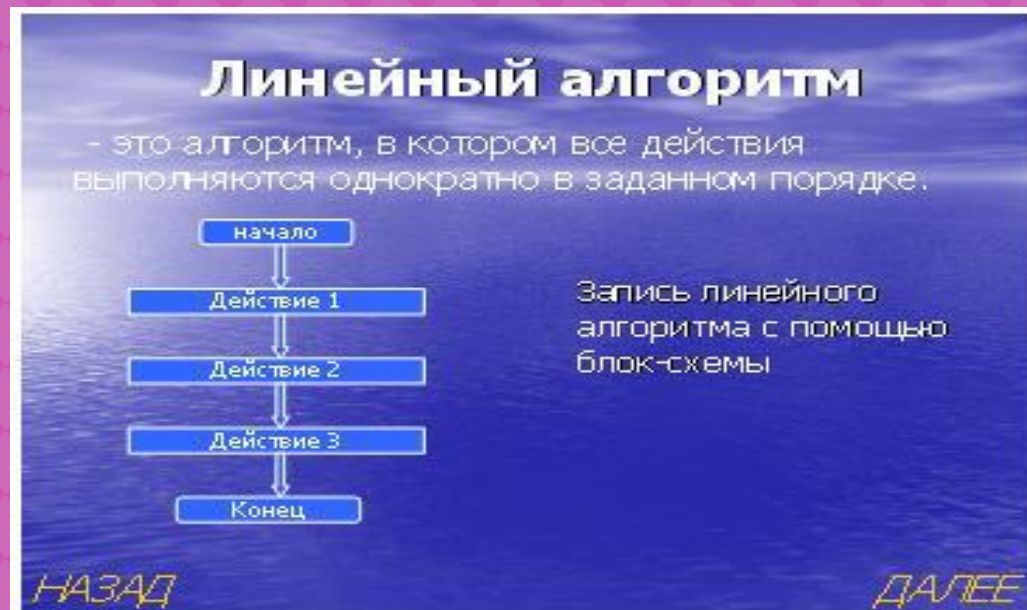


ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ

*Выполнила: Рябова Ксения
Николаевна
Студентка группы эф-201*

ЛИНЕЙНЫЙ АЛГОРИТМ

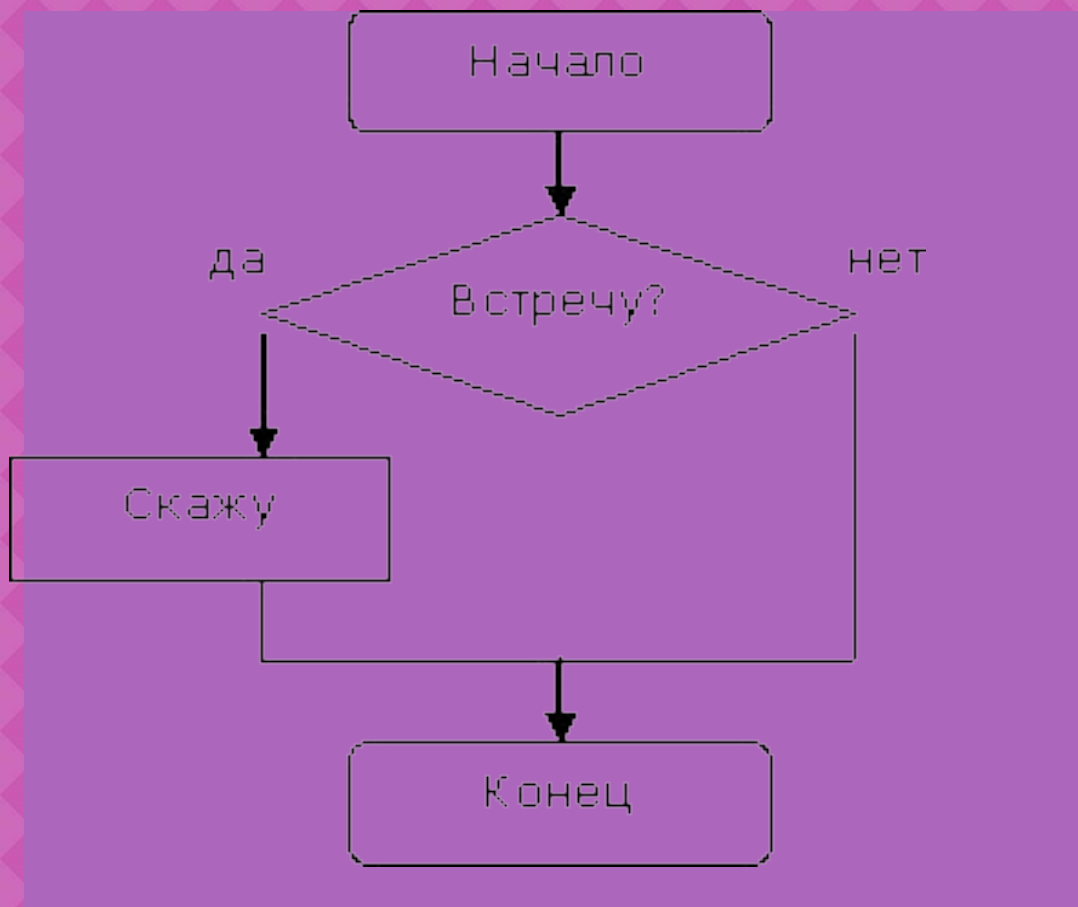
- Алгоритм - описание последовательности действий (план), строгое исполнение которых приводит к решению поставленной задачи за конечное число шагов



БЛОК СХЕМЫ И ОПИСАНИИ

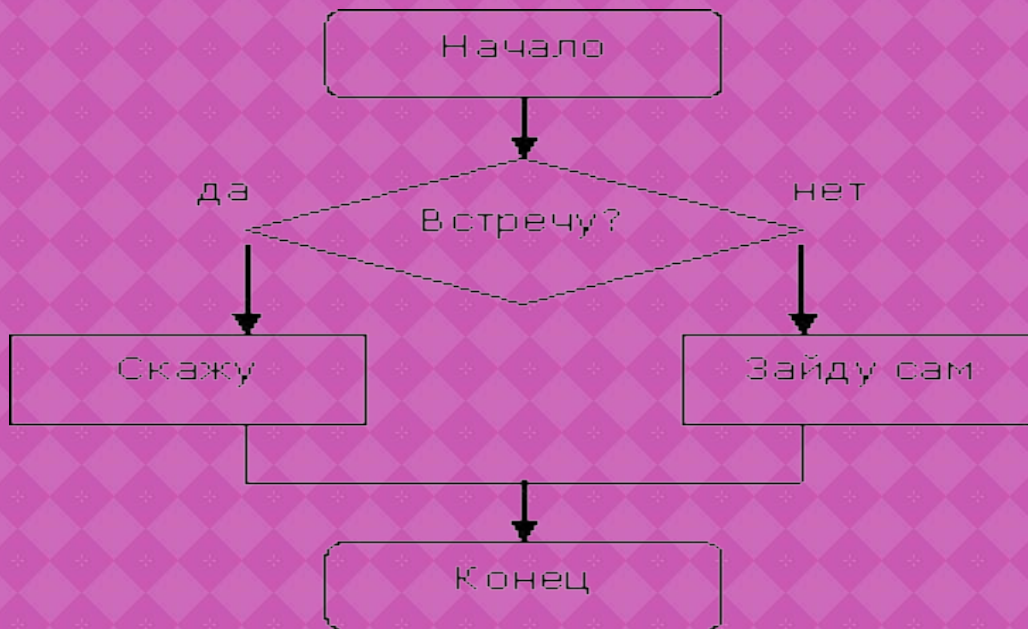
- *Блок-схема* – распространенный тип схем (графических моделей), описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности.

АЛГОРИТМИЧЕСКАЯ СТРУКТУРА «ВЕТВЛЕНИЕ». КОМАНДА ВЕТВЛЕНИЯ. ПРИМЕР НЕПОЛНОГО ВЕТВЛЕНИЯ.



АЛГОРИТМИЧЕСКАЯ СТРУКТУРА «ВЕТВЛЕНИЕ».

КОМАНДА ВЕТВЛЕНИЯ. ПРИМЕР ПОЛНОГО ВЕТВЛЕНИЯ.



КОМАНДА ВЕТВЛЕНИЯ

- *Вся программа состоит из команд (операторов). Команды бывают простые и составные (команды, внутри которых встречаются другие команды). Составные команды часто называют управляющими конструкциями. Этим подчеркивается то, что эти операторы управляют дальнейшим ходом программы.*
- *Рассмотрим запись условного оператора на языке Basic.*
- *Простая форма оператора выглядит следующим образом:*
- ```
#include <iostream> using namespace std; int main() {
 setlocale(0, ""); double num; cout << "Введите
 произвольное число: "; cin >> num; if (num < 10) { //
 Если введенное число меньше 10. cout << "Это число
 меньше 10." << endl; } else { // иначе cout << "Это число
 больше либо равно 10." << endl; } return 0; }
```

# ОПЕРАТОР SWITCH

- Оператор **switch** обеспечивает ясный способ переключения между различными частями программного кода в зависимости от значения одной переменной или выражения. Общая форма этого оператора такова:
- Результатом вычисления выражения может быть значение любого простого типа, при этом каждое из значений, указанных в операторах **case**, должно быть совместимо по типу с выражением в операторе **switch**. Все эти значения должны быть уникальными литералами. Если же вы укажете в двух операторах **case** одинаковые значения, транслятор выдаст сообщение об ошибке.
- Оператор **switch** работает следующим образом. Вычисленное значение выражения сравнивается со всеми значениями, указанными в операторах **case**. Если при этом находится оператор **case** со значением, которое совпадает со значением выражения, управление передается стоящему за ним (после двоеточия) коду. Если же значению выражения не соответствует ни один из операторов **case**, управление передается коду, расположенному после ключевого слова **default**. Отметим, что оператор **default** необязателен. В случае, когда ни один из операторов **case** не соответствует значению выражения и в **switch** отсутствует оператор **default**, выполнение программы продолжается с оператора, следующего за оператором **switch**.
- Внутри оператора **switch** (а также внутри циклических конструкций) **break** без метки приводит к передаче управления на код, стоящий после оператора **switch**. Если **break** отсутствует, после текущего раздела **case** будет выполняться следующий. Иногда бывает удобно иметь в операторе **switch** несколько смежных разделов **case**, не разделенных оператором **break**.

# ОПЕРАТОР SWITCH

- *Общий формат оператора множественного выбора - switch*

- *switch (switch\_expression)*

- *{*

- *case constant1: statement1; [break;]*

- *case constant2: statement2; [break;]*

- *case constantN: statementN; [break;]*

- *[else: statement N+1;]*

- *}*

- *switch (switch\_expression)*

- *{*

- *case constant1, case constant2: statement1; [break;]*

- *case constantN: statementN; [break;]*

- *[default: statement N+1;]*

- *}*

- *switch (switch\_expression)*

- *{*

- *case constant1: statement1; [break;]*

- *case constant2: statement2; [break;]*

- *case constantN: statementN; [break;]*

- *[default: statement N+1;]*

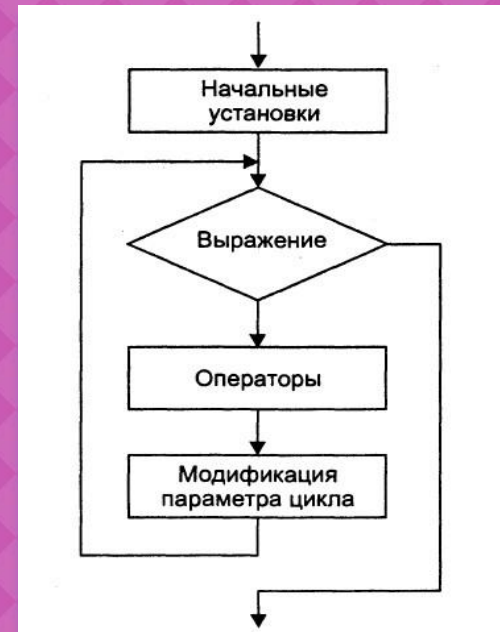


# ОПЕРАТОР WHILE

- *Оператор while*
- *Оператор цикла while называется циклом с предусловием и имеет следующий формат:*
- *while (выражение) тело ;*
- *В качестве выражения допускается использовать любое выражение языка программирования C, а в качестве тела любой оператор, в том числе пустой или составной. Схема выполнения оператора while следующая:*
- *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.*
- *Оператор цикла вида*
- *for (выражение-1; выражение-2; выражение-3) тело ;*
- *может быть заменен оператором while следующим образом:*
- *выражение-1; while (выражение-2) { тело выражение-3; }* Так же как и при выполнении оператора for, в операторе while вначале происходит проверка условия. Поэтому оператор while удобно использовать в языке программирования C 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.*

# ОПЕРАТОР ЦИКЛА С ПРЕДУСЛОВИЕМ (WHILE) В C++

- Блок схема:
- Структура цикла *while* в C++ такова:
- *While*(выражение)оператор



# ЦИКЛ DO WHILE

- Цикл *do-while*
- Как вы видели, если в начальный момент условное выражение, управляющее циклом *while*, ложно, тело цикла вообще не будет выполняться. Однако иногда желательно выполнить тело цикла хотя бы один раз, даже если в начальный момент условное выражение ложно. Иначе говоря, существуют ситуации, когда проверку условия прерывания цикла желательно выполнять в конце цикла, а не в его начале. К счастью, Java поддерживает именно такой цикл: *do-while*. Этот цикл всегда выполняет тело цикла хотя бы один раз, поскольку его условное выражение проверяется в конце цикла. Общая форма цикла *do-while* следующая:
- ```
do { // тело цикла  
} while {условие};
```

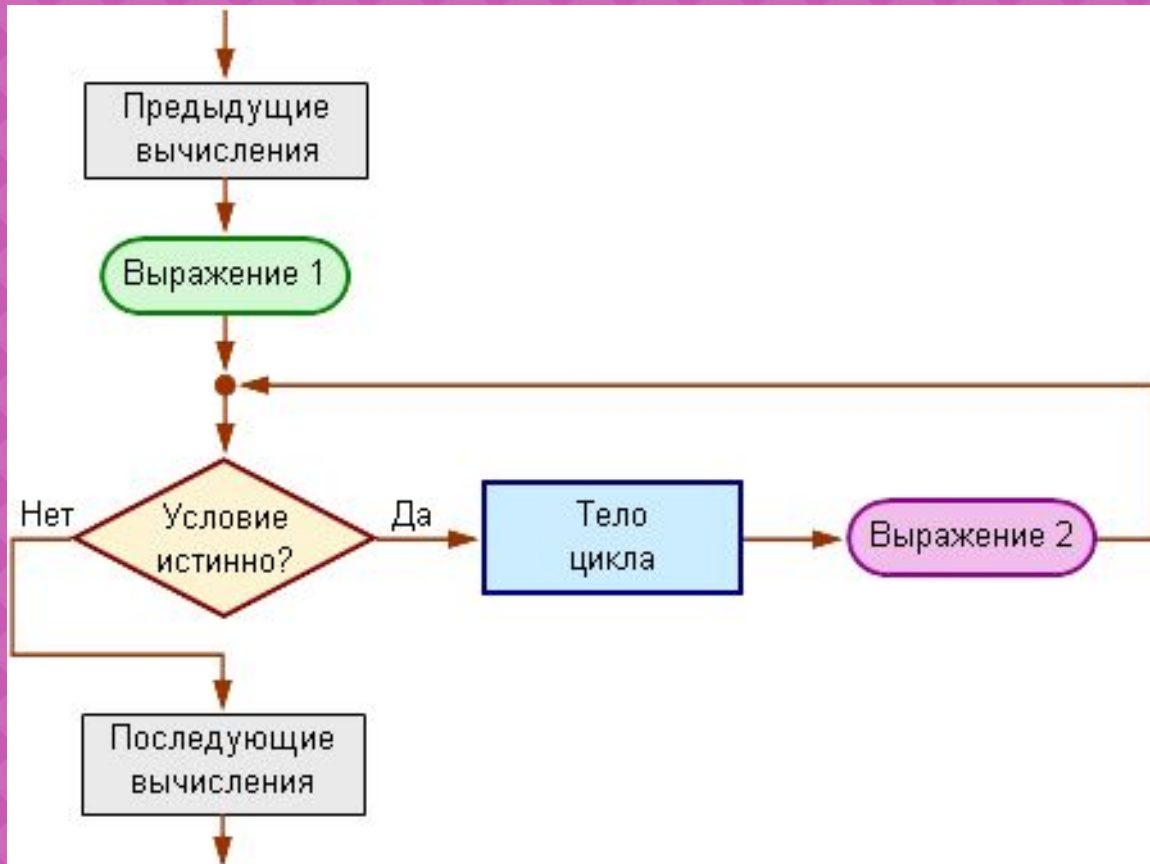
ПРИМЕР ЦИКЛА DO WHILE

```
○ #include "stdafx.h"  
○ #include <iostream>  
○ #include <ctime>  
○ using namespace std;  
○  
○ int main(int argc, char* argv[])  
○ {  
○     srand(time(0));  
○     int balance = 8; // баланс  
○     do // начало цикла do while  
○     {  
○         cout << "balance = " << balance << endl; // показать баланс  
○         int removal = rand() % 3; // переменная, для хранения вычитаемого  
○         значения  
○         cout << "removal = " << removal << endl; // показать вычитаемое значение  
○         balance -= removal; // управление условием  
○     }  
○     while ( balance > 0 ); // конец цикла do while  
○     system("pause");  
○     return 0;
```

ОПЕРАТОР ЦИКЛА FOR

- *Оператор цикла for*
- *Часто при организации цикла требуется перебирать значение счетчика в заданном диапазоне значений и с заданным шагом изменения. Например, чтобы перебрать элементы вектора (массива), нужно организовать счетчик от 1 до N с шагом 1, где N - число элементов вектора. Чтобы вычислить сумму ряда, также задается счетчик от a до b с требуемым шагом изменения step. И так далее. В связи с тем, что подобные задачи часто встречаются в практике программирования, для их реализации был предложен свой оператор цикла for, который позволяет проще и нагляднее реализовывать цикл со счетчиком.*

БЛОК СХЕМА



ПРИМЕР ЦИКЛА

- `#include <iostream>`
`using namespace std;`
`void main()`
`{`
`for(int i=1; i<10; i++)`
- Задача: Вывести на экран числа от 1 до 10.
- `{cout<<i<<endl;}`
`}`