

# Основы алгоритмизации и программирования

## Лекция 5

### Базовые инструкции языка С

# Стандартная библиотека языка Си

В любой программе кроме операторов и операций используются средства **библиотек**, входящих в среду программирования. Часть библиотек стандартизована и поставляется с компилятором. **Функции**, входящие в библиотеку **языка Си**, намного облегчают создание программ. Расширение библиотечных файлов ***\*.lib***.

В **стандартную библиотеку** входят также **прототипы функций**, **макросы**, **глобальные константы**. Это заголовочные файлы с расширением ***\*.h***, которые хранятся в папке ***include*** и подключаются на этапе предпроцессорной обработки исходного текста программ.

# Стандартные математические функции

Математические функции языка **Си** декларированы в файлах ***math.h*** и ***stdlib.h***.  
В приведенных здесь функциях аргументы и возвращаемый результат имеют тип ***double***.  
Аргументы тригонометрических функций должны быть заданы в **радианах** (**2π радиан = 360°**)

Математическая функция	ID функции в языке Си
$\sqrt{x}$	sqrt(x)
$ x $	fabs(x)
$e^x$	exp(x)
$x^y$	pow(x,y)
$\ln(x)$	log(x)
$\lg_{10}(x)$	log10(x)
$\sin(x)$	sin(x)
$\cos(x)$	cos(x)
$\operatorname{tg}(x)$	tan(x)
$\arcsin(x)$	asin(x)
$\arccos(x)$	acos(x)
$\operatorname{arctg}(x)$	atan(x)
$\operatorname{arctg}(x / y)$	atan2(x)
$\operatorname{sh}(x)=0.5 (e^x-e^{-x})$	sinh(x)
$\operatorname{ch}(x)=0.5 (e^x+e^{-x})$	cosh(x)
$\operatorname{tgh}(x)$	tanh(x)
остаток от деления x на y	fmod(x,y)
наименьшее целое $\geq x$	ceil(x)
наибольшее целое $\leq x$	floor(x)

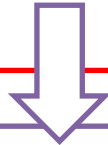
# Функции вывода данных на дисплей

В языке Си нет встроенных средств ввода/вывода данных. Ввод/вывод информации осуществляется с помощью библиотечных функций и объектов. Декларации функций ввода/вывода приведены в заголовочном файле *stdio.h*

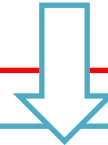
Для вывода информации на экран монитора (дисплей) в языке Си чаще всего используются функции: *printf()* и *puts()*.

Формат функции форматного вывода на экран:

*printf( управляющая строка , список объектов вывода );*



В *управляющей строке*, заключенной в кавычки, записывают: **поясняющий текст**, который выводится на экран без изменения (комментарии), **список модификаторов форматов**, указывающих компилятору способ вывода объектов (признак модификатора формата – символ **%**) и **специальные символы**, управляющие выводом (признак – символ **\**)



В *списке объектов вывода* указываются **идентификаторы** печатаемых объектов, разделенных запятыми: **переменные**, **константы** или **выражения**, вычисляемые перед выводом

# Функции вывода данных на дисплей

**Количество и порядок** следования форматов должен совпадать с количеством и порядком следования выводимых на экран объектов.

Функция ***printf*** выполняет **вывод** данных в соответствии с **указанными форматами**, поэтому формат может использоваться и **для преобразования типов** выводимых объектов

Если признака модификации (**%**) нет, то вся информация выводится как **комментарии**

Для чисел ***long*** добавляется символ ***l***, например, ***%ld*** – длинное целое, ***%lf*** – число вещественное с удвоенной точностью – ***double***

## Основные модификаторы формата



<b>%d (%i)</b>	– десятичное целое число;
<b>%c</b>	– один символ;
<b>%s</b>	– строка символов;
<b>%f</b>	– число с плавающей точкой, десятичная запись;
<b>%e</b>	– число с плавающей точкой, экспоненциальная запись;
<b>%g</b>	– используется вместо f, e для исключения незначащих нулей;
<b>%o</b>	– восьмеричное число без знака;
<b>%x</b>	– шестнадцатеричное число без знака.

# Функции вывода данных на дисплей

Управляют выводом  
специальные  
символы:

**\n** – новая строка;  
**\t** – горизонтальная  
табуляция;  
**\b** – шаг назад;  
**\r** – возврат каретки;  
**\v** – вертикальная  
табуляция;  
**\\** – обратная косая;  
**\'** – апостроф;  
**\"** – кавычки;  
**\0** – нулевой символ  
(пусто)

Если нужно напечатать сам символ **%**, то его нужно указать 2 раза:  
**printf ("Только %d%% предприятий не работало. \n",5);**  
Получим: *Только 5% предприятий не работало.*

## Пример

```
#define PI 3.14159
...
int number = 5;
float bat = 255;
int cost = 11000;
...
printf(" %d студентов съели %f бутербродов. \n", number, bat);
printf(" Значение числа pi равно %f. \n", pi);
printf(" Стоимость этой вещи %d %s. \n", cost, "Руб.");
```

# Функции вывода данных на дисплей

В модификаторах формата функции **printf** после символа **%** можно указывать число, задающее минимальную ширину поля вывода, например, **%5d** – для **целых**, **%4.2f** – для **вещественных** – две цифры после запятой для поля шириной 4 символа. Если **указанных позиций** для вывода целой части числа **не хватает**, то **происходит автоматическое расширение**

Если после «**%**» указан знак «**минус**», то выводимое значение будет печататься с левой позиции поля вывода, заданной ширины, например: **% - 10d**

## Использование функции **printf** для преобразования данных

<b>printf("%d", 336.65);</b>	получим:	336 int
<b>printf("%o", 336);</b>	получим:	520 oct
<b>printf("%x", 336);</b>	получим:	150 hex

## Использование **printf** для нахождения кода **ASCII** символа

```
printf (" %c – %d\n", 'a', 'a');
```

Функция **puts(ID строки);** **выводит** на экран дисплея **строку символов**, автоматически **добавляя** к ней символ перехода на начало новой строки (**\n**). Аналогом такой функции будет:  
**printf(“%s \n”, ID строки);**

Функция **putchar()** **выдает** на экран дисплея один символ без добавления символа **‘\n’**

# Функции ввода информации

Функция,  
**форматированного**  
**ввода** исходной  
информации с  
клавиатуры

***scanf*** (*управляющая строка* , *список адресов объектов ввода*);

В **управляющей строке** указываются только модификаторы форматов, количество и порядок следования которых должны совпадать с количеством и порядком следования вводимых объектов, а тип данных будет преобразовываться в соответствии с модификаторами

**Список объектов** ввода представляет собой **адреса переменных**, разделенные запятыми, т.е. **для ввода** значения переменной перед ее идентификатором **указывается** символ **&**, обозначающий операцию «**взять адрес**»

Если нужно **ввести** значение **строковой переменной**, то использовать символ **&** **не нужно**, т.к. **строка** – это **массив** символов, а **ID** массива является адресом его первого элемента.

Пример\*

```
int course;  
double grant;  
char name[20];  
printf (" Укажите курс, стипендию, имя \n ");  
scanf ("%d %lf %s", &course, &grant, name);
```

\*Вводить данные с клавиатуры можно как в одной строке через пробелы, так и в форме разных строк, нажимая после ввода текущего объекта клавишу *Enter*.



# Функции ввода информации

Функция **scanf()** использует практически тот же набор модификаторов форматов, что и **printf()**; **отличия** от функции вывода **следующие**: **отсутствует формат %g**, форматы **%e,%f** – эквивалентны. Для ввода коротких целых чисел введен модификатор формата **%h**.

**Внимание.** Функцией **scanf()** по формату **%s** строка вводится только до первого пробела!



Для ввода фраз, состоящих из слов, разделенных пробелами, используется функция:

**gets (ID строковой переменной);**

Символы вводятся при помощи функции **getch()**. Причем простой ее **вызов** организует **паузу**, при которой **система** программирования **приостановит** **выполнение** программы и будет **ждать нажатия** любой клавиши. Так поступают в том случае, когда нужно просмотреть какие-то результаты работы, при выводе их на экран монитора.

Если использовать **getch()**; ее в правой части операции присваивания, **например**:  
char c;  
c = getch();  
символьная переменная **c** получит значение кода нажатой клавиши

# Функции ввода информации

С началом работы любой программы автоматически открываются стандартные потоки для ввода (***stdin***) и вывода данных (***stdout***), которые по умолчанию связаны с клавиатурой и экраном монитора соответственно.

**Внимание.** Ввод данных функциями ***gets()***, ***getch()*** выполняется с использованием потока ***stdin***. Если указанная функция не выполняет своих действий (**проскакивает**), перед использованием необходимо **очистить поток** (буфер) ввода с помощью функции:

***fflush(stdin);***