основы алгоритмизации и программирования

ВВЕДЕНИЕ В ЯЗЫК С

ПЛАН ЛЕКЦИИ

1. Введение
2. Структура программы
3. Директивы
4. Библиотеки
5. Типы данных
    - 5.1 Простые данные
    - 5.2 Составные данные
    - 5.3 Другие типы данных

ПЛАН ЛЕКЦИИ

6. Преобразование типов
7. Переменные и константы
8. Операции отношения. Логические операции
9. Приоритет операций
10. Условные операторы
11. Операторы цикла
12. Функции

ВВЕДЕНИЕ

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 C. В настоящее время, кроме стандарта ANSI C разработан международный стандарт ISO C (International Standard Organization C).

СТРУКТУРА 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

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

ТИПЫ ДАННЫХ

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.

ПРЕОБРАЗОВАНИЯ ТИПОВ

Приведением типа называется преобразование значения переменной одного типа в значение другого типа.

При явном приведении перед выражением следует указать в круглых скобках имя типа, к которому необходимо преобразовать исходное значение.

Пример явного приведения типа.

```
int x = 5;
double y = 15.3;
x = (int) y;
y = (double) x;
```

При неявном приведении преобразование происходит автоматически, по правилам, заложенным в языке Си.

Пример неявного приведения типа.

```
int x = 5;
double y = 15.3;
y = x; //здесь происходит неявное приведение типа к double
x = y; //здесь происходит неявное приведение типа к int
```

ПЕРЕМЕННЫЕ И КОНСТАНТЫ

Переменная — область памяти, в которую могут помещаться различные значения.
Любая переменная до ее использования в программе на языке Си должна быть объявлена, то есть для нее должны быть указаны тип и имя.
Объявление переменных в Си осуществляется в форме:

ТипПеременной ИмяПеременной; `// int i;`

При объявлении переменной ей может быть присвоено начальное значение в форме:

ТипПеременной ИмяПеременной=значение; `// int i = 10;`

Константа — это ограниченная последовательность символов алфавита языка, представляющая собой изображение неизменяемого объекта.
Константы бывают числовые, символьные и строковые. Числовые константы делятся на целочисленные и вещественные.
Объявление константы в Си осуществляется в форме:

const тип ИмяПеременной = НачальноеЗначение; `// const int i = 10;`

ОПЕРАЦИИ ОТНОШЕНИЯ И ЛОГИКИ

Операции отношения:

- a == b эквивалентно — проверка на равенство;
- a != b не равно — проверка на неравенство;
- a < b меньше;
- a > b больше;
- a <= b меньше или равно;
- a >= b больше или равно.

Операции отношения используются при организации условий и ветвлений.

Логические операции:

a && b — И (конъюнкция) — требуется одновременное выполнение всех операций отношения;

a || b — ИЛИ (дизъюнкция) — требуется выполнение хотя бы одной операции отношения;

!a — НЕ (инверсия) — требуется невыполнение операции отношения.

Логические операции чаще всего используются в операциях проверки условия if и могут выполняться над любыми объектами.

ПРИОРИТЕТ ОПЕРАЦИЙ

1.	Унарные	! ++ -- + -	Логическое НЕ Инкрементирование Декрементирование Унарный плюс Унарный минус
2.	Арифметические	*, / +,-	Умножение, деление Сложение, вычитание
3.	Отношения	>, < >= <= ==, !=	Больше, меньше Больше или равно Меньше или равно Равно, не равно
4.	Логические	&& ||	Логическое И Логическое ИЛИ
5.	Условная	?:	Тернарное условие
6.	Присваивания	= +=, -= *=, /=, %=	Простое присваивание Присваивание через сумму и разность Присваивание через произведение, частое и остаток

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.

Полная развилка	Неполная развилка

if (Условие)
{
  БлокОпераций1;
}
	if (Условие)
	{
	  БлокОпераций1;
	}
	else
	{
	  БлокОпераций2;
	}

УСЛОВНЫЕ ОПЕРАТОРЫ

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:

```
switch (ЦелоеВыражение)
{
  case Константа1: БлокОпераций1;
    break;
  case Константа2: БлокОпераций2;
    break;
  . . .
  case Константаn: БлокОперацийn;
    break;
  default: БлокОперацийПоУмолчанию;
    break;
}
```

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:

Цикл с предусловием	Цикл с постусловием	Параметрический цикл

while (Условие)
{
БлокОпераций;
}
	do {
	БлокОпераций;
	}
	while (Условие);
		for
		(Инициализация; Условие; Модификация)
		{
		БлокОпераций;
		}

Особенность этого цикла состоит в том, что вполне возможно, что тело цикла не будет выполнено ни разу если в момент первой проверки проверяемое условие окажется ложным.	Использовать цикл лучше в тех случаях, когда должна быть выполнена хотя бы одна итерация, либо когда инициализация объектов, участвующих в проверке условия, происходит внутри тела цикла.	Для организации такого цикла необходимо осуществить три операции: Инициализация - присваивание параметру цикла начального значения; Условие - проверка условия повторения цикла (сравнение величины параметра с некоторым граничным значением); Модификация - изменение значения параметра для следующего прохождения тела цикла.

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: Функция сложения двух вещественных чисел.

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