Калинин Станислав
Web2019

Homework #5

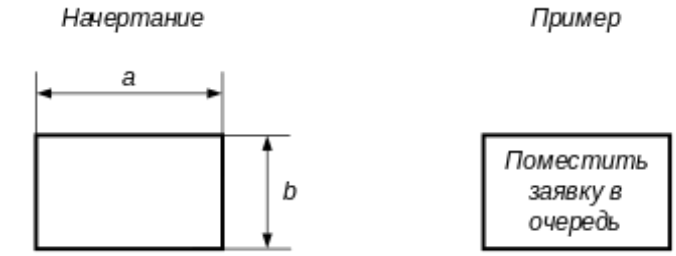
1
Нарисуйте и опишите основные свойства блок-схем алгоритмов

Основные элементы схем алгоритма

При начертании элементов рекомендуется придерживаться строгих размеров, определяемых двумя значениями a и b. Значение a выбирается из ряда 15, 20, 25.. мм. Определение размеров несет рекомендательный характер, однако, стоит отметить, что при соблюдении выполнения размеров блок-схемы имеют более аккуратный вид.

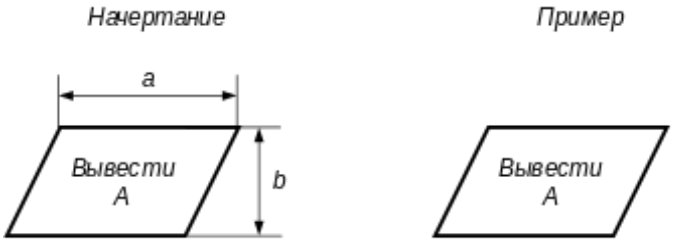
Действие

Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).



Данные (ввод/вывод)

Символ отображает данные, носитель данных не определен.

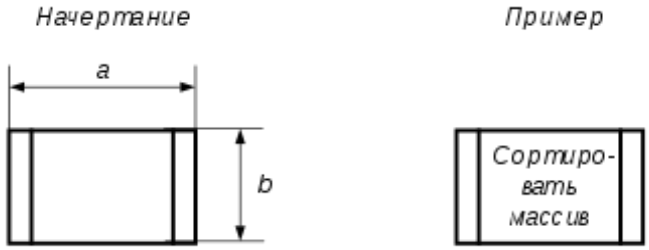


Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод). Данный символ не определяет носителя данных (для указания типа носителя данных используются специфические символы).

Калинин Станислав
Web2019

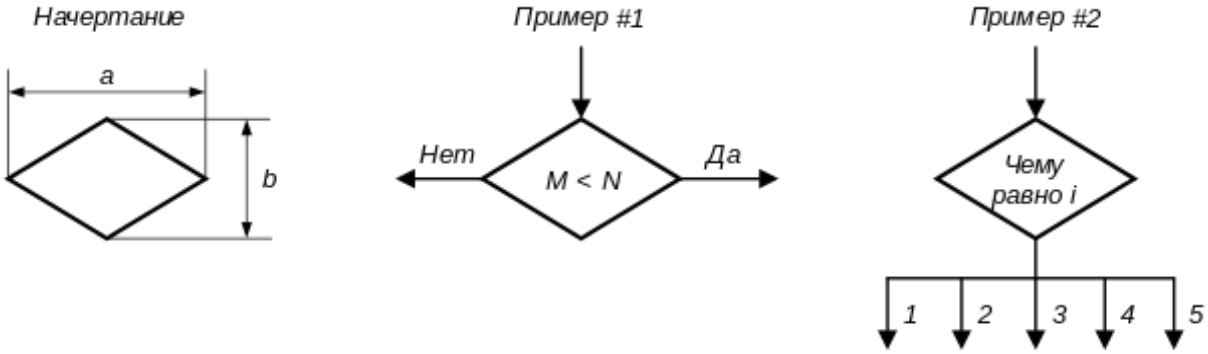
Предопределенный процесс (функция)

Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле). Например, в программировании − вызов процедуры или функции.



Вопрос (условие или решение)

Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.

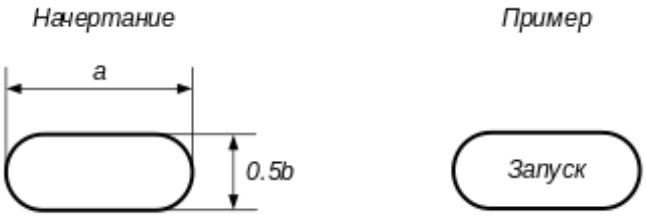


Отображает решение или функцию переключательного типа с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления условий, определенных внутри этого элемента. Вход в элемент обозначается линией, входящей обычно в верхнюю вершину элемента. Если выходов два или три, то обычно каждый выход обозначается линией, выходящей из оставшихся вершин (боковых и нижней). Если выходов больше трех, то их следует показывать одной линией, выходящей из вершины (чаще нижней) элемента, которая затем разветвляется. Соответствующие результаты вычислений могут записываться рядом с линиями, отображающими эти пути. Примеры решения: в общем случае − сравнение (три выхода: >, <, =); в программировании − условные операторы `if` (два выхода: `true`, `false`) и `case` (множество выходов).

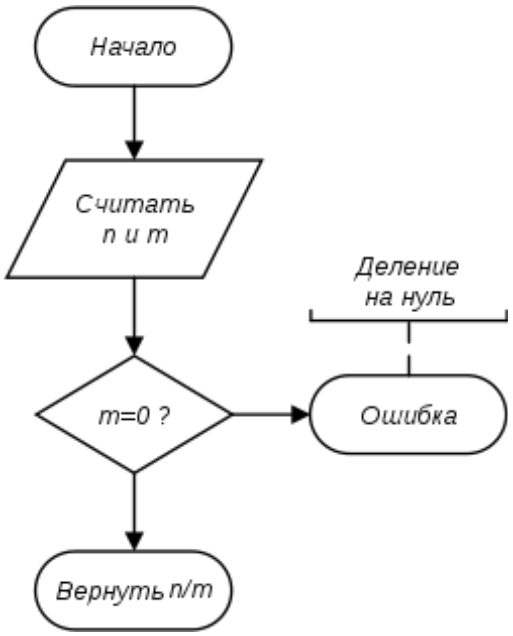
Ограничитель

Символ отображает вход из внешней среды и выход во внешнюю среду (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).

Калинин Станислав
Web2019

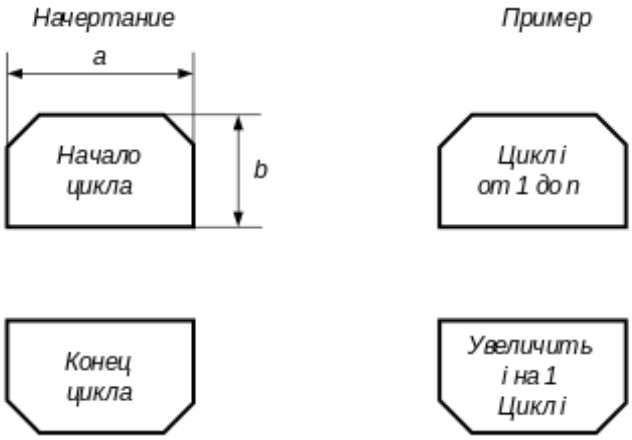


На практике имеют смысл следующие описания ограничителей: начало/конец, запуск/останов, перезапуск (подразумевает перезапуск данной блок-схемы), ошибка (подразумевает завершение алгоритма с ошибкой), исключение (подразумевает исполнение программного исключения)



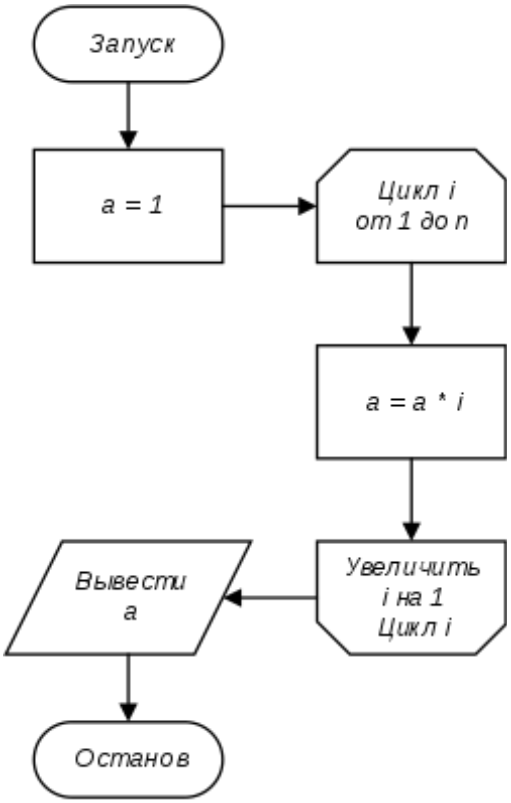
Цикл

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

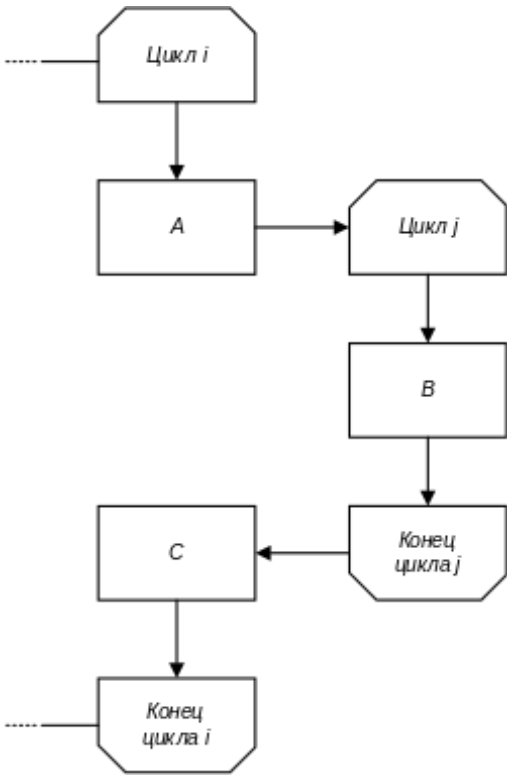


Калинин Станислав
Web2019

Пример блок-схемы расчета факториала с использованием цикла



Пример вложенных циклов

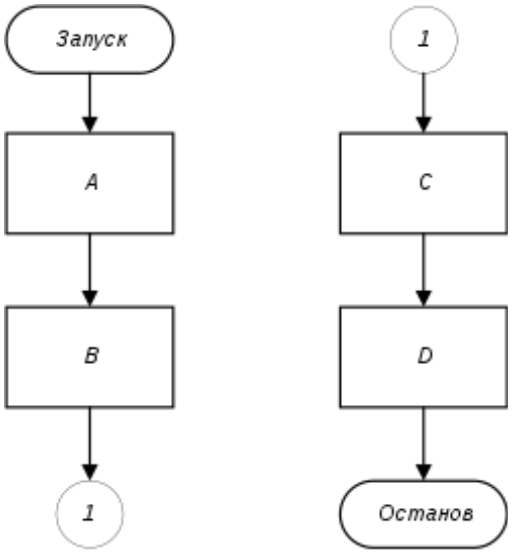


Калинин Станислав
Web2019

Соединитель

Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.

Разделение алгоритма на две части с использованием соединителей



Калинин Станислав
Web2019

2
В операторе switch за что отвечают конструкции: case, default, break?

Конструкция `switch` заменяет собой сразу несколько `if`.

Она представляет собой более наглядный способ сравнить выражение сразу с несколькими вариантами.

Синтаксис

Конструкция `switch` имеет один или более блок `case` и необязательный блок `default`.

Выглядит она так:

```
switch(x) {
  case 'value1':  // if (x === 'value1')
    ...
    [break]

  case 'value2':  // if (x === 'value2')
    ...
    [break]

  default:
    ...
    [break]
}
```

- Переменная `x` проверяется на строгое равенство первому значению `value1`, затем второму `value2` и так далее.
- Если соответствие установлено – `switch` начинает выполняться от соответствующей директивы `case` и далее, до ближайшего `break` (или до конца `switch`).
- Если ни один `case` не совпал – выполняется (если есть) вариант `default`.

Калинин Станислав
Web2019

3
Явное преобразование типов. Опишите конструкции явного преобразования в строку, число, булево значение

В JavaScript есть три преобразования:

1. Строковое: `String(value)` – в строковом контексте или при сложении со строкой. Работает очевидным образом.
2. Численное: `Number(value)` – в численном контексте, включая унарный плюс `+value`. Происходит при сравнении разных типов, кроме строгого равенства.
3. Логическое: `Boolean(value)` – в логическом контексте, можно также сделать двойным НЕ: `!!value`.

Точные таблицы преобразований даны выше в этой главе.

Особым случаем является проверка равенства с `null` и `undefined`. Они равны друг другу, но не равны чему бы то ни было ещё, этот случай прописан особо в спецификации.

[*Больше, читайте по ссылке*]

Калинин Станислав
Web2019

4
Что такое массивы? Зачем их использовать?

Массив – это особый тип объекта, предназначенный для работы с упорядоченным набором элементов.

- Объявление:

```
// квадратные скобки (обычно)
let arr = [item1, item2...];

// new Array (очень редко)
let arr = new Array(item1, item2...);
```

- Вызов `new Array(number)` создаёт массив с заданной длиной, но без элементов.
- Свойство `length` отражает длину массива или, если точнее, его последний цифровой индекс плюс один. Длина корректируется автоматически методами массива.
- Если мы уменьшаем `length` вручную, массив укорачивается.

Мы можем использовать массив как двустороннюю очередь, используя следующие операции:

- `push(...items)` добавляет `items` в конец массива.
- `pop()` удаляет элемент в конце массива и возвращает его.
- `shift()` удаляет элемент в начале массива и возвращает его.
- `unshift(...items)` добавляет `items` в начало массива.

Чтобы пройтись по элементам массива:

- `for (let i=0; i<arr.length; i++)` – работает быстрее всего, совместим со старыми браузерами.
- `for (let item of arr)` – современный синтаксис только для значений элементов (к индексам нет доступа).
- `for (let i in arr)` – никогда не используйте для массивов!

[*Узнать больше, перейдите по этой ссылке*]