# Лабораторная работа №1 "Основные сведения о языке С"

**(продолжительность 2 часа)**

**Цель:** Получить практические навыки анализа сложности алгоритмов.

**1. Краткие теоретические сведения**

Язык Си, созданный Денисом Ритчи в начале 70-х годов в Bell Laboratory американской корпорации AT&T, является одним из универсальных языков программирования. Язык Си считается языком системного программирования, хотя он удобен и для написания прикладных программ. Среди преимуществ языка Си следует отметить переносимость программ на компьютеры различной архитектуры и из одной операционной системы в другую, лаконичность записи алгоритмов, логическую стройность программ, а также возможность получить программный код, сравнимый по скорости выполнения с программами, написанными на языке ассемблера. Последнее связано с тем, что хотя Си является языком высокого уровня, имеющим полный набор конструкций структурного программирования, он также обладает набором низкоуровневых средств, обеспечивающих доступ к аппаратным средствам компьютера. С 1989 года язык Си регламентируется стандартом Американского института национальных стандартов ANSI С. В настоящее время, кроме стандарта ANSI C разработан международный стандарт ISO C (International Standard Organization C).

Пример структуры программы:

#include <stdio.h> //Подключение библиотеки ввода/вывода

#define C 23 //Объявление константы C = 23

int global = 10; //Объявление глобальной переменной

//Функция суммы двух целых чисел

int sum(int a, int b)

{

return a + b;

}

//Точка входа в программу - главная функция

int main()

{

int local = 5; //Локальная переменная

const int b = 3; //Константа типа int

//Вызов функции и присваивание её результата переменной local

local = sum(global, b);

printf("%d", local); //Вывод результата на экран: 13

//Возврат целочисленного значения перед завершением функции

return 0;

}

**Директивы препроцессора** представляют собой инструкции, записанные в тексте программы на СИ, и выполняемые до трансляции программы. Все директивы препроцессора начинаются со знака #. После директив препроцессора точка с запятой не ставятся.

**Основные директивы препроцессора:**

* #include широко используется для включения в программу так называемых заголовочных файлов, содержащих прототипы библиотечных функций, и поэтому большинство программ на СИ начинаются с этой директивы.
* #define используется, что бы создать константы.

Библиотеки:

**Стандартная библиотека языка Си** – это набор отдельных файлов, которые расширяют возможности языка Си.

Возможности языка Си, без функций стандартной библиотеки, очень ограничены. Не представляется возможным даже вывести значение переменной на экран. Но благодаря дополнительным модулям (стандартным заголовочным файлам) возможности языка могут быть существенно расширены. Библиотеки можно подключать с помощью директивы #include.

Вот некоторые, **наиболее практичные:**

* <stdio.h> //Осуществляет ввод и вывод на экран
* <math.h> //Для вычисления основных математических функций
* <string.h> //Позволяет работать с различными видами строк
* <time.h> //Для конвертации между различными форматами даты и времени
* <locale.h> //Используется для задач, связанных с локализацией

**Тип данных** определяет множество значений, набор операций, которые можно применять к таким значениям и способ реализации хранения значений и выполнения операций.

Различают следующие типы данных:

**Простые данные:**

целочисленные,

вещественные,

символьные

логические.

**Составные данные:**

массив,

строковый тип ,

структура.

**Другие типы данных:**

указатель.

Простые данные:

**Целочисленные данные** могут быть представлены в знаковой и беззнаковой форме.  
Основные типы и размеры целочисленных данных:

* short int (2 байта)
* int (4 байта)
* long int (8 байт)

**Вещественный тип** предназначен для представления действительных чисел.

Различают три основных типа представления вещественных чисел в языке Си:

* float (4 байта)
* double (8 байт)
* long double (16 байт)

**Символьный тип** хранит код символа и используется для отображения символов в различных кодировках. Символьные данные задаются в кодах и по сути представляют собой целочисленные значения. Для хранения кодов символов в языке Си используется тип char (1 байт).

**Логический тип** применяется в логических операциях, используется при алгоритмических проверках условий и в циклах и имеет два значения: истина — true

ложь — false. Для хранения используется тип bool (1 байт).

Составные данные:

**Массив** — индексированный набор элементов одного типа.

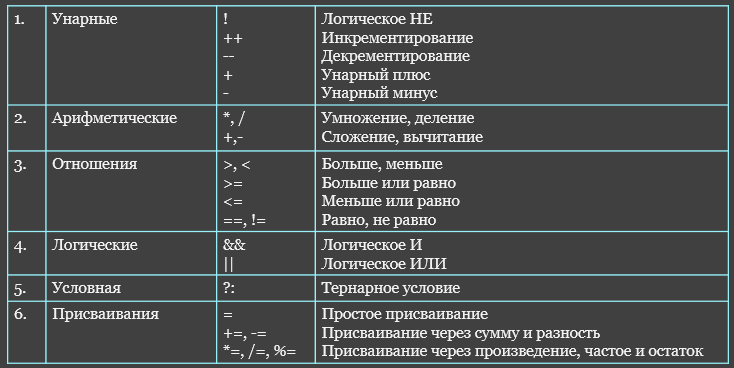
**Строковый тип** — массив, хранящий строку символов.

**Структура** — набор различных элементов (полей записи), хранимый как единое целое и предусматривающий доступ к отдельным полям структуры.

Другие типы данных:

**Указатель** — хранит адрес в памяти компьютера, указывающий на какую-либо информацию, как правило — указатель на переменную.

Приоритет операций:

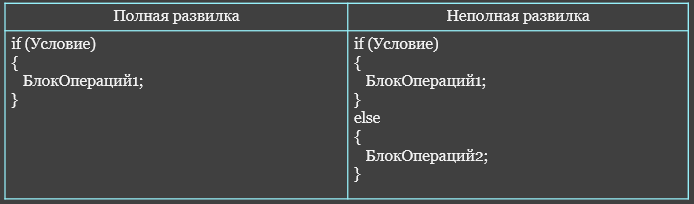


Условные операторы:

**Разветвляющимся** называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса.   
Признаком разветвляющегося алгоритма является наличие операций проверки условия. Чаще всего для проверки условия используется условный оператор if.

**Условный оператор if**

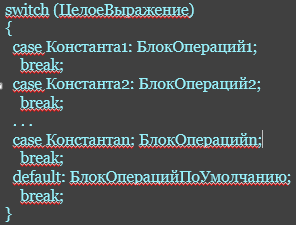
Условный оператор if может использоваться в форме ***полной*** или ***неполной*** развилки.



**Оператор ветвления switch** (оператор множественного выбора)

Оператор if позволяет осуществить выбор только между двумя вариантами. Для того, чтобы производить выбор одного из нескольких вариантов необходимо использовать вложенный оператор if. С этой же целью можно использовать оператор ветвления switch.

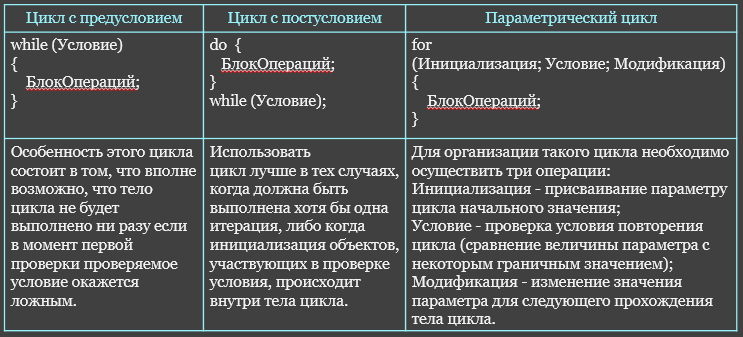
Общая форма записи:



Константы в опциях case должны быть целого типа (могут быть символами).

Операторы цикла:

**Циклом** называется блок кода, который для решения задачи требуется повторить несколько раз. В языке Си следующие виды циклов:



Функции:

**Функция** — это подпрограмма, которая может содержаться в основной программе, а может быть создана отдельно (в библиотеке). Каждая функция выполняет в программе определенные действия.   
Определение функции имеет следующий синтаксис:

ТипВозвращаемогоЗначения ИмяФункции(СписокФормальныхАргументов)  
{  
   ТелоФункции;  
  ...  
  return(ВозвращаемоеЗначение);  
}

Пример функции сложения двух вещественных чисел:

float function(float x, float z)  
{  
  float y;  
  y=x+z;  
  return(y);  
}

**2. Постановка задачи**

Необходимо написать и добиться безошибочного выполнения кодов следующих задач:

1. Вычисление алгебраического выражения, содержащего операции различного уровня приоритетности Задается преподавателем
2. Вычисление алгебраического выражения, требующего преобразования типов. Задается преподавателем

**3. Содержание отчета**

1. Титульный лист.

2. Название и цель работы.

3. Постановка задачи.

4. Листинг программы с комментариями.