Способы передачи аргументов в функции в языке Си

Передача по значению и передача по ссылке

Обычно подпрограммы могут передавать аргументы двумя способами. Первый **называется передачей по значению**. Данный метод копирует содержимое аргумента в формальный параметр подпрограммы. Изменения, сделанные в параметре, не влияют на значение переменной, используемой при вызове.

Передача по ссылке является вторым способом передачи аргументов. В данном методе копируется адрес аргумента. В подпрограмме адрес используется для доступа к настоящему аргументу, используемому при вызове.

То есть, изменения, сделанные в параметре, влияют на содержимое переменной, используемой при вызове.

Передача по значению

Помимо нескольких исключений, Си для передачи аргументов использует передачу по значению. Это означает, что обычно нельзя изменять переменные, используемые при вызове функции. Рассмотрим пример функции $y=x^2$

Передача по значению

```c
#include <stdio.h>
int sqr (int x);
int main( )
{
int t=10;
printf("%d %d", sqr(t), t);
return 0;
}
int sqr (int x)
{
x = x*x; return x;
}
```

```
100 10
Process returned 0 (0x0)   execution time : 0.516 s
Press any key to continue.
```

Передача по значению

В данном примере значение аргумента 10, передаваемого в sqr(), копируется в параметр x. Когда происходит присваивание x = x * x, модифицируется только локальная переменная x. Переменная t, используемая при вызове sqr(), по-прежнему содержит значение 10. Следовательно, на экране появится «100 10».

Следует помнить, что только копия аргумента передается в функцию. Все, что происходит в функции, не влияет на переменную, используемую при вызове.

Передача по ссылке

Хотя, как правило, передача параметров в Си происходит по значению, можно передать параметр по ссылке. Поскольку в данном случае происходит передача адреса аргумента, возможно изменение значения аргумента, находящегося вне функции.

Указатели передаются в функции как и обычные значения. Необходимо только объявлять параметры типа указатель.

Передача по ссылке. Пример

```c
#include <stdio.h>

void multi (int *px, float y);

main () {
   int x = 34, y = 6; float z = 368;

   multi(&x, z);
   multi(&y, 91);
   printf("%d %d  %f \n", x, y, z);
}

void multi (int *base, float pow) {
   while (pow >= 10) {
      *base = *base * 10;
      pow = pow / 10;
   }
}
```

```
3400 60  368.000000

Process returned 21 (0x15)   execution time : 0.547 s
Press any key to continue.
```

Унарный оператор & используется для получения адресов переменных. Поэтому в функцию передаются адреса x и y, а не их значения.

Указатели

Оператор * используется для доступа к переменным, на которые указывают операнды. Следовательно, содержимое переменных, используемых при вызове функции, обменивается.

Функция multi() ничего не возвращает, что подчеркнуто с помощью ключевого слова void. Принимает эта функция адрес, который присваивается локальной переменной-указателю, и целое число. В теле функции происходит изменение значения по адресу, содержащемуся в указателе. Но по сути это адрес переменной x из фукнции main(), а значит меняется и ее значение

Указатели

Когда multi() вызывается в main(), то в качестве первого параметра мы должны передать адрес, а не значение. Поэтому, например, вызов multi(x, 786) привел бы к ошибке, а вызов multi(&x, 786) — правильный, т.к. мы берем адрес переменной x и передаем его в функцию.

Адрес

При этом можно объявить в main() указатель и передавать именно его (в данном случае сама переменная p содержит адрес):

```
#include <stdio.h>
void multi (int *px, float y);
int main () {
   int x = 34, y = 6;
   int *p;
   p = &x;
   multi(p, 367);
   p = &y;
   multi(p, 367);
   printf("%d %d\n", x, y);
   return 0;
}
void multi (int *base, float pow) {
   while (pow >= 10) {
      *base = *base * 10;
      pow = pow / 10;
   }
}
```

```
3400 600
```

Передача массивов в функции

- Когда массив используется в качестве аргумента функции, передается только адрес массива, а не копия всего массива. При вызове функции с именем массива в функцию передается указатель на первый элемент массива. Надо помнить, что в Си имена массивов без индекса - это указатели на первый элемент массива. Параметр должен иметь тип, совместимый с указателем. Имеется три способа объявления параметра, предназначенного для получения указателя на массив.

Передача массивов в функции. Первый способ

Он может быть объявлен как массив. Хотя параметр num объявляется как целочисленный массив из десяти элементов, Си автоматически преобразует его к целочисленному указателю, поскольку не существует параметра, который мог бы на самом деле принять весь массив. Передается только указатель на массив, поэтому должен быть параметр, способный принять его.

Передача массивов в функции. Первый способ

```
#include <stdio.h>
void display(int num[10]);
int main (void) /* вывод чисел */
{
int  t [10], i;
for (i=0; i<10; ++i) t[i]=i;
display(t);
return 0;
}
void display(int num[10])
{
int i;
for (i=0; i<10; i++) printf ("%d", num[i]);
}
```

```
0123456789
Process returned 0 (0x0)   execution time : 0.859 s
Press any key to continue.
```

Передача массивов в функции. Второй способ

Следующий способ состоит в объявлении параметра для указания на безразмерный массив, как показано ниже:

```
void display(int num[])
{
int i;
for (i=0; i<10; i++) printf("%d ", num[i]);
}
```

где num объявлен как целочисленный массив неизвестного размера. Поскольку Си не предоставляет проверку границ массива, настоящий размер массива не имеет никакого отношения к параметру (но, естественно, не к программе). Данный метод объявления также определяет num как

Передача массивов в функции. Третий способ

Последний способ, которым может быть объявлен num, - это наиболее типичный способ, - через указатель, как показано ниже:

```
void display(int *num)
{
int i;
for (i=0; i<10; i++) printf ("%d ", num[i]);
}
```

Он допустим, поскольку любой указатель может быть индексирован с использованием [ ], если он является массивом.

Все три метода объявления параметра приводят к одинаковому результату - указателю.

Передача массивов в функции

С другой стороны, элемент массива используется как аргумент, трактуемый как и другие простые переменные. Например, программа может быть написана без передачи всего массива:

```
#include <stdio.h>
void display(int num);
int main(void) /* вывод чисел */
{
int t[10], i;
for (i=0; i<10; ++i) t[i] = i;
for (i=0; i<10; i++) display(t[i]);
return 0;
}

void display(int num)
{
printf ("%d ", num);
}
```

```
0 1 2 3 4 5 6 7 8 9
Process returned 0 (0x0)   execution time : 0.887 s
Press any key to continue.
```

Аргументы функции main()

Си поддерживает два аргумента main() – это **argc и argv.** Они позволяют передавать аргументы командной строки в программу. Аргументы командной строки - это информация, следующая за именем программы в командной строке операционной системы.

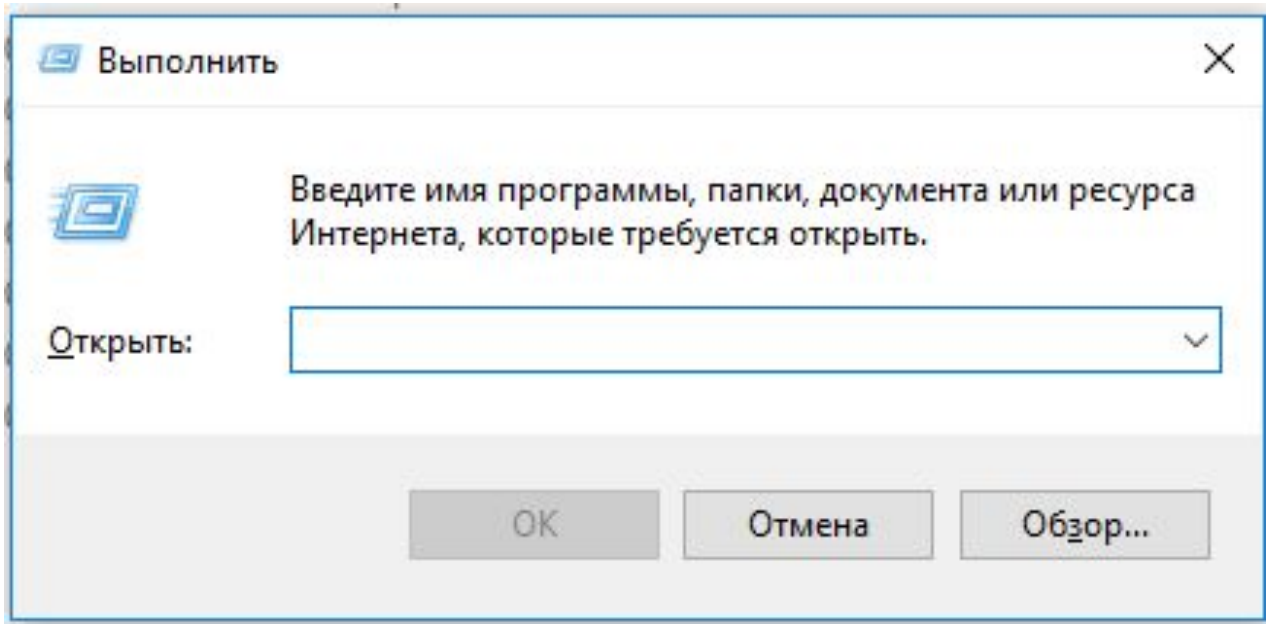
Командная строка

Командная строка в любой версии windows является важным атрибутом.

Для ее вызова можно воспользоваться командой «Выполнить», для чего нужно вызвать специальное окно комбинацией клавиш «**Win»+» R**».

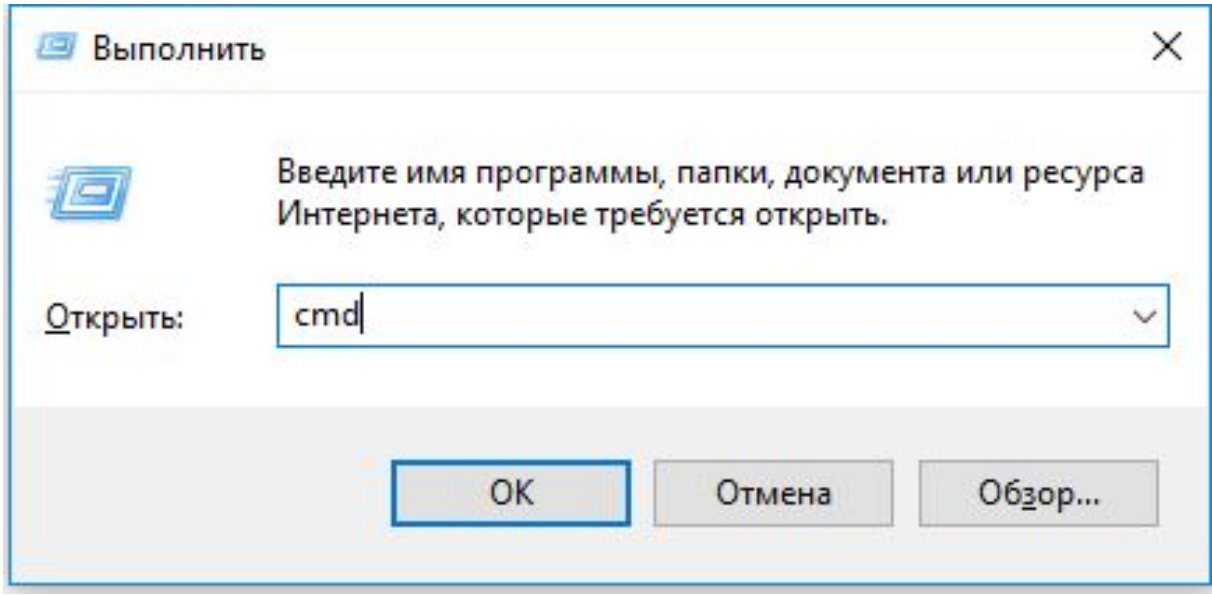
Командная строка

Появляется окно «Выполнить»

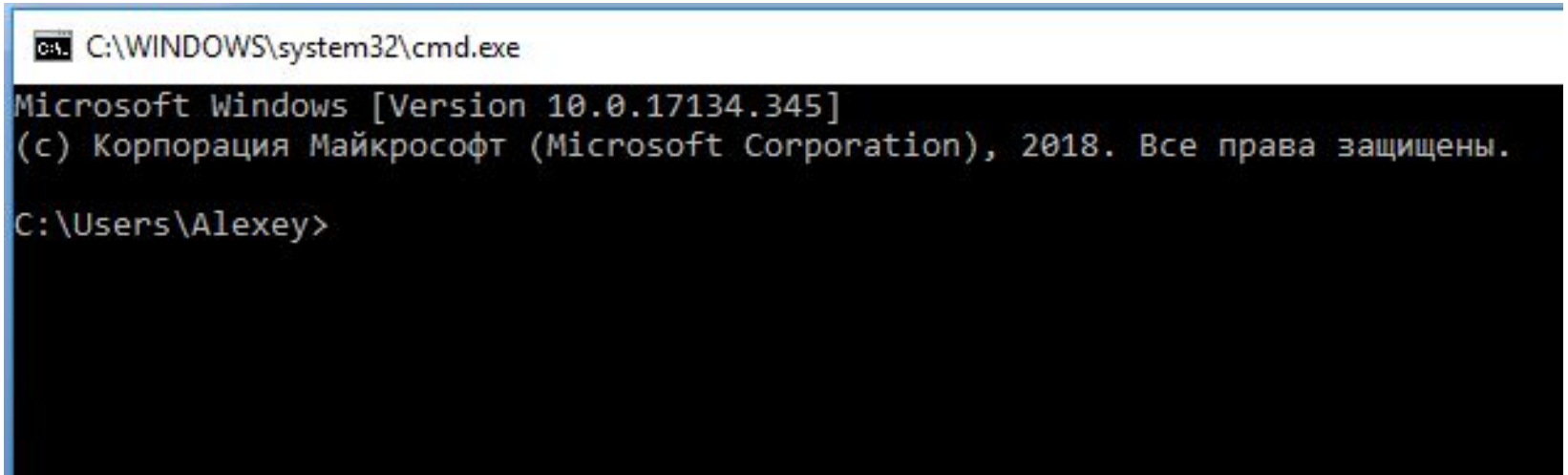


Командная строка

В этом окне набираем «cmd»



Далее нажимается «**Ок**» и командная строка запустится.



Аргументы функции main()

Параметр **argc** содержит число аргументов командной строки и является целым числом. Он всегда равен, по крайней мере, 1, поскольку имя программы квалифицируется как первый аргумент. Параметр **argv** - это указатель на массив символьных указателей. Каждый элемент данного массива указывает на аргумент командной строки. Все аргументы командной строки - это строки. Все числа конвертируются программой во внутренний формат.

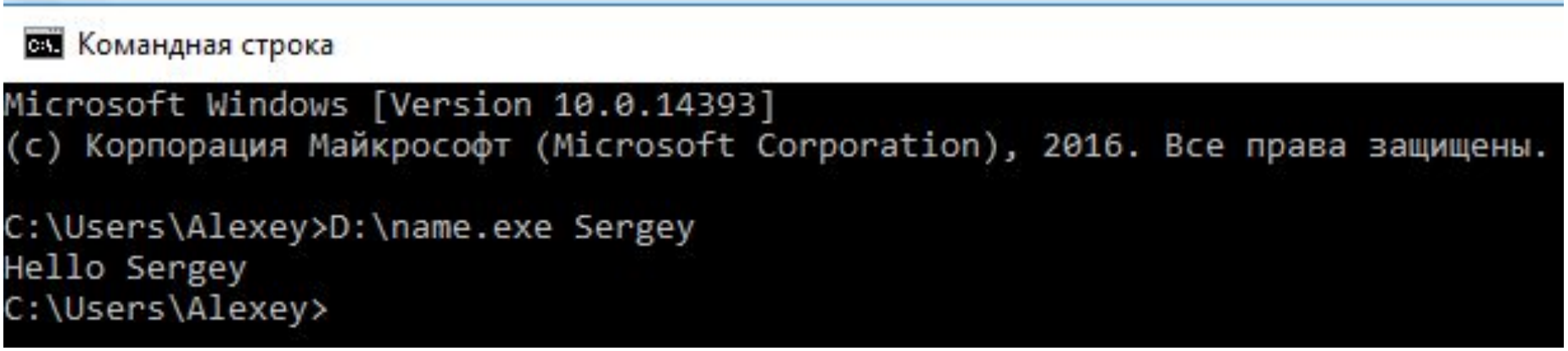
Аргументы функции main()

- Аргументы командной строки должны отделяться пробелами или табуляциями. Запятые, точки с запятыми и им подобные символы не рассматриваются как разделители. Если необходимо передать строку, содержащую пробелы или табуляции в виде одного аргумента, следует ее заключить в двойные кавычки. Например: "this is a test". Важно правильно объявить argv. Наиболее типичным методом является:
- char *argv[];
- Обычно argc и argv используются для получения исходных команд.

Аргументы функции main()

- Программа выводит «Hello», а затем имя пользователя, если его набрать прямо за именем программы:

```c
#include <stdio.h>
int main(int argc, char *argv[])
{
if(argc!=2)
{
printf ("You forgot to type your name\n");
return 1;
}
printf("Hello %s", argv[1]);
return 0;
}
```



Аргументы функции main()