ПЗ-02: **Программирование циклических алгоритмов**

Рассмотрим очередную алгоритмическую структуру — **цикл**.

* ***Циклом*** *называется блок кода, который для решения задачи требуется повторить несколько раз*.

Каждый цикл состоит из

* блока проверки условия повторения цикла
* тела цикла

**Цикл** выполняется до тех пор, пока блок проверки условия возвращает истинное значение.

Тело цикла содержит последовательность операций, которая выполняется в случае истинного условия повторения **цикла**. После выполнения последней операции тела цикла снова выполняется операция проверки условия повторения **цикла**. Если это условие не выполняется, то будет выполнена операция, стоящая непосредственно после цикла в коде программы.

В языке **Си** доступны следующие **виды** **циклов**:

1. **while** — цикл с предусловием;
2. **do**…**while** — цикл с постусловием;
3. **for** — параметрический цикл (цикл с заданным числом повторений);

Плюс, дополнительно циклы можно организовать с помощью:

1. **меток** и оператора **goto**;
2. **рекурсии.**

В **Си** допускаются ***вложенные циклы***, то есть когда один цикл находится внутри другого

**Цикл с предусловием while**

|  |  |
| --- | --- |
| **while (Условие)**  **{**  **БлокОпераций;**  **}**  Если Условие выполняется (выражение, проверяющее Условие, не равно нулю), то выполняется БлокОпераций, заключенный в фигурные скобки, затем Условие проверяется снова.  Последовательность действий, состоящая из проверки Условия и выполнения БлокаОпераций, 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. | **Пример на Си**: *Посчитать сумму чисел от 1 до введенного k*  *#define \_CRT\_SECURE\_NO\_WARNINGS /\* для возможности использования scanf \*/*  *#include <stdio.h>*  *int main() {*  *int k; // объявляем целую переменную key*  *int i = 1;*  *int sum = 0; // начальное значение суммы равно 0*  *printf("k = ");*  *scanf("%d", &k); // вводим значение переменной k*  *while (i <= k) // пока i меньше или равно k*  *{*  *sum = sum + i; // добавляем значение i к сумме*  *i++; // увеличиваем i на 1*  *}*  *printf("sum = %d\n", sum); // вывод значения суммы*  *getchar(); getchar();*  *return 0;*  *}* |

|  |  |  |
| --- | --- | --- |
| Пример бесконечного цикла:  **while (1)**  **{**  **БлокОпераций;**  **}** | | **while** — цикл с предусловием, поэтому вполне возможно, что тело цикла не будет выполнено ни разу если в момент первой проверки проверяемое условие окажется ложным: **while (-1)** |
| https://narodstream.ru/wp-content/uploads/2019/08/c018img00.jpg | Составляя алгоритм нашего проекта, порой нам хочется, а иногда и это и необходимо, повлиять на ход программы во время её выполнения каким-то нестандартным образом: досрочно выйти из тела условия или цикла, пропустить итерацию в цикле, а также перейти вообще в любое место программы. Для этого существуют определённые инструменты – это инструкции **break**, **continue**, а также оператор **goto**.  **break** – это инструкция, которая вызывает немедленный выход из самого внутреннего из окружающих её циклов.  То есть по данной инструкции мы не можем выйти из нескольких вложенных циклов. Если вдруг такое потребуется, то в этом случае нужно использовать оператор **goto**.  **continue** – инструкция, которая заставляет цикл, в теле которого она используется, прервать текущую итерацию и начать новую.  Данная инструкция используется реже, чем **break**.  **goto** – инструкция, позволяющая совершить переход в то место программы, в котором установлена **метка**, имя которой следует сразу же после оператора **goto**.  **Метка** – это идентификатор, за которым следует двоеточие, обозначающий место в программе, на которое можно перейти с помощью инструкции goto из любого места программы. | |
| https://narodstream.ru/wp-content/uploads/2019/08/c018img01.jpg |

|  |  |
| --- | --- |
| **Цикл с постусловием do...while**  Общая форма записи:  **do**  **{**  **БлокОпераций;**  **} while (Условие);** | Цикл **do...while** — это цикл с постусловием, где истинность выражения, проверяющего Условие проверяется после выполнения Блока Операций, заключенного в фигурные скобки. 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.

Эти три операции записываются в скобках и разделяются точкой с запятой.

Как правило, параметром цикла является целочисленная переменная.

Инициализация параметра осуществляется только один раз — когда цикл **for** начинает выполняться.

Проверка **Условия** повторения цикла осуществляется перед каждым возможным выполнением тела цикла. Когда выражение, проверяющее **Условие** становится ложным (равным нулю), цикл завершается. **Модификация** параметра осуществляется в конце каждого выполнения тела цикла. **Параметр** может как увеличиваться, так и уменьшаться.

**ЗАДАЧИ. *Решить задачи пятью способами организации циклов***(см.выше)

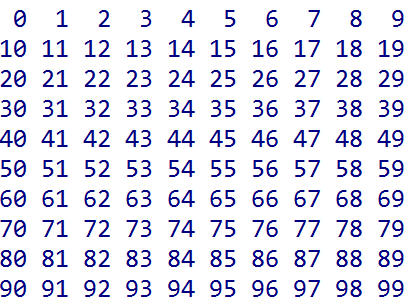
1) Необходимо вывести на экран числа от 1 до 5. *На экране должно быть: 1 2 3 4 5*

2) Необходимо вывести на экран числа от 5 до 1. *На экране должно быть: 5 4 3 2 1*

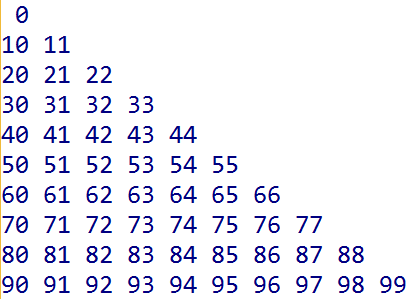
3) Необходимо вывести на экран таблицу умножения на 3.

4) Пользователь вводит любое целое положительное число. Программа суммирует все числа от 1 до введенного пользователем числа.

5) Вывести числа от 0 до 99, по 10 в каждой строке:



6) Вывести числа от 0 до 99 ниже главной диагонали:



7) Вывести числа от 0 до 99 исключая числа, оканчивающиеся на 5 или 8:

