|  |  |  |  |
| --- | --- | --- | --- |
| Length | Name | Example |  |
| 1 | Bit | 0 |  |
| 4 | Nibble | 1011 |  |
| 8 октет | =1 Byte | 10110101 о | Октет числа от 0 до 255 (256 |

1. **Система** [**типов**](https://metanit.com/cpp/tutorial/2.3.php) **данных C**

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Число байтов | Диапазон |  |
| char | 1 – 8 bit | -128 / +127 | арифметика |
| unsigned char | 1 – 8 bit | 0 / 255 |  |
| short | 2 – 16 bit | -32768 / +32767 |  |
| unsigned short | 2 – 16 bit | 0 / 65536 |  |
| Int | 4 – 32 bit | −2147483648 / 2147483647 |  |
| unsigned int | 4 – 32 bit | 0 / 4 294 967 295 |  |
| long long | 8 – 64 bit | −9223372036854775808 / +922… |  |
| unsigned long long | 8 – 64 bit | 0 / 18446744073709551615 |  |
| \* pointer | 8 – 64 bit | адрес |  |

type name(type1 param1, type2 param2, ...)

{

local variables declaratin // объявление локальных переменных, существуют только внутри функции

instructions // инструкции

}

устанавливаем тип функции (тип результата (возвращаемое значение)). Получает значения через параменты и выдает результат

список параметров которые получает функция

Int sum(int value1, int value2) // sum приводит к целому числу int, получает два парамента

{

Int total; //объявили локал переменную(целочисленную) – существует внутри функ.

Instrctions

}

1. Arrays / Массив

type name[numeric value];

int tab[42]; //тип имя[42 целых числа] от 0 до 41

Последовательность элементов(объекта) одного типа int в памяти

2. Structures / Структура

struct struct\_name struct s\_fortytwo

{ {

type name; int a; //целое число a

type name; char b[21] //массив b из 21 символа

}; };

struct struct\_name name; struct s\_fortytwo a;

определить представление объекта в памяти, который будет составлен в свою очередь из более простых элементов, которые могут быть тоже структурами, массивами и т.д.

Мы резервируем в памяти необходимое пространство, и когда мы используем это имя, программа узнает, что это структура.

**Main** – первая функция выполняемая в коде

int main() // возвращает инт

{

}

**3 вида инструкций**

1. Blocks:

{

declarations;

instructions;

}

2. Expressions / Выражения:

expressions;

3. Control structures / Структуры управления

**Выражения**

1. Непосредственные числовые значения:

|  |  |  |  |
| --- | --- | --- | --- |
| число | число | шестнадцатеричном | восьмеричном |
| 0 | 42 | 0x1A | 010 |

2. Variables переменные

a // переменная int a.

b[18] // 10-ый элемент массива

stf\_var.a // элемент a structure

3. Function calls: вызов функции

sum(18, a) //имя функции и параметры.

4. Computations: + - \* / % (). комбинировать

1 + 3 – ( a \* 42 + b[0]) / sum(21, 34 – b[10])

5. Assignations: присваивание

a = 42

b[2 +a] = sum(12, 30) // индекс массива 2+42, b[44] = 42

sft\_var.a = 0

6. Компораторы / [Операторы](https://ru.wikipedia.org/wiki/Операторы_в_C_и_C%2B%2B) сравнения. == != < > <= >= || && !

a < 2 + b[23]

a – 2 == 35 || b[11] + 3 >= 17

a == b // содержимое перемнной, если равно, то вернет true=1, нет false=0

"==" (равенство), "!="(неравенство), "<" (меньше), ">" (больше), ">=" (больше или равно), "<=" (меньше или равно).

"||" (или), «&&» (и), "!" (не)

7. Бинарные (троичные) операторы: >> << | & ^

8. Составное присваивание:

a += 10 //a = a + 10

a++ //a += 1 //a = a + 1

b[50] /= 2 //b[50] = b[50] / 2

9. Особенности

‘A’ // ASCII-код буквы А = 65

sizeof(тип перемнной или сама переменная) // возвращает размер перемнной

&a // адрес переменной в памяти RAM

exp1 ? exp2 : exp3 // если exp1 истинна, то вычиляется exp2, в противном случае — exp3

“hello forty-two” // В массиве char будет автоматически храниться последовательность ASCII-кодов. Массив заканчивается на \0 (терминирующий ноль), этот ноль вычисляет адрес в памяти первого символа.

1. **Указатели**

1. Когда мы перечисляли типы в С, то там мы видели определенный тип – указатели (pointers).

Указатель — переменная, содержащая адрес объекта. Указатель не несет информации о содержимом объекта, а содержит сведения о том, где размещен объект. Она тоже имеет адрес, а ее значение является адресом некоторой другой переменной.

int a; // объявляем переменную a типа int

int \*p; // объявляем указатель **p** содержит адрес переменной **a** типа **int**

p = &a; // указатель p получает адрес переменной a

\*p = 1 // присваиваем указателю p – 1, исходя из этого переменная a = 1 (\*p = 1; <> a =1;)

int tab[42];

int \*p;

p = &(tab[0]); // вводим в указатель p – первый индекс массива tab

\*(p + 5) // получаем доступ к значению 5 массива tab, тоже самое, если напишу

// tab[5] <> \*(p + 5)

1. **Управление**

1.

If если

{

}

Else иначе

{

}

If else

{

}

2

while (Условие)

{

БлокОпераций;

}

1. return оператор завершает выполнение функции, осуществляет возврат одно значения в точку ее вызова. Это значение присваивается переменной, тип которой должен соответствовать типу возвращаемого значения функции. Не обязательно в конце.

return(значение); // в функции, которая возвращает значение

main(void)

{

return; // в функции, которая не возвращает значение или полностью отсутсвовать

}