# 1. Этапы решения задач на ЭВМ

В процессе разработки программ с использованием процедурного подхода можно

выделить следующие этапы:

* постановка задачи – определение требований к программному продукту;
* анализ – осуществление формальной постановки задачи и определение методов ее решения;
* проектирование – разработка структуры программного продукта, выбор структур
* реализация – составление программы на выбранном языке программирования, ее
* тестирование и отладка.
* модификация – выпуск новых версий программного продукта.

Разберем подробнее каждый из них:

**Постановка задачи**. Изначально устанавливают набор выполняемых функций, а также характеристики и перечень исходных данных. Затем определяют перечень результа-тов, их характеристики и способы представления (таблицы, диаграммы, графики). В дополнение ко всему определяют среду функционирования финального продукта: конкретная комплектация, параметры технических средств, версия операционной системы. При работе с обработкой и хранением информации, необходимо учитывать возможные проблемы при сбоях оборудования и энергоснабжения.

**Анализ.** На этом этапе по результатам анализа условия задачи выбирают математические абстракции, *адекватно*, т.е. с требуемой точностью и полнотой, представляющие исходные данные и результаты, строят модель задачи и определяют метод преобразования исходных данных в результат (*метод решения задачи*).

Определив методы решения, следует для некоторых вариантов исходных данных вручную или на калькуляторе подсчитать ожидаемые результаты. Эти данные в дальнейшем будут использованы при тестировании программы.

**Проектирование.** Существует два вида проектирование: *логическое* и *физическое.* В логическом проектировании не учитывается особенности среды, в которой будет выполняться программа (программные и технические средства компьютера), в отличии от физического, где все эти параметры необходимо учесть.

*Логическое проектирование*. Логическое проектирование при процедурном подходе предполагает детальную проработку последовательности действий будущей программы. Его начинают с определения структуры будущего программного продукта: отдельная программа или программная система, состоящая из нескольких взаимосвязанных программ. Затем переходят к разработке алгоритмов программ.

*Физическое проектирование*. При выполнении физического проектирования осуществляют привязку разрабатываемого программного обеспечения к имеющемуся набору технических и программных средств.

**Реализация.** Разработанные алгоритмы реализуют, составляя по ним текст программы с использованием конкретного языка программирования. Язык может быть определен в техническом задании, а может выбираться исходя из особенностей конкретной разработки.

Вначале осуществляют ввод программы в компьютер. Для ввода используют специальную программу - *текстовый редактор*, с помощью которого создают файл, содержащий текст программы.

Затем программу необходимо перевести в последовательность машинных команд (машинный код). Для этого запускают специальную программу-переводчик – *компилятор*. В процессе разбора и преобразования программы *компилятор* может обнаружить ошибки. Тогда он аварийно завершает работу, выдав программисту сообщения об *ошибках компиляции*. Для исправления этих ошибок обычно достаточно внимательно изучить соответствующий фрагмент с учетом текста сообщения об ошибке и внести требуемое изменение в программу.

**Тестирование.** Помимо указанных выше типов ошибок, обнаруживаемых автоматически компилятором, компоновщиком или операционной системой, существует еще группа очень опасных логических ошибок. Наличие таких ошибок в программе приводит к выдаче неправильных результатов. Для их обнаружения параллельно с отладкой программы осуществляют ее *тестирование*. *Тестированием* называют процесс выполнения программы при различных тестовых наборах данных с целью *обнаружения ошибок.*

**Модификация.** В большинстве случаев разработанное программное обеспечение через некоторое время требует обновления, и, следовательно, в процессе разработки программного продукта необходимо учитывать возможность его модификации.

Причинами выпуска новых версий являются:

* необходимость исправления ошибок, выявленных в процессе длительной эксплуатации;
* необходимость совершенствования, например, улучшения интерфейса или расширения состава выполняемых функций;
* изменение среды (появление новых технических средств и/или программных продуктов).

# 2. Понятие алгоритма. Формы представления алгоритмов.

*Алгоритмом* называют формально описанную последовательность действий, кото-рые необходимо выполнить для получения требуемого результата, при разработке программ.

Любой алгоритм обладает следующими свойствами: детерминированностью, массовостью, результативностью, дискретностью и конечностью.

*Детерминированность* (определенность, обусловленность) означает, что набор указаний алгоритма должен быть однозначно и точно понят любым исполнителем. Это свойство определяет однозначность результата работы алгоритма при записи исходных данных.

*Массовость* алгоритма предполагает возможность варьирования исходных данных в определенных пределах. Свойство массовости определяет пригодность использования алгоритма для решения множества задач данного класса.

*Результативность.* Предполагает, что для любых допустимых исходных данных он должен через конечное число шагов (или итераций) завершить свою работу.

*Дискретность.* Допускает разбиение определенного алгоритмического процесса на отдельные элементарные этапы.

*Конечность* алгоритма означает, что он должен выполняться за конечное время.

Существуют три последовательности действий (вычислений): линейная, разветвленная и циклическая структуры.

В *линейной* структуре процесса вычислений предполагается, что для получения ре-зультата необходимо выполнить некоторые операции в определенной последователь-ности.

В *разветвленной* структуре процесса конкретная последовательность операций зависит от значений одного или нескольких параметров.

*Циклическая* структура процесса вычислений предполагает, что для получения результата некоторые действия необходимо выполнить несколько раз.

Алгоритмы циклической структуры делятся на три группы:

* циклические процессы, для которых количество повторений известно - *счетные циклы* или *циклы с заданным количеством повторений*;
* циклические процессы, завершающиеся по достижении или нарушении некоторых условий - *итерационные циклы*;
* циклические процессы, из которых возможны два варианта выхода: выход по завершении процесса и досрочный выход по какому-либо дополнительному условию - *поисковые циклы*.

Формальное описание алгоритмов осуществляют с использованием схем алгоритмов и псевдокодов.

**Таблица 1**. Основные обозначения блоков алгоритмов

и программ

|  |  |  |
| --- | --- | --- |
| Название блока | Обозначение | Назначение блока |
| Терминатор |  | Начало, завершение программы или подпрограммы |
| Процесс |  | Обработка данных (вычисления, пересылки и т. п.) |
| Данные |  | Операции ввода-вывода |
| Решение |  | Ветвления, выбор, итерационные и поисковые циклы. |
| Подготовка |  | Счетные циклы |
| Границы цикла | C:\Users\Семейка\Desktop\Границы цикла.png | Любые циклы |
| Предопределенный процесс |  | Вызов процедур |
| Соединитель |  | Маркировка разрывов линий |
| Комментарий | C:\Users\Семейка\Desktop\Комментарий.png | Пояснения к операциям |

Для записи любого сколь угодно сложного алгоритма достаточно трех базовых структур:

* *следование* - обозначает последовательное выполнение действий
* *ветвление* - соответствует выбору одного из двух вариантов действий
* *цикл-пока* - определяет повторение действий, пока не будет нарушено условие, выполнение которого проверяется в начале цикла.

Помимо базовых структур используют три дополнительные структуры, производные от базовых:

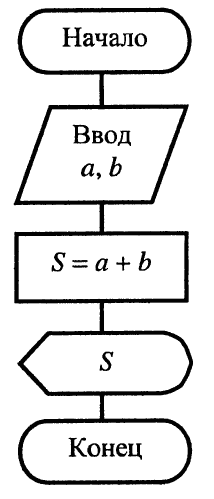
* *выбор* - выбор одного варианта из нескольких в зависимости от значения некоторой величины;
* *цикл-до* - повторение некоторых действий до выполнения заданного условия, проверка которого осуществляется после выполнения действий в цикле;
* цикл *с заданным числом повторений (счетный цикл)* – повторение некоторых действий указанное число раз.

*Псевдокод* - описание алгоритма, которое базируется на тех же основных структурах, что и структурные схемы алгоритма.

# 3. Линейные алгоритмы.

*Линейной* называют алгоритмическую конструкцию, реализованную в виде последовательности действий (шагов), в которой каждое действие (шаг) алгоритма выполняется ровно один раз, причем после каждого i-го действия (шага) выполняется (i+1)-е действие (шаг), если i-е действие — не конец алгоритма.

Пример. Опишем алгоритм сложения двух чисел на псевдокоде в виде блок-схемы.

Псевдокод:

1. Ввод двух чисел a, b.

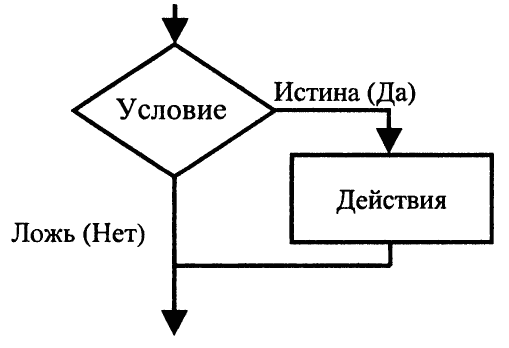
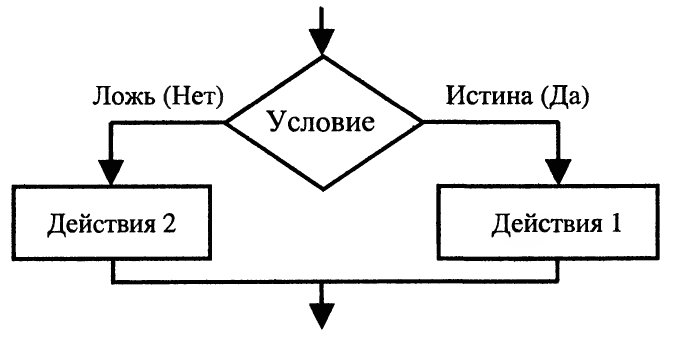
2. Вычисляем сумму S = а + b.

3. Вывод S.

4. Конец.

# 4. Разветвляющиеся вычислительные процессы.

*Разветвляющейся* (или *ветвящейся*) называется алгоритмическая конструкция, обеспечивающая выбор между двумя альтернативами в зависимости от значения входных данных. При каждом конкретном наборе входных данных разветвляющийся алгоритм сводится к линейному. Различают неполное (*если — то*) и полное (*если — то — иначе*) ветвления. Полное ветвление позволяет организовать две ветви в алгоритме (то или иначе), каждая из которых ведет к общей точке их слияния, так что выполнение алгоритма продолжается независимо от того, какой путь был выбран. Неполное ветвление предполагает наличие некоторых действий алгоритма только на одной ветви (*то*), вторая ветвь отсутствует, т.е. для одного из результатов проверки никаких действий выполнять не надо, управление сразу переходит к точке слияния.

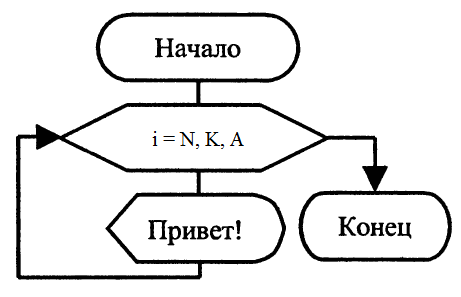


**Рис**. Полное ветвление **Рис**. Неполное ветвление

**5. Циклические алгоритмы**

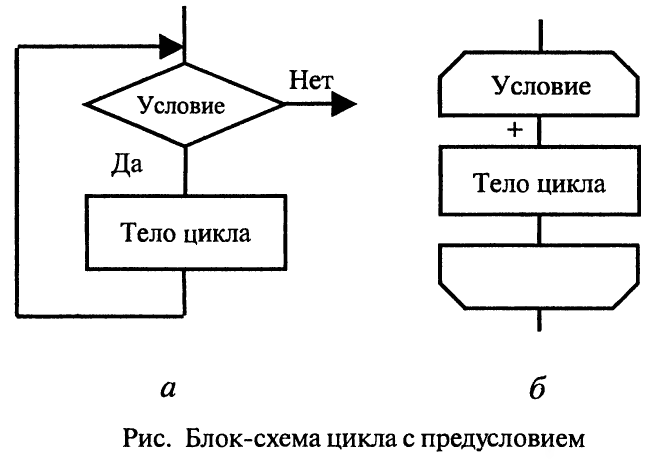
*Циклической* (или *циклом*) называют алгоритмическую конструкцию, в которой некая, идущая подряд группа действий (шагов) алгоритма может выполняться несколько раз, в зависимости от входных данных или условия задачи. Группа повторяющихся действий на каждом шагу цикла называется телом цикла. Любая циклическая конструкция содержит в себе элементы ветвящейся алгоритмической конструкции.

Существует три типа циклических алгоритмов: *цикл с параметром* (который называют *арифметическим* циклом), *цикл с предусловием* и *цикл с постусловием* (их называют итерационными).

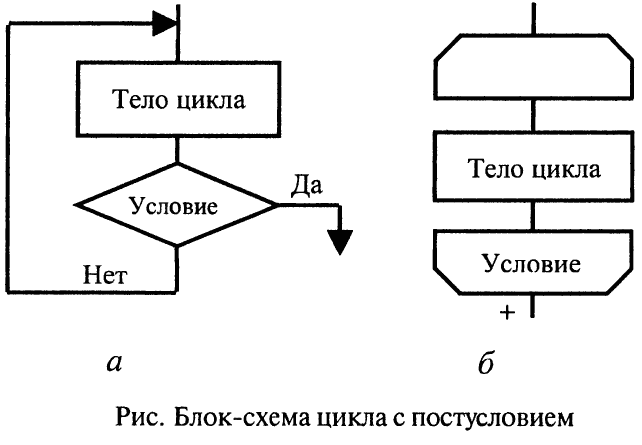
В *арифметическом* цикле число его шагов (повторений) однозначно определяется правилом изменения параметра, которое задается с помощью *начального (N)* и *конечного (К)* значений параметра и *шагом (А)* его изменения. Т.е., на первом шаге цикла значение параметра равно N, на втором — N + h, на третьем — N + 2h и т.д. На последнем шаге цикла значение параметра не больше К, но такое, что дальнейшее его изменение приведет к значению, большему, чем К.

**Рис.** Арифметический цикл

*Цикл с предусловием.* Количество шагов цикла заранее не определено и зависит от входных данных задачи. В данной циклической структуре сначала проверяется значение условного выражения (условие) перед выполнением очередного шага цикла. Если значение условного выражения истинно, исполняется тело цикла. После чего управление вновь передается проверке условия и т.д. Эти действия повторяются до тех пор, пока условное выражение не примет значение ЛОЖЬ. При первом же несоблюдении условия цикл завершается.



*Цикл с постусловием.* В циклической конструкции с постусловием заранее не определено число повторений тела цикла, оно зависит от входных данных задачи. Тело цикла с постусловием всегда будет выполнено хотя бы один раз, после чего проверяется. В этой конструкции тело цикла будет выполняться до тех пор, пока значение условного выражения ложно. Как только оно становится истинным, выполнение команды прекращается.



**6. Компьютер как исполнитель алгоритмов. Программа как изображение алгоритма в терминах команд, управляющих работой компьютера.**

Компьютер является автоматическим исполнителем алгоритмов. Исполнитель способен выполнить только ограниченное количество команд. Поэтому алгоритм разрабатывается и детализируется так, чтобы в нем присутствовали только те команды и конструкции, которые может выполнить исполнитель. Исполнитель, как и любой объект, находится в определенной среде и может выполнять только допустимые в нем действия. Если исполнитель встретит в алгоритме неизвестную ему команду, то выполнение алгоритма прекратится.

В свою очередь алгоритм, записанный на «понятном» компьютеру языке программирования, называется программой.

Для решения задачи с помощью компьютера, сначала разрабатывают алгоритм ее решения, отлаживают его, а потом, уже отлаженный алгоритм изображают в терминах команд, понятных ЭВМ.

Известно, что ЭВМ работает с двоичным кодом (в виде набора единиц и нулей), а команды на различных языков программирования в диалоговом окне набираются в виде команд (операторов) и символов на данном языке. Возникает необходимость преобразования языка высокого уровня на язык, понятный ЭВМ – машинный код. Эту операцию производит транслятор. Исходный модуль преобразуется в объектный модуль. А исполняемая эту операцию программа называется транслятором.