На территории Российской Федерации действует *единая система программной документации (ЕСПД)*, частью которой является Государственный стандарт — ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем».

Схема алгоритма отображает последовательность операций в программе и представляет из себя совокупность символов, соответствующих этапам работы алгоритма, и соединяющих их линий.

Основные элементы схем алгоритмов

Схема алгоритма состоит из:

1) символов процесса, указывающих фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);

2) линейных символов, указывающих поток управления;

3) специальных символов, используемых для облегчения написания и чтения схемы.

Пунктирная линия используется для соединения символа с комментарием.

Сплошная линия отражает зависимости по управлению между символами и может снабжаться стрелкой.

Направление потока слева направо и сверху вниз считается стандартным. Стрелка при этом не указывается! Если в линии встречаются направления снизу вверх или справа налево, то в этом случае в конце линии добавляется стрелка.

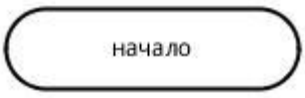
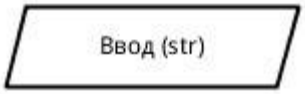
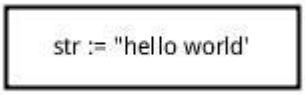
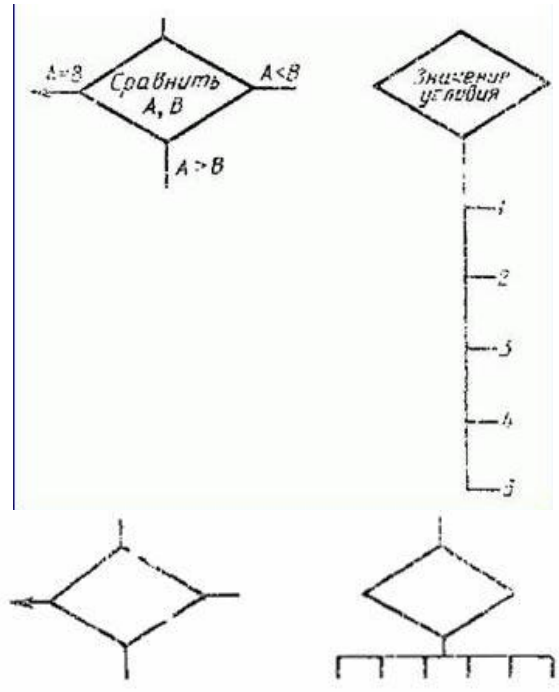
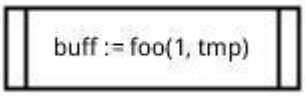
В схемах следует избегать пересечения линий.

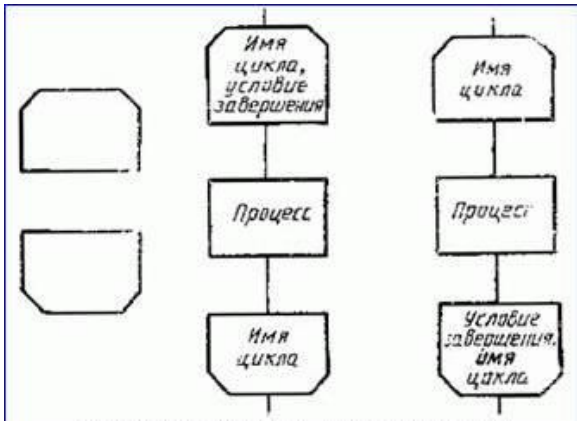
Если две или более линии объединяются в одну линию, место объединения должно быть смещено.

Согласно п. 4.2.4 ГОСТа, линии должны подходить к символу слева, либо сверху, а исходить снизу, либо справа.

Все блоки схемы алгоритма должны быть одинаковыми по ширине внутри одной схемы!

Основные символы

 Терминатор начала и конца работы алгоритма	Терминатором начинается и заканчивается любой алгоритм.
 Символ операций ввода и вывода данных	Данный символ используется, если источник данных не принципиален.
 Символ выполнения операций над данными	В блоке операций обычно размещают одну или несколько (ГОСТ не запрещает) операций присваивания, не требующих вызова внешних функций.
 Символ, иллюстрирующий ветвление алгоритма	Блок в виде ромба имеет один вход и несколько подписанных выходов. В случае, если блок имеет 2 выхода (соответствует оператору ветвления), на них подписываются условия перехода (например, «да/нет»). Если из блока выходит большее число линий (оператор выбора), внутри него записывается имя переменной, а на выходящих дугах — значения этой переменной.
 Символ вызова внешней процедуры	Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).

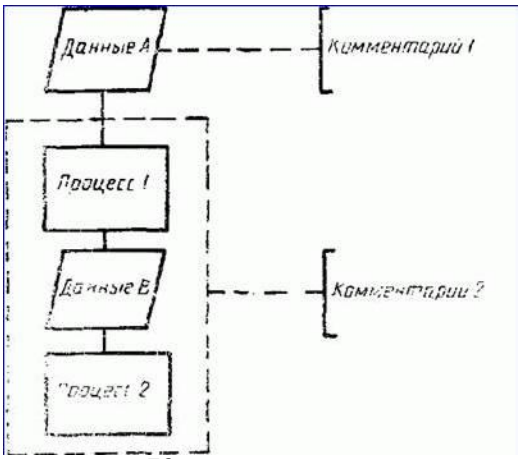


Символы начала и конца цикла

Символы начала и конца цикла содержат имя и условие. Условие может отсутствовать в одном из символов пары. Расположение условия, определяет тип оператора, соответствующего символам на языке высокого уровня — оператор с предусловием (while) или постусловием (do … while).

Соединитель

В случае, если схема алгоритма не умещается на лист, используется символ соединителя, отражающий переход потока управления между листами. Символ может использоваться и на одном листе, если по каким-либо причинам тянуть линию неудобно.



Комментарий

Комментарий может быть соединен как с одним блоком, так и группой.

Группа блоков выделяется на схеме пунктирной линией.

Примеры схем алгоритмов

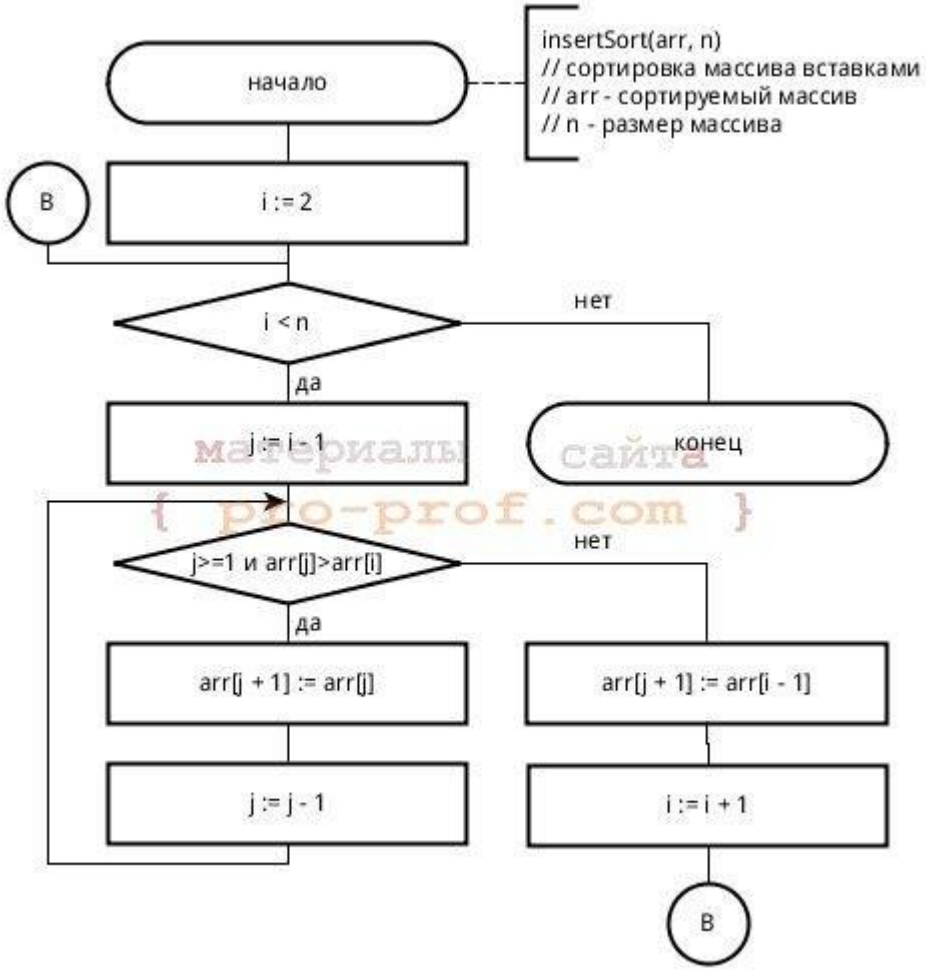


Схема алгоритма сортировки вставками

В приведенной схеме для организации цикла используется символ ветвления. Здесь также показано каким образом может использоваться символ перехода — его можно использовать не только для соединения частей схем, размещенных на разных листах, но и для сокращения количества линий. В ряде случаев это позволяет избежать пересечения линий и упрощает восприятие алгоритма.

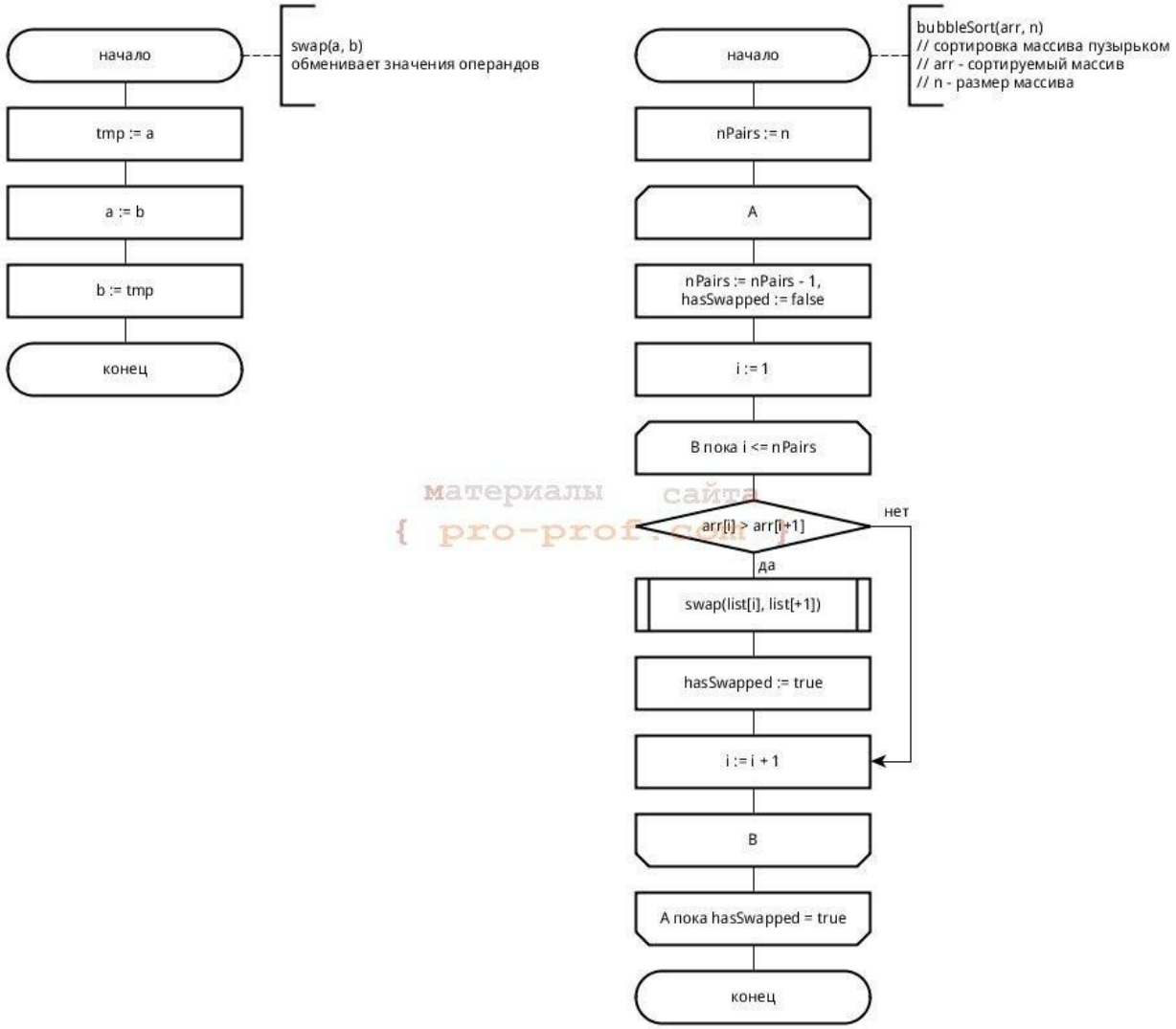


Схема алгоритма сортировки пузырьком

На схеме показано использование символов начала и конца цикла. Условие внешнего цикла (A) проверяется в конце (*с постусловием*), он работает до тех пор, пока переменная *hasSwapped* имеет значение *true*. Внутренний цикл использует *предусловие* для перебора пар сравниваемых элементов. В случае, если элементы расположены в неправильном порядке, выполняется их перестановка посредством вызова *внешней процедуры* (*swap*). Для того, чтобы было понятно назначение внешней процедуры и порядок следования ее аргументов, необходимо писать комментарии. В случае, если функция возвращает значение, комментарий может быть написан к символу терминатору конца.

Для создания схем алгоритмов удобно использовать утилиту *yEd*, которая является бесплатной и кроссплатформенной (https://www.yworks.com/downloads#yEd). С ее помощью можно также строить диаграммы UML.