Основы алгоритмизации

Алгоритм

Алгоритм — это последовательность действий, которые необходимо выполнить для решения определенной задачи или получения результата.

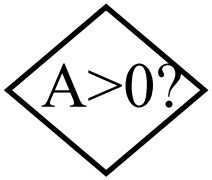
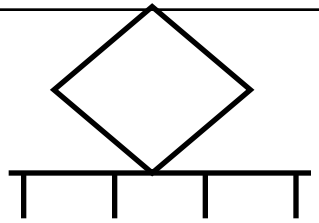
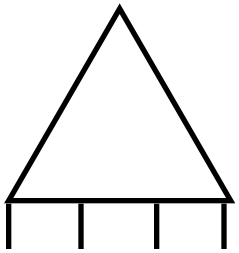
Алгоритм обладает **обязательными свойствами**.

- **Дискретность** — алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов. Новое действие исполняется только после того, как закончилось исполнение предыдущего.
- **Определенность** — каждое действие алгоритма должно пониматься исполнителем однозначно (нарисуйте корову с закрытыми глазами).
- **Результативность (конечность)** — алгоритм должен приводить к решению задачи за конечное число шагов.
- **Массовость** — алгоритм решения задачи разрабатывается в общем виде, то есть он должен быть применим для некоторого класса задач, различающихся только исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

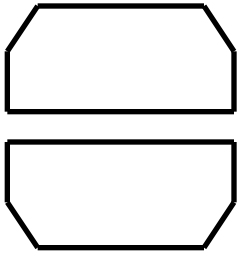
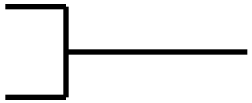
Изображение алгоритмов с помощью блок-схем

Наименование	Обозначение	Функция
Начало-конец (пуск-остановка)		Обозначает начало и конец программы или подпрограммы
Действие		Выполнение одной или нескольких операций
Ввод/вывод		Операции ввода или вывода даных
Предопределенный процесс		Вызов подпрограммы

Изображение алгоритмов с помощью блок-схем

Наименование	Обозначение	Функция
Условие		Проверка значение выражение и разветвление алгоритма
Множественные условия	 Или 	Проверка значение выражение и разветвление алгоритма на более, чем три ветви.

Изображение алгоритмов с помощью блок-схем

Наименование	Обозначение	Функция
Границы цикла		Обозначает начало и конец цикла, где указываются условия цикла и приращения индексов.
Соединитель		Нужен для соединения линий алгоритма при их разрыве или переходе на другую страницу
Комментарий		Применятся для описания набора блоков

Язык структурного программирования

Разработан в 1970 годах Эдсгером Вибе Дейкстрой.

Основная идея – отказ от оператора перехода (Goto) в языках программирования высокого уровня, так как они могут создавать проблемы с чтением программы и с её выполнением (не выполняется очистка памяти в точке «ухода» и инициализация переменных в точке «прихода»).

Язык включает 3 основные операции:

1. Действие
2. Условие — ЕСЛИ (условие) ТО
   ИНАЧЕ
3. Цикл — ПОКА (условие)
   перечень действий

Язык структурного программирования

При записи алгоритма содержимое условий и циклов должно записываться с отступом, образуя структурированный уровневый вид:

```
действия
ЕСЛИ (условие) ТО
    действия
    ПОКА (условие)
            действия
    действия
ИНАЧЕ
    действия
```

Принципы структурного программирования

1. Следует отказаться от использования оператора безусловного перехода.

2. Любая программа строится на основе трех базовых конструкций: действие, условие, цикл.

3. Базовые конструкции могут быть вложены друг друга в произвольной форме и неограниченном количестве.

4. Повторяющиеся фрагменты желательно оформлять в виде подпрограмм.

5. Логически законченные группы инструкций желательно объединять в блоки.

6. Все перечисленные конструкции должны иметь один вход и один выход.

7. Разработка программы ведется пошагово, используя метод «сверху-вниз».

Метод «сверху-вниз»

При разработке алгоритма или программы, первоначально реализуется основной управляющий алгоритм без детализации функциональных элементов, которые заменяются функциями-заглушками, возвращающими постоянное значение.

По мере разработки каждая функция-заглушка реализуется своим алгоритмом, вызывающим, при необходимости, другие функции-заглушки.

Разработка заканчивается когда все функции-заглушки реализованы в виде набора базовых конструкций.

Метод «снизу-вверх»

Предполагает обратный путь разработки: Сначала реализуются элементарные функции из которых собираются более сложные конструкции. В настоящее время активно используется, так как существует большое количество библиотек, содержащих готовые функции.

Элементарные алгоритмы

Элементарная программа

Запускаем среду «Исполнители»:
C:\robowin&logic\robowin\robot.exe
Закрываем блокноты.
В теле программы (между фигурными скобками) пишем:

```
целые А,Б,В;
вывод "введите А";
ввод А;
вывод "введите Б";
ввод Б;
В=А+Б;
выводстр "А+Б=",В;
```

Запускаем программу кнопкой F9.

Поиск минимума (максимума)

Обобщенный алгоритм:

```
min=A[0]
i=1
ПОКА (i<n)
    ЕСЛИ (min>A[i]) ТО
        min=A[i]
    i=i+1
```

Чем будет отличаться алгоритм поиска максимума?

Поиск минимума (максимума)

Правильный обобщенный алгоритм:

ЕСЛИ (n>0) ТО
  min=A[0]
  i=1
  ПОКА (i<n)
    ЕСЛИ (min>A[i]) ТО
      min=A[i]
    i=i+1

Поиск второго максимума

Алгоритм:

```
ЕСЛИ (n>1) ТО
  ЕСЛИ (A[0]>A[1]) ТО
    max_b=A[0]
    max_m=A[1]
  ИНАЧЕ
    max_b=A[1]
    max_m=A[0]
  i=2
  ПОКА (i<n)
    ЕСЛИ (A[i]>max_b) ТО
      max_m=max_b
      max_b=A[i]
    ИНАЧЕ
      ЕСЛИ (A[i]>max_m) ТО
        max_m=A[i]
    i=i+1
```

Поиск третьего максимума

Как будет выглядеть алгоритм поиска третьего максимума?
А четвертого минимума?
А если в массиве элементы повторяются?

Более простой способ – отсортировать массив, а потом в нем искать необходимый элемент.

Алгоритмы сортировки

Самый простой алгоритм – «Пузырек».

```
i=0
ПОКА (i<n-1)
    j=i+1
    ПОКА (j<n)
        ЕСЛИ (A[i]>A[j]) ТО //сортируем по возрастанию
            врем=A[i]
            A[i]=A[j]
            A[j]=врем
        j=j+1
    i=i+1
```

Программа сортировки

```
//Часть 1: ввод массива:
целые И,Ж,К;
целые А[5];

И=0;
пока (И<5)
{
    вывод "Введите А[",И,"]";
    ввод А[И];
    И=И+1;
}
```

Программа сортировки

```
//Часть 2: сортировка массива:
И=0;
пока (И<4)
{
    Ж=И+1;
    пока (Ж<5)
    {
        если (А[И]>А[Ж])
        {
            К=А[И];
            А[И]=А[Ж];
            А[Ж]=К;
        }
        Ж=Ж+1;
    }
    И=И+1;
}
```

Программа сортировки

```
//Часть 3: вывод массива:

И=0;
пока (И<5)
{
    выводстр "А[",И,"]=", А[И];
    И=И+1;
}
```