



<http://0861.ru>

Парадигмы программирования

Лекция 2

Императивное (процедурное) программирование

ст. препод. каф. ПОВТиАС
Голубничий Артем Александрович
artem@golubnichij.ru

Абакан, 2019

Структура занятия

- понятие алгоритма и свойства алгоритмов;
- представление алгоритмов;
- машина Тьюринга;
- понятия процедурного программирования;
- архитектура фон Неймана, принципы фон Неймана;
- безусловный переход goto (jump);
- языки процедурного программирования.

Алгоритм и свойства алгоритмов

Алгоритм – набор инструкций, описывающих порядок (последовательность) действий исполнителя для достижения некоторого результата.



Аль-Хорезми

Свойства алгоритма

Дискретность

Детерминированность

Понятность

Завершаемость

Массовость

Результативность

Алгоритм и свойства алгоритмов

- **дискретность** — алгоритм должен представлять процесс решения задачи как выполнение некоторых простых шагов;
- **детерминированность** — в каждый момент времени следующий шаг работы однозначно* определяется состоянием системы;
- **понятность** — алгоритм должен включать только те команды, которые доступны исполнителю и входят в его систему команд;
- **завершаемость** — при правильно заданных данных алгоритм должен завершать работу и выдавать результат за определенное число шагов;
- **массовость** — алгоритм должен быть применим к разным наборам начальных данных;
- **результативность** — завершение алгоритма определенными результатами.

* — существуют стохастические (вероятностные) алгоритмы

Представление алгоритмов



Представление алгоритмов

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

Словесная форма — последовательность действий в виде описаний на естественном языке.

Псевдокод — компактный язык описания алгоритмов, использующий ключевые слова императивных языков программирования, но опускающий несущественные подробности и специфический синтаксис.

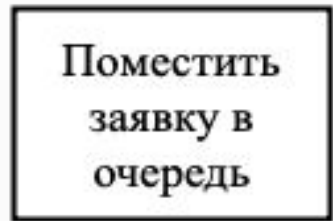
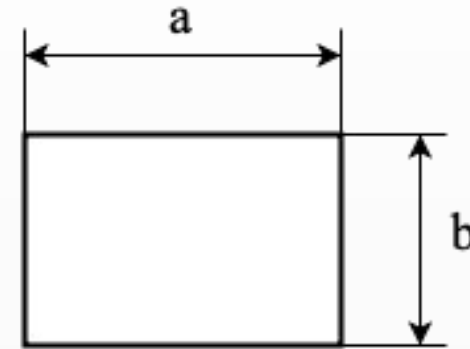
Программный код — алгоритм предназначен для исполнения техническим устройством.

Блок-схемы

Элементы строятся на основании параметров a и b , связанных следующим соотношением $2a = 3b$.

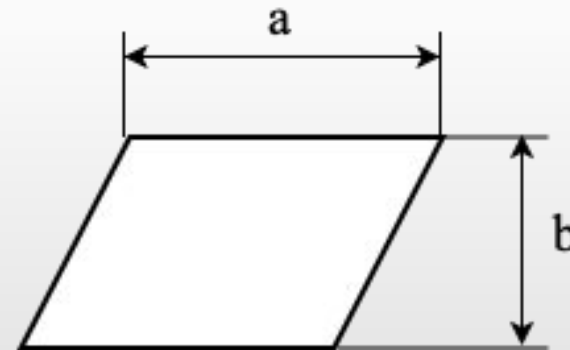
Действие

Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации и т.д.).



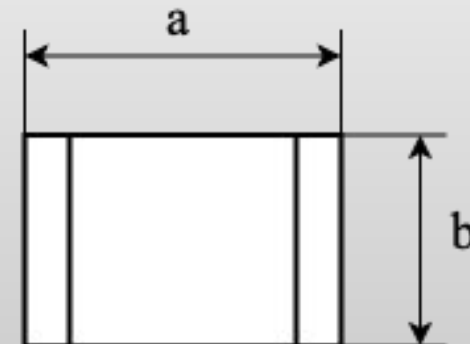
Данные (ввод/вывод)

Символ отображает данные (носитель не определен). Включает ввод или вывод данных.



Предопределенный процесс (функция)

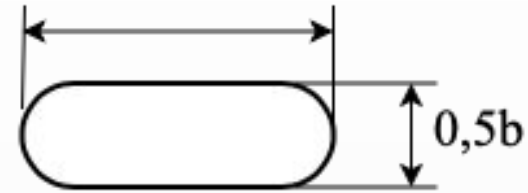
Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы (вызов процедуры или функции).



Блок-схемы

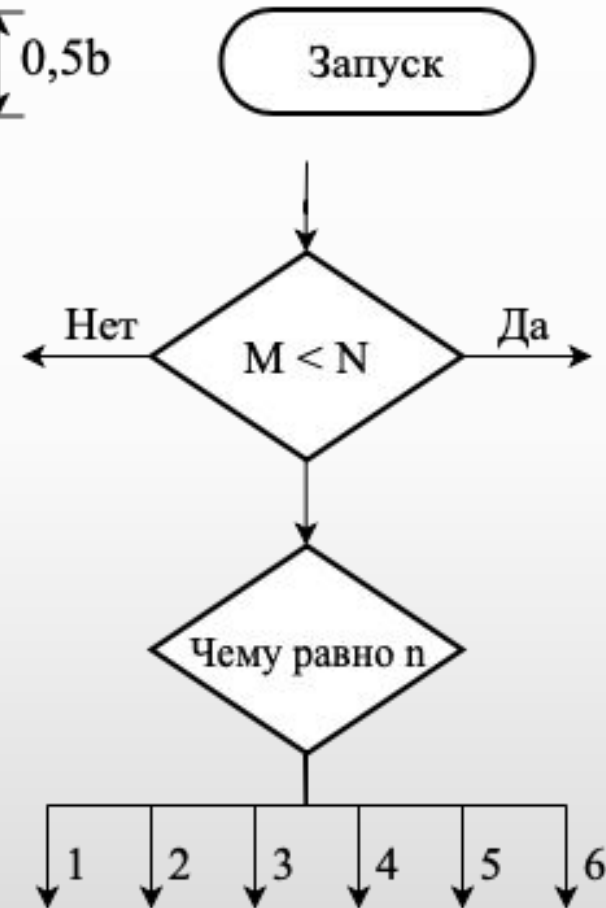
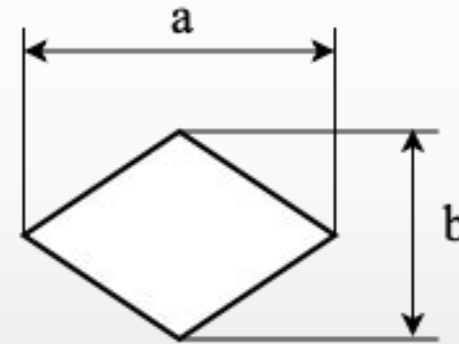
Ограничитель

Символ отображает вход из внешней среды или выход во внешнюю среду (начало, конец, перезапуск, ошибка и выход, исключение)



Вопрос (условие или решение)

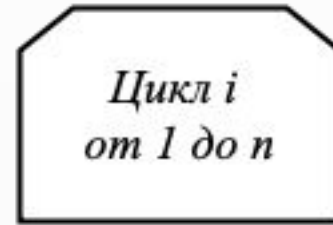
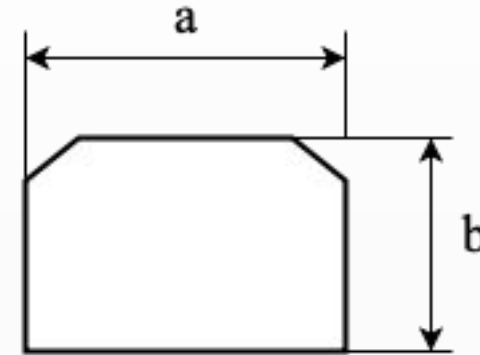
Символ отображает функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, после вычислений активизируется только один из них. Результат вычисления записывается рядом с линией пути. (if предполагает два выхода: True и False, case – множество выходов).



Блок-схемы

Цикл

Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или в в конце в зависимости от расположения операции, проверяющей условие.



Псевдокод (на учебном алгоритмическом языке)

Служебное слово	Значение	Служебное слово	Значение	Служебное слово
алг	заголовок алгоритма	длин	длина	и
нач	начало алгоритма	нц	начало цикла	или
кон	конец алгоритма	кц	конец цикла	не
арг	аргумент	надо	цель выполнения	да
рез	результат	если	реализация ветвления	нет
цел	целый	то	реализация ветвления	при
сим	символьный	иначе	реализация ветвления	ввод
лит	литерный	пока	проверка условия	вывод
лог	логический	для	работа со счетчиком	
вещ	вещественный	от	работа со счетчиком	
таб	таблица	до	работа со счетчиком	

Представление алгоритмов (расчет площади круга)

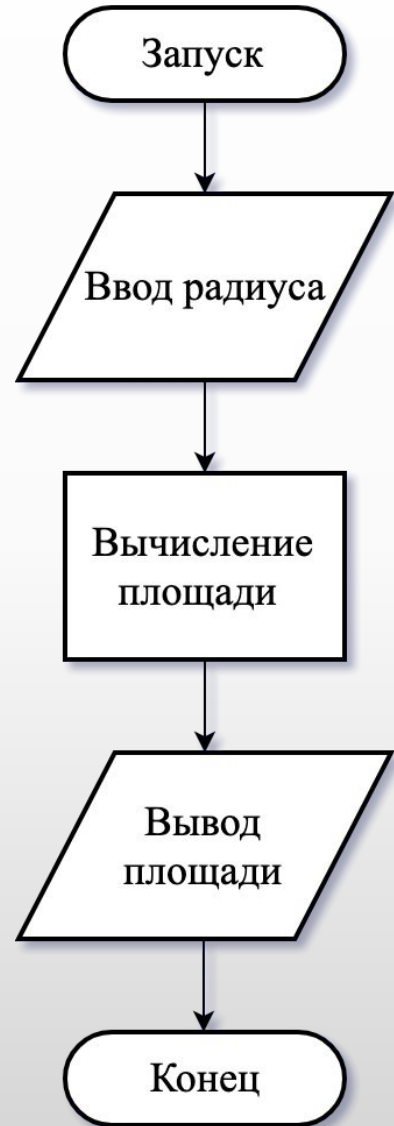
1. Прочитать радиус круга
2. Вычислить площадь используя формулу: $\text{площадь} = \text{радиус}^2 * \pi$
3. Вывести результат

Словесное описание

```
r <- as.double(readline("Введите радиус в см: "))  
s <- r^2 * pi  
cat("Площадь круга =", s, "см^2")
```

Код на языке R

Представление алгоритмов (расчет площади круга)



Блок-схема

```
алг Вычислить площадь круга (арг вещ r, рез вещ S)
    дано | r > 0, pi = 3.14
    надо | S
нач
    ВВОД r
    S = r^2 * pi
    ВЫВОД "Площадь круга = ", S, " см^2"
кон
```

Псевдокод

Машина Тьюринга

Машина Тьюринга — абстрактный исполнитель (абстрактная вычислительная машина). Была предложена Аланом Тьюрингом в 1936 году для формализации понятия алгоритма.



Машина Тьюринга является расширением конечного автомата и способна имитировать всех исполнителей (с помощью задания правил перехода), каким-либо образом реализующих процесс пошагового вычисления, в котором каждый шаг вычисления достаточно элементарен.

Императивное программирование

Императивное программирование – парадигма, для которой характерны следующие особенности:

- исходный код построен из команд (инструкций);
- инструкции выполняются последовательно;
- данные, получаемые при выполнении текущих инструкций, могут читаться из памяти последующими инструкциями;
- данные, полученные при выполнении инструкций, могут записываться в память.

Переменная – это именованная область памяти для хранения данных, которые могут изменяться в процессе исполнения программы.

Присваивание

Присваивание — механизм связывания, позволяющий динамически изменять связь имен объектов данных (как правило, переменных) с их значениями. На физическом уровне результат операции присвоения состоит в проведении записи и перезаписи ячеек памяти или регистров процессора.

<выражение_слева> <оператор_присваивания> <выражение_справа>

Присваивание

Выражение слева — часть конструкции присваивания, отвечающая за местоположение объекта данных.

Выражение справа — часть конструкции присваивания, отвечающая за величину, присваиваемую к объекту данных.

Оператор присваивания — часть конструкции присваивания, отвечающая за связывание значения с объектом данных.

```
s <- r^2 * pi  
iris[iris[,2] > 3, 2] <- NA
```

Примеры присваивания (код на языке R)

Алгоритм присваивания:

1. Вычисление левостороннего значения (первый операнд);
2. Вычисление правостороннего значения (второй операнд);
3. Присваивание правостороннего значения левостороннему;
4. Возвращение вычисленного правостороннего значения.

Понятия процедурного программирования

Процедура — законченная точно определенная последовательность операций для решения отдельной задачи.

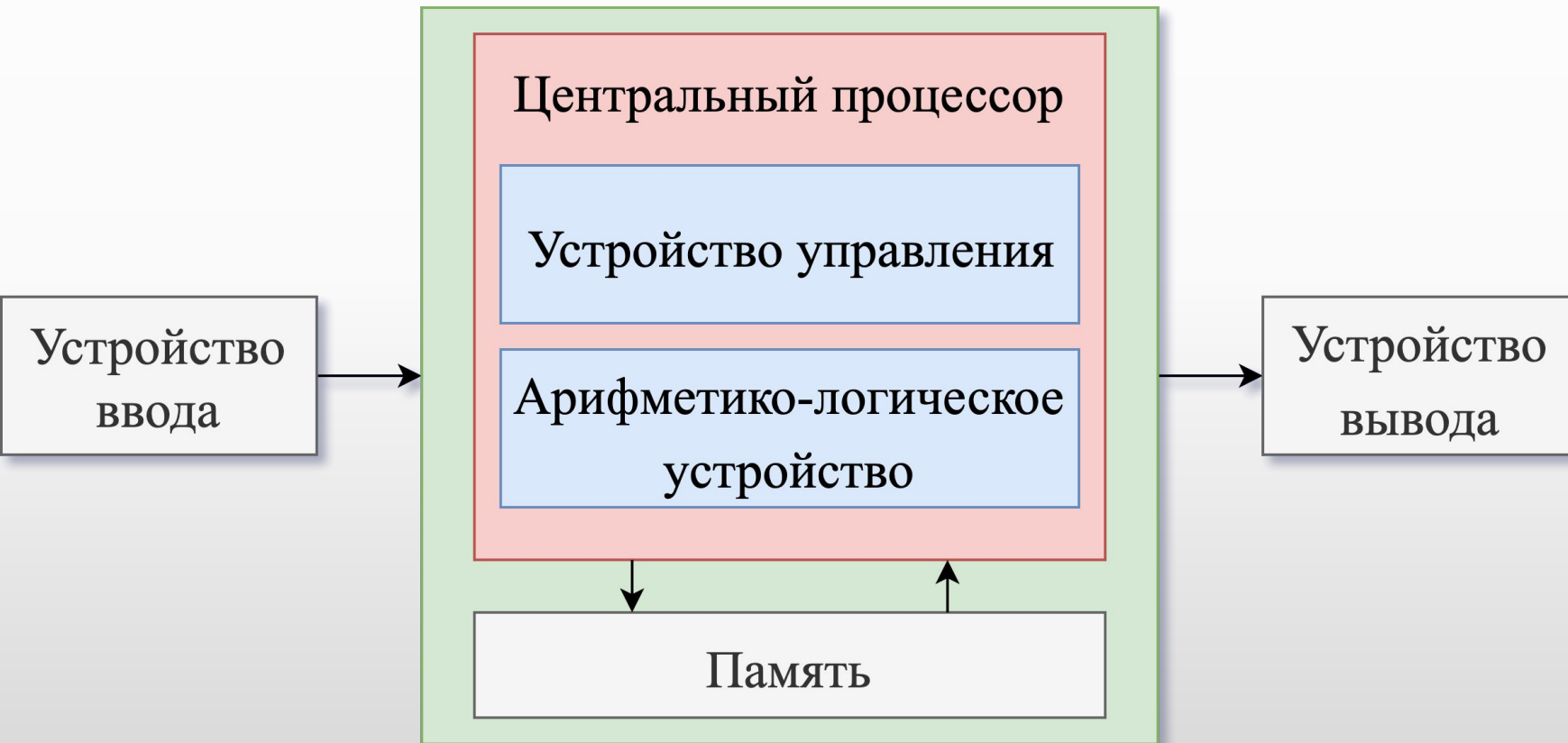
Процедурная декомпозиция — разделение большой программы на отдельные части. Процедуры облегчают разработку, отладку и сопровождение программы.

Процедурное программирование — выполнение программы сводится к последовательному выполнению инструкций с целью преобразования исходного состояния памяти, т.е. значений исходных данных, в заключительное, т.е. в результаты

Процедурное программирование — парадигма программирования, отражающая традиционную архитектуру Фон Неймана (Принстонскую архитектуру).

Архитектура фон Неймана

Архитектура фон Неймана – широко известный принцип совместного хранения команд и данных в памяти компьютера.



Схематичное изображение машины фон Неймана

Принципы архитектуры фон Неймана

Принцип двоичного кодирования. Вся информация, как данные, так и команды, кодируются 0 и 1;

Принцип программного управления. Вычисления, предусмотренные алгоритмом решения, должны быть представлены в виде программы, состоящей из последовательности управляющих слов — команд;

Принцип однородности памяти. Команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы;

Принцип адресности. Память состоит из пронумерованных ячеек, причем процессору в произвольный момент доступна любая ячейка (принцип открывает возможность использования переменных);

Принцип возможности условного перехода. Выполнение программы осуществляется последовательно, однако в программах возможно реализовать переход к любой части кода.

Оператор безусловного перехода goto (jump)

goto (от англ. go to – «перейти на») – оператор безусловного перехода (перехода к определенной точке программы, обозначенной номером строки либо меткой) в некоторых языках программирования.

Применение:

- ВЫХОД ИЗ ВЛОЖЕННЫХ ЦИКЛОВ;
- вместо средств обработки исключений.

Распространение:

оператор goto имеется в таких языках, как Фортран, Алгол, Кобол, Бейсик, С и С++, С#, D, Паскаль, Perl, Ада, PHP и многих других.

Он присутствует также ***во всех языках ассемблера*** (обычно под названием jmp, jump или bra (от англ. branch – ветвь)).

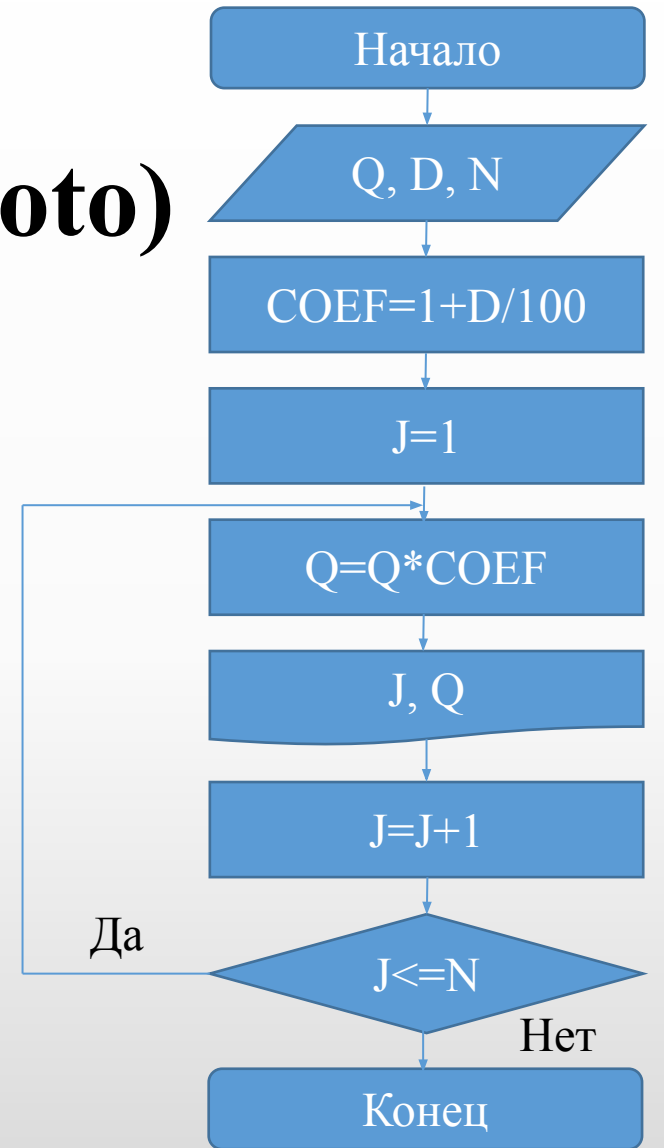
Пример выхода из цикла (на языке C++)

```
int matrix[n][m];
int value;
...
for(int i=0; i<n; ++i)
    for(int j=0; j<m; ++j)
        if (matrix[i][j] == value)
        {
            printf("value %d found in cell (%d,%d)\n",value,i,j);
            //act if found
            goto end_loop;
        }
printf("value %d not found\n",value);
//act if not found
end_loop: ;
```

Расчет сложных процентов (реализация в процедурном стиле с goto)

```
10 PRINT «Расчет сложных процентов»  
20 INPUT «Введите Q, D, N», Q, D, N  
30 COEF=1+D/100  
40 J=1  
50 Q=Q*COEF  
60 PRINT J, Q  
70 J=J+1  
80 IF J<=N THEN GOTO 50  
90 END
```

Расчет сложных процентов BASIC



Расчет сложных процентов блок-
схема

Языки поддерживающую процедурную парадигму

