Программирование

Простейшая программа на C

Структура программы С

```
#include <stdio.h>
int main(void)          //начало программы в Си начинается с main()
{                       //любая программа начинается с открывающей скобки
    оператор 1;         // gосле каждого оператора ставится ;
    оператор 2;
    …
    оператор n;
}                       //любая программа заканчивается закрывающей скобкой
```

Простейшая программа на С

```
#include <stdio.h>

int main(void)
{
    printf("Hello world!");                    // выводим на экран
}
```

// - однострочный комментарий

/* */ - многострочный комментарий

Переменные в С

Переменные в С служат для хранения, изменения информации. У каждой **переменной в С** есть имя и значения. Значения переменной в ходе программы си можно изменять. Чтобы работать с **переменными в С**, их нужно сначала объявить. Это значит указать их тип и имя.

Типы данных:

1) int – целочисленный

2) double – вещественный

3) char – символьный

В 90% случаев прикладного программирования этих трёх типов достаточно

Примеры операторов объявления (declaration)

```
int a;          // переменная целого типа
double b;  // переменная вещественного типа
char c;       // переменная "символьного" целого типа
```

Примеры операторов инициализации (initialization)

```
int a = 1;                  // инициализация целой константой
const double b = 6.2;       // создание именованной константы
char c = ’a’;               // инициализация символьной константой
```

Правила:

1) каждое имя должно быть объявлено

2) объявление и инициализация – разные вещи

```
Тип Имя;                          // объявление имени
Тип Имя = НачальноеЗначение;      // инициализация
```

3) Имя начинается не с цифры и состоит из любых символов кроме

; : ’ " < > , ( ) * ^ % $ # . / ? ! @ ~ + = - | \

Примеры имен:

Name name _name name_var name4

Арифметических действий с числовыми переменными

+ — - сложение, вычитание
* / - умножение, деление
% - остаток от деления

Стандартные функции для числовых переменных

abs(i) - модуль целого числа i
fabs(x) - модуль вещественного числа x
sqrt(x) - квадратный корень x
pow(x,y) - вычисляет x в степени y

Ввод и вывод

Вывод в С

Для вывода текста на экран в С используется оператор printf.

printf (“текст”); // вывод текста
printf (“текст \n”); // вывод текста и переход на другую строку

Для вывода значений на экран используется формат вывода, который задается в кавычках:

printf(“формат вывода”, имя переменной)

printf (“текст %d”, имя переменной); // выводит текст и значение переменной

%d – спецификатор формата

Спецификаторы формата

В формате вывода для разных типов переменных используется:

%d - вывод целого числа (переменная типа int)
%f - вывод вещественного числа (переменная типа double)
%c - вывод одного символа (переменная типа char)
%s - для вывода символьной строки

Пример программы на С

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
    int a,b,c;                  // объявление целочисленных переменных a,b,c
    a=10;                       // присваиваем переменной a значение 10
    b=20;                       // присваиваем переменной  b значение 20
    c=a+b;                      // присваивание переменной c суммы a и b
    printf ("%d+%d= %d", a,b,c);    // форматированный вывод  выражения a+b =c

}
```

Ввод в С

Чтобы ввести записать информацию в переменную **с клавиатуры в С с помощью оператора scanf** необходимо указать сообщение и переменную, в которую будет записываться с клавиатуры значение

scanf ("формат ввода", &имя переменной);

Спецификаторы формата

В формате ввода для разных типов переменных используется:

%d - вывод целого числа (переменная типа int)
%lf - вывод вещественного числа (переменная типа double)
%c - вывод одного символа (переменная типа char)
%s - для вывода символьной строки

Пример программы на С

```
#include <stdio.h>

int main(void)
{
    int a,b,c;
    printf("Введите первое целое число\n");
    scanf("%d", &a);                    // ввод  переменной a с клавиатуры
    printf("Введите второе целое число\n");
    scanf("%d", &b);                    // ввод  переменной  b с клавиатуры
    c=a+b;                             //  присваиваем переменной с значение a+b
    printf("%d+%d= %d\n", a,b,c);              // вывод  выражения a+b =c

}
```

Операторы

Операторы С

1) Оператор присваивания

a = 5;

2) Условные операторы

if else

3) Операторы цикла

while

do while

for

4) Операторы перехода

break

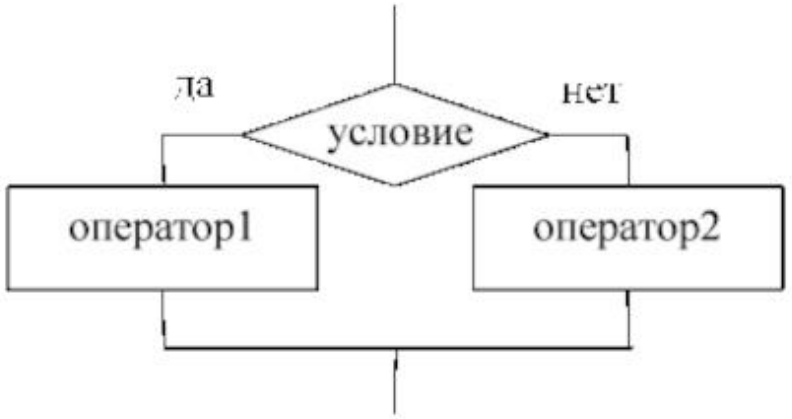
return

goto

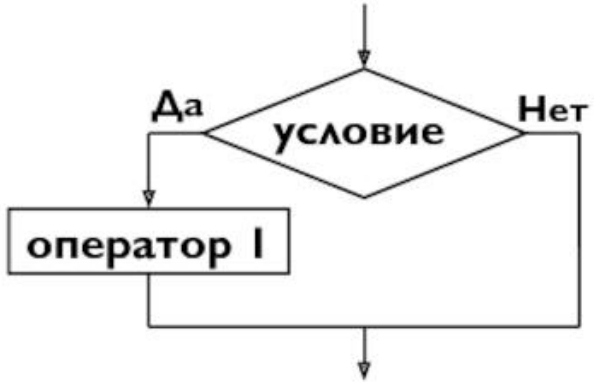
5) Оператор продолжения

continue

Условные операторы




```
If (условие)
{
    Оператор 1;
} else {
        Оператор 2;
}
```




```
If (условие)
{
    Оператор 1;
}
```

Логические операции

&&- логическое И
|| - логическое ИЛИ
! - логическое НЕ

Операции отношения

> - больше
< - меньше
== - равно
>= - больше или равно
<= - меньше или равно
!= - не равно

Множественный выбор в С

Если в программе си есть несколько вариантов действий и их выбор зависит от значения переменной, удобно использовать в Си оператор множественного выбора switch

```
switch (переменная)
{
    case  значение 1 : действие 1; break;
    case  значение 2 : действие 2; break;
    …
    default : действие если переменная не принимает указанных выше значений;
}
```

В случае если переменная будет равна значению 1, то будет выполнено действие 1, в случае если значение переменной равно значению 2,то будет выполнено действие 2

и так далее.

Пример программы на С

```
#include <stdio.h>

int main(void)
{
	double a, b;//  1и 2  число в примере
	char op; // символ алгебраической операции
	printf(" Введите пример : ");
	scanf("%lf %c %lf", &a, &op, &b); // ввести 2 числа и знак алгебраической операции
	switch (op) // оператора выбора в зависимости от знака алгебраической операции
	{
		case '+': printf(" %f %c %f=%f", a,op,b, a+b); break;
		case '-': printf(" %f %c %f=%f", a,op,b, a-b); break;
		case '*': printf(" %f %c %f=%f", a,op,b, a*b); break;
		case '/': printf(" %f %c %f=%f", a,op,b, a/b); break;
		default: printf(" Некорректно введен пример"); // по умолчанию
	}

}
```

Операторы цикла

Цикл с предусловием while

Общая форма записи:
while (Условие)
{
    БлокОпераций;
}

Если **Условие** выполняется, то выполняется **БлокОпераций**, заключенный в фигурные скобки, затем **Условие** проверяется снова.
Последовательность действий, повторяется до тех пор, пока выражение, проверяющее **Условие**, не станет ложным (равным нулю). При этом происходит выход из цикла, и производится выполнение операции, стоящей после оператора цикла.

Цикл с предусловием do…while

Общая форма записи:
do
{
    БлокОпераций;
} while (Условие);

Тело цикла выполняется до тех пор, пока выражение, проверяющее Условие, не станет ложным, то есть тело цикла с постусловием выполнится хотя бы один раз.
Использовать цикл do...while лучше в тех случаях, когда должна быть выполнена хотя бы одна итерация, либо когда инициализация объектов, участвующих в проверке условия, происходит внутри тела цикла.

Параметрический цикл for
Общая форма записи

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

for — параметрический цикл (цикл с фиксированным числом повторений).

Для организации такого цикла необходимо осуществить три операции:
Инициализация - присваивание параметру цикла начального значения;
Условие - проверка условия повторения цикла, чаще всего - сравнение величины параметра с некоторым граничным значением;
Модификация - изменение значения параметра для следующего прохождения тела цикла.

Пример: Посчитать сумму чисел от 1 до введенного k

```
#include <stdio.h>
int main(void)

{
    int k;                      // объявляем целую переменную key
    int sum = 0;                // начальное значение суммы равно 0
    printf("k = ");
    scanf("%d", &k);                // вводим значение переменной k
    for(int i=1; i<=k; i++)         // цикл для переменной i от 1 до k с шагом 1
    {
        sum = sum + i;              // добавляем значение i к сумме
    }
    printf("sum = %d\n", sum);      // вывод значения суммы
    return 0;
}
```

Массивы

Массивы

При решении задач с большим количеством данных одинакового типа использование переменных с различными именами, не упорядоченных по адресам памяти, затрудняет программирование. В подобных случаях в языке Си используют объекты, называемые массивами.

Массив — это непрерывный участок памяти, содержащий последовательность объектов одинакового типа, обозначаемый одним именем.

Массив в С определяется следующим образом

<тип> <имя массива>[<размер>]; Например, **int a[100];**

Массив с именем a, который содержит сто элементов типа int.

Для получения доступа к первому элементу, в квадратных скобках указывается его номер (индекс).

Например: **a[0] = 10**;

Объявление массива

```
int a[20];                  // объявлен массив из 20-ти элементов

int a2[ ] = {1, 2, 3, 4, 5};        // число элементов массива равно 5

int ar3[5] = {};                // все равны нулю

int ar4[n];                 // n может быть определена где-то выше. Элементы не
        равны нулю
```

Вывод элементов массива на экран

```
#include <stdio.h>

int main(void)

{

    int arr[5] = {2, 4, 3, 5, 5};

    printf("%d %d %d %d %d\n",arr[0], arr[1], arr[2], arr[3], arr[4]);

    return(0);

}
```

Правильнее будет использовать циклы

Обработка и вывод массива

```
#include <stdio.h>
int main(void)
{
	int arr[100] = {0};
	for(int i = 0; i < 100; i = i + 1)
	{
		arr[i] = 2*i;
	}
	for(int i = 0; i < 100; i = i + 1)
	{
		printf("%d\t",arr[i]);
	}
	return(0);
}
```

Программа в первом цикле сохраняет в массив первую сотню чётных чисел, а во втором цикле выводит их на экран.

Многомерные массивы

Массивы могут быть и двумерными и трехмерными и, даже, n-мерными. Многомерные массивы — это массивы, у которых есть более одного индекса. Вместо одной строки элементов, многомерные массивы можно рассматривать как совокупность элементов, которые распределены по двум или более измерениям. Вот так, например, можно визуализировать двумерный массив:

```
[][][][][]
[][][][][]
[][][][][]
```

int array[3][5];

Пример использования массивов

```
#include <stdio.h>
 int main()
{
	int i, j;
	int myArray[8][8]; // объявляем массив размером 8*8 элементов
	for ( i = 0; i < 8; i++ )
	{
		for ( j = 0; j < 8; j++ )
		{
			myArray[i][j] = i * j;   // каждому элементу присваиваем значение произведения
	//текущих индексов элемента массива
		}
	}

	printf( "Вот такой массив у нас получился:\n" );
	for ( i = 0; i < 8; i++ )
	{
		 for ( j = 0; j < 8; j++ )
		{
			printf( "[%d][%d]=%d ", i, j, myArray[i][j] );
		}
		printf( "\n" );
	 }
}
```