*Одномерный массив == Вектор*

*Каждый элемент одномерного массива имеет индекс, который говорит о расположении данного элемента.*

*А = [3, -5, 15, -11, 44, 32, -2, 0]*

*0, 1, 2, 3, 4, 5, 6, 7*

*Массив должен быть объявлен заранее:*

*Тип\_элемента имя [кол-во элементов];*

*int A [8];*

*double arr [15];*

*int vector [4\*N]; // N - переменная*

*Для каждого массива компилятор выделяет непрерывный участок памяти величина которого: N \* sizeof(тип\_елемента)*

*Объявляя массив можно инициализировать его элементы(2 способа):*

1. *С определением кол-ва элементов: int simple [5] = {2, 3, 4, 5, 7, 11}*
2. *Без определения кол-ва елемента: char glash[] = {‘a’, ‘e, ‘o’, ‘u’}*

*Замечание: обратиться к элементу массива можно через его индекс*

*Замечание 2: Компилятор С не контролирует соответствие кол-ву индексов массива его размерности*

*Пример: в одномерном массиве получить сумму тех элементов массива которые отрицательны и нечетны одновременно*

*Алгоритм*

*Something something output*

*(НЕ) в соответствии с алгоритмом код программы выглядит следующим образом:*

*#include <stdio.h>*

*#define M 20*

*Int main(void)*

*{*

*int vector [M], n, i, sum=0;*

*printf(“input n ”);*

*scanf(“%i”, &n)*

*printf(“input elements[%i]:”, n)*

*for (i=0; i<n; i++)*

*scanf(“%i”, &vector[i])*

*for (i=0; i<n; i++)*

*if ((vector[i] %2!=0) && (vector[i]<0))*

*sum += vector[i]*

*printf(“sum = %i\n”, sum)*

*return 0*

*}*

*Пример 2*

*Алгоритм*

*(╯ ͠° ͟ʖ ͡°)╯┻━┻*

*凸༼ຈل͜ຈ༽凸*

*Текст программы*

¯\\_(ツ)\_/¯

29.10.2019

Текст программы

#include <stdio.h>

#define Size 20

int vector[Size], n, i, res;

int main(void)

{

printf(“input Size: ”)

scanf(“%i”, &n)

printf(“elements of list [%i]: ”, n)

for (i=0,i<n, i++)

{

scanf(“%i”, &vector[i])

}

res = vector[0]

for (i=1, i<n, i++)

{

If (vector[i] < res)

res = vector[i];

}

printf(“minimal value of list is %i\n”, res)

return 0;

}

Многомерные массивы

Язык С интерпретирует многомерный массив как массив массивов.

int matr[10][5]; - массив из 10 элементов, каждый из которых это одномерный массив из 5 элементов. Для любого одномерного массива компилятор выделяет непрерывный участок памяти.

В ОП схема расположения элементов массива выглядит

¯\\_(ツ)\_/¯

Многомерные массивы можно инициализировать при объявлении

Int mt[3][3] = {1, 2, 3, 4, 5, 6, 7 ,8, 9}

Пример:

Вычислить сумму элементов каждой строки введенной матрицы

|a00 a01 a02|

|a10 a11 a12|

|a20 a21 a22|

Алгоритм

♪~ ᕕ(ᐛ)ᕗ

(НЕ очень) В соответствии с приведенным алгоритмом код программы выглядит:

#include <stdio.h>

#define sz 10

05.11.2019

Особенность ф-ции main: она начинает работу программы, а ее завершение оканчивает работу всей программы

В ф-ции main сосредоточены все основные шаги алгоритма

Вложенность ф-ций в языке С запрещена

Каждая фция может быть откомпилирована отдельно, сохраняется в obj файл, затем obj файл подсоединяется к .exe

Структура ф-ции и синтаксис

Function\_type name(formal\_parametrs)

{

Declare internal(local) variables

Operators

}

Где Function\_type значение которое возвращает ф-ция в точку вызова после завершения ее выполнения.

Отдельным типом значения является void, который говорит, что ф-ция не возвращает ничего.

Наличие круглых скобок является обязательным, даже если нет параметров

Пример:

Заданы числа S и t

f(t, -2S, 1.17) + f(2.2, t, S-t)

Найти f(a, b, c) = (2\*a-b-sin(c))/(5+|c|)

Алгоритм

(ง ͠° ͟ل͜ ͡°)ง

НЕ в соответствии с алгоритмом код программы

#include <stdio.h>

#include <math.h>

Float function(float a, float b, float c)

{

Float w;

W = ((2\*a-b-sin(c))/(5+fabs(c)))

Return w

}

Int main(void)

{

Float s, t

printf(“input s, t: ”);

scanf(“%f%f”, &s, &t);

Float res = f(t, -2\*S, 1.17) + f(2.2, t, S-t);

printf(“res=%.3f”, r)

Return 0

}

Алгоритм

¯\\_(ツ)\_/¯

¯\\_(ツ)\_/¯

Не оч в соответствии с алгоритмом код программы

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

void rand\_array(int[],int);

void output(const int[], int);

int max\_num(int[], int, int &);

int main(void)  
{

int const maxsz=10, A[maxsz], size, n, maxel;

printf(“input size of array: ”);

scanf(“%i”, &size);

rand\_array(A, size);

output(A, size);

n = max\_num(A, size, max\_el);

printf(“something something %i, number number %i\n”, max\_el, n);

return 0;

}

void rand\_array(int arr[], int size)

{

srand((unsigned)time(null));

For (int i=0; i<size; i++)

arr[i] = rand();

}

void output(const int arr[]; int size)

{

printf(“elements of array”)

for (int i=0; i<size; i++)

printf(“%i”, arr[i])

}

Int max\_num()

{

Int num=0

max=arr[0]

For (ಠ\_ಠ)

{

[̲̅$̲̅(̲̅ ͡° ͜ʖ ͡°̲̅)̲̅$̲̅]=

}

}

Сортировка методом пузырька

Сортировка выбором

Пузырьковая Сортировка

Код

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

Алгоритм

ʕ•ᴥ•ʔʕ•ᴥ•ʔʕ•ᴥ•ʔ

Указатель - это особый тип данных значением которого является адрес определенного байта RAM(обычно указатель сохраняет адрес первого байта одного из объектов программы)

Указатели бывают:

1. Постоянные - сохраняют постоянный адрес определенного участка RAM - имена массивов и строк.
2. Переменные - им можно указывать значения адресов различных участков памяти.

Указатели должны быть описаны:

Type \*name\_of\_pointer;

Type - тип объектов программы, адреса которых может сохранять данный указатель(стандартный\пользовательский)

Char \*pc;

Unsigned \*p1, \*p2;

Long \*g, h, w; (h, w переменные, g - поинтер)