СТРУКТУРЫ ПОВТОРЕНИЯ

5 лекция

Дисциплина «Программирование микроконтроллеров на языке С++»

В процессе программирования часто возникает необходимость повторять многократно или выполнять многократно один и тот же фрагмент программы. В этом случае используются операторы циклов.

C++ обеспечивает **три типа структур повторения**

while (пока)

do/while (цикл - пока)

for (цикл)

Виды циклов

Вычислительные процессы с многократным повторением однотипных вычислений/действий для различных значений входящих величин/данных называются циклическими, повторяющиеся участки вычислений – циклами.

for	**Цикл с предусловием, цикл с параметром**
while	**Цикл с предусловием**
do while	**Цикл с постусловием**

Например,

Цикл с предусловием – «определить сумму положительных чисел»

Цикл с постусловием – «определить сумму всех целых чисел, вводимых с клавиатуры до тех пор, пока не будет введен ноль»

Структура повторения for (ЦИКЛ)

Если необходимо создать цикл с известным числом повторений, то проще всего использовать оператор **for**. Он может быть представлен в следующем формате:

for (выражение-1; выражение-2; выражение-3)
тело_цикла;

Выражение 1 используется для установления начального значения переменных, управляющих циклом.

Выражение 2 определяет условие, при котором тело цикла будет выполняться.

Выражение 3 определяет изменение переменных, управляющих циклом после каждого выполнения тела цикла.

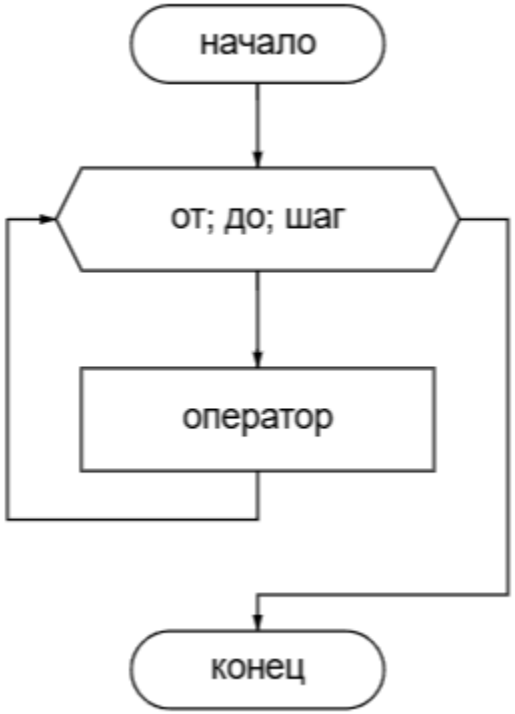


Схема выполнения оператора for

```
for (инициализация ; проверка_условия ; коррекция)
    {
    операторы;
    }
```

В таком цикле сначала выполняется инициализация. Обычно – это присвоение начального значения переменной цикла. Инициализация выполняется только один раз.

Далее происходит проверка условия. Если условие верно, то выполняются операторы внутри цикла. После этого выполняется коррекция и повторяется проверка.

Переменная, связанная с числом повторений цикла, значение которой изменяется от начального до конечного с постоянным шагом, называется **параметром цикла**, а цикл – **цикл с параметром**.

Это важно:

- проверка условия всегда выполняется в начале цикла, следовательно, тело цикла может ни разу не выполниться, если условие сразу будет ложным;
- на месте **выражение 1** и **выражение 3** может использоваться несколько выражений, записанных через запятую, и выполняемых последовательно;
- тело цикла ***for*** может быть пустым (такая форма оператора может быть использована для организации поиска).
- тело цикла ***for*** может быть составным (обязательно использовать {});
- оператор ***for*** может быть вложенным.

Примеры

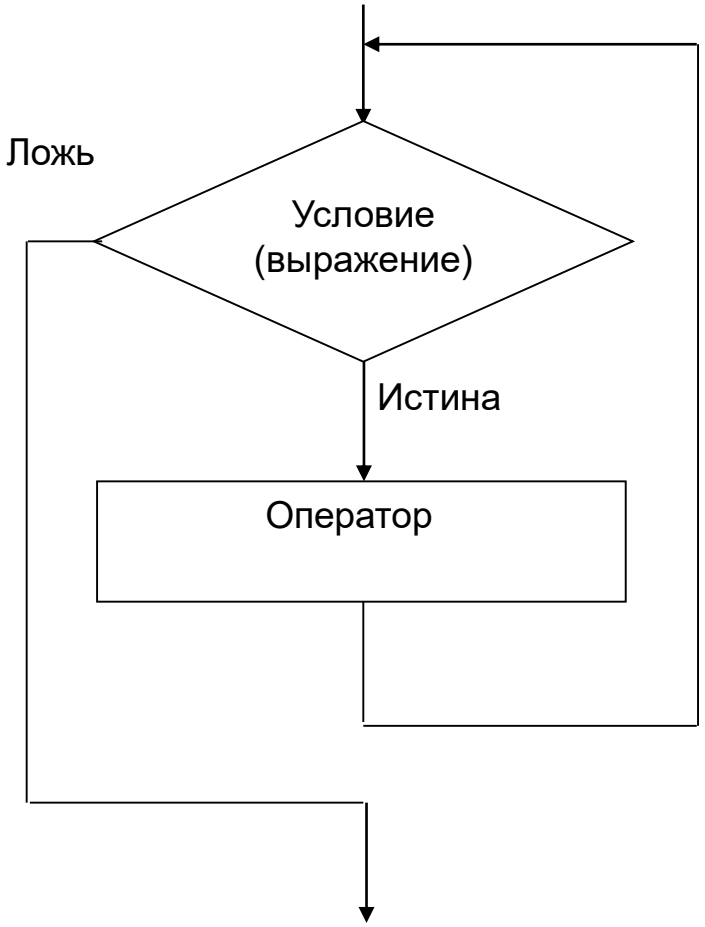
```
// Суммирование первых пяти целых чисел:
int sum=0;
for(int i=1;i<5;i++)
        {
        sum+=i;
        }

// Вывод десяти строк
for (int i = 0; i < 10; i++)
        {
        cout << i << endl;
        }
```

Структура повторения while (ПОКА)

Оператор цикла ***while*** называется циклом с предусловием и имеет следующий формат:

while (выражение) тело ;



Блок-схема оператора цикла while

Схема выполнения оператора while

```
while (условие)
        {
        операторы;
        }
```

1. Вычисляется выражение.

2. Если выражение ложно, то выполнение оператора ***while*** заканчивается и выполняется следующий по порядку оператор. Если выражение истинно, то выполняется тело оператора ***while***.

3. Процесс повторяется с пункта 1.

Это важно:

- т.к. ***while*** – это цикл с предусловием (как и в ***for*** вначале происходит проверка условия), то возможно ситуация, при которой инструкции (операторы) тела цикла ни разу не будут выполнены; поэтому оператор ***while*** удобно использовать в ситуациях, когда тело оператора не всегда нужно выполнять;
- внутри операторов ***for*** и ***while*** можно использовать локальные переменные, которые должны быть объявлены с определением соответствующих типов;
- тело цикла ***while*** может быть составным (обязательно использовать {});
- оператор ***while*** может быть вложенным.

Примеры

```
int sum=0;
while(a != 0 )
        {
        sum+=a;
        }
```

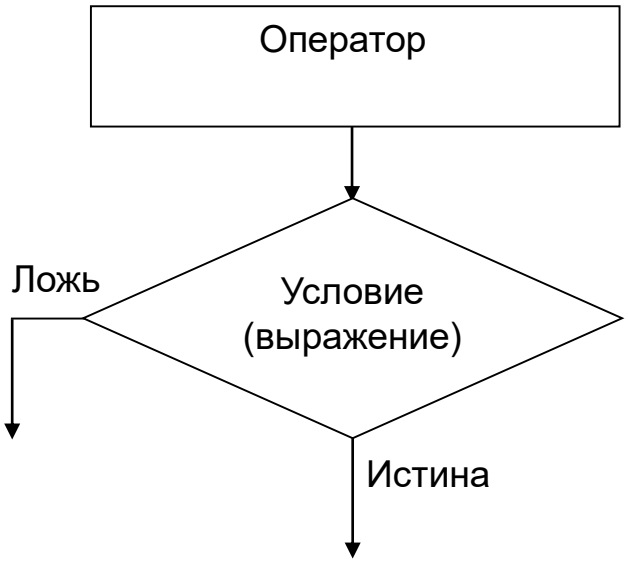
Такой цикл повторяется многократно до тех пор, пока значение условия истина. Как только это значение становится ложным, выполнение цикла прекращается.

Структура повторения do/while

Оператор цикла **do/while** называется оператором цикла с постусловием. Аналогичный вариант цикла, но с проверкой в конце условия продолжения или завершения цикла.

Формат оператора имеет следующий вид:

do тело while (выражение);



Блок-схема
оператора цикла do/while

Схема выполнения оператора do/while

```
Do
        {
        операторы;
        }
while (условие);
```

1. Выполняется тело цикла (которое может быть составным оператором).

2. Вычисляется выражение.

3. Если выражение ложно, то выполнение оператора *do while* заканчивается и выполняется следующий по порядку оператор. Если выражение истинно, то выполнение оператора продолжается с пункта 1.

Это важно:

- т.к. *do while* – это цикл с постусловием, то инструкции (операторы) тела цикла будут выполнены хотя бы один раз;
- чтобы прервать выполнение цикла до того, как условие станет ложным, можно использовать оператор *break* (см. п.7);
- операторы *do while* может быть вложенным.

Примеры

```
// возведение в степень y числа x
int main()
{
double x = 5, y =0, c;
do
        {
        y++;
        c=pow (x, y); cout << " stepen chisla =" <<c <<endl;
        }
 while (y<3);
 system ("pause");
 return 0;
}
```

Сравнение циклов

for	for (выражение-1; выражение-2; выражение-3) тело;
while	выражение-1; while (выражение-2) {тело выражение-3; }
do while	выражение-1; do {тело выражение-3; } while(выражение-2)

Управляющие операторы

- Для выхода из цикла можно использовать оператор ***break***.
- Если выполнение очередного шага цикла требуется завершить до того, как будет достигнут конец тела цикла, то можно использовать оператор ***continue***. Оператор ***continue***, как и оператор ***break***, используется только внутри операторов цикла, но в отличие от него выполнение программы продолжается не с оператора, следующего за прерванным оператором, а с начала прерванного оператора.
- Оператор ***return*** завершает выполнение функции, в которой он задан, и возвращает управление в вызывающую функцию, в точку, непосредственно следующую за вызовом.
- Использование оператора безусловного перехода ***goto*** в практике программирования на языке C/C++ не рекомендуется, так как он затрудняет понимание программ и возможность их модификаций. Оператор ***goto*** передает управление на оператор, помеченный меткой.

Управляющие операторы

Оператор	Назначение	Пример кода	Результат

break	Прерывает выполнение цикла полностью	for (int i = 1; i <= 10; i++)
		{
		if (i == 5)
		break;
		cout << i << " ";
		}
			1 2 3 4

continue	Пропускает текущую итерацию цикла и переходит к следующей	for (int i = 1; i <= 5; i++)
		{
		if (i == 3)
		continue;
		cout << i << " ";
		}
			1 2 4 5

Управляющие операторы

Оператор	Назначение	Пример кода	Результат

return	Завершает выполнение функции и возвращает управление в вызывающий код (с возможным значением)	int square(int x)
		{
		return x * x;
		}
		int main()
		{
		cout << square(5);
		}
			25

goto	Переход по метке (использовать не рекомендуется)	int i = 1;
		start:
		cout << i << " ";
		i++;
		if (i <= 5)
		goto start;
			1 2 3 4 5

Выводы

- **Циклы** (for, while, do...while) позволяют многократно выполнять однотипные действия. Они удобны для организации повторяющихся вычислений, ввода данных или обработки последовательностей. **Управляющие операторы** расширяют возможности циклов.
- При написании программ важно:
  - правильно выбирать тип цикла (например, for для фиксированного количества шагов, while для неопределённого числа повторений);
  - уметь использовать break и continue для управления процессом выполнения;
  - обеспечивать условие выхода из цикла, чтобы программа не зацикливалась бесконечно;
  - стремиться к читаемости кода, избегая избыточного использования goto.