if (c >= '0' && c <= '9')
```

Фрагмент таблицы кодировки UTF-8

Цифра	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
16	30	31	32	33	34	35	36	37	38	39
10	48	49	50	51	52	53	54	55	56	57

Функции

- … - это способ собрать в одном месте и скрыть (инкапсулировать) последовательность вычислительных операций, а затем обращаться к ним много раз, не беспокоясь об особенностях реализации

```c
#include <stdio.h>

int power(int base, int power);

int main() {
   int i;
   for (i = 0; i < 10; ++i)
      printf("%d %d %d\n", i, power(2, i), power(-3, i));
   return 0;
}

/* Возведение числа base в степень power => 0 */
int power(int base, int power) {
   int i, p;
   p = 1;
   for (i = 1; i <= power; ++i)
      p = p * base;
   return p;
}
```

Функции в C

```
тип-возвращаемого-значения имя-функции(параметры) {
   объявления
   операторы
}
```

Функции в C

- Объявление или прототип функции

```
int power(int base, int power);
```

- Определение или реализация функции

```
int power(int base, int power) {
   int i, p;
   p = 1;
   for (i = 1; i <= power; ++i)
      p = p * base;
   return p;
}
```

Функции в C

- **Параметр** - переменная в скобках в прототипе или объявлении функции

- **Аргумент** - значение параметра при вызове функции

- Оператор **return** возвращает значение из функции в точку, откуда ее вызывают

```
return выражение;
```

Передача аргументов по значению

- Вызываемая функция получает значение аргументов в виде копии

- Вызываемая функция не может изменить переменные в вызывающей функции

- Вызываемая функция изменяет локальные временные копии переменных

- В большинстве случае это плюс, а не минус

Передача аргументов по значению

```
int power(int base, int power) {
   int p;
   for (p = 1; power > 0; --power)
      p = p * base;
   return p;
}
```

Массивы символов

Поиск самой длинной строки

Самая длинная строка

```
while (поступает следующую строка)
  if (она длиннее предыдущей самой длинной)
    сохранить ее
    сохранить ее длину
вывести самую длинную строку
```

Разделение обязанностей

- Ввод новой строки - функция **getline**
- Сохранение самой длинной строки - копирование в надежное место - функция **copy**
- Функция **main** для управления работой **getline** и **copy**

```c
#include <stdio.h>

#define MAXLINE 1000

int getline(char line[], int MAXLINE);
void copy(char to[], char from[]);

int main() {
   int len;
   int max;
   char line[MAXLINE];
   char longest[MAXLINE];
   max = 0;
   while ((len = getline(line, MAXLINE)) > 0)
      if (len > max) {
         max = len;
         copy(longest, line);
      }
   if (max > 0)
      printf("%s", longest);
   return 0;
}
```

```c
int getline(char s[], int lim) {
   int c, i;

   for (i = 0;
        i < lim-1 && (c = getchar()) != EOF && c != '\n';
        ++i) {
      s[i] = c;
   }
   if (c == '\n'; {
      s[i] = c;
      ++i;
   }

   s[i] = '\0';

   return i;
}
```

Строки в языке С

- \0 - символ конца строки, нулевой символ (код равен 0)

- \0 в конце строки принят за стандарт в языке С

- Размещение в памяти строки “hello\n”

Оперативная память как массив, каждая ячейка - 1 байт

h	e	l	l	o	\n	\0

```
void copy(char to[], char from[])
{
   int i;
   i = 0;
   while ((to[i] = from[i]) != '\0')
      ++i;
}
```