##### ЭТАП Схемы алгоритмов

##### **Задание**

Разработать алгоритмы, связанные с решением задачи, и подготовить соответствующий раздел пояснительной записки к курсовой работе. Вы должны нарисовать не менее трех схем алгоритмов (например, алгоритм работы программного меню, содержательный алгоритм игры, вспомогательный алгоритм проверки проигрыша/выигрыша).

##### Оформление

Схемы алгоритмов должны быть выполнены в соответствии с требованиями ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». Основные положения стандарта приведены в [методических указаниях](_blank) по выполнению курсовой работы. Для рисования схем алгоритмов рекомендуется использовать редактор UMLet, который содержится в среде разработки MinIDE. Возможно использование web-интерфейса на сайте [draw.io](https://app.diagrams.net/)

Общий балл за этап складывается из следующих показателей:

– приведено не менее трех схем алгоритмов, связанных с решаемой задачей – 5 баллов, иначе 0 баллов;

– схемы выполнены в соответствии с требованиями – 10 балла, оценка снижается на 1 балл за каждую ошибку.

Максимальная оценка за этап – 15 баллов.

Описать предметную область и решаемую задачу, провести обзор существующих решений (выявить достоинства и недостатки).

Для описания алгоритмов можно использовать запись на естественном языке, с помощью графических обозначений, на псевдокоде и непосредственно на языке программирования.

На начальных этапах разработки алгоритма рекомендуется использовать графические обозначения. Это позволяет лучше увидеть связи между основными частями алгоритма. При последующей детализации рекомендуется сначала переписать алгоритм на псевдокод. Псевдокод — это код на выбранном для реализации языке программирования с включениями на естественном языке там, где сложно записать действие как один оператор на языке программирования. В дальнейшем эти включения заменяются на несколько операторов или вызовы подпрограмм с сохранением описаний на естественном языке в форме комментариев. Таким образом сразу будет получаться код с необходимым и достаточным количеством комментариев.

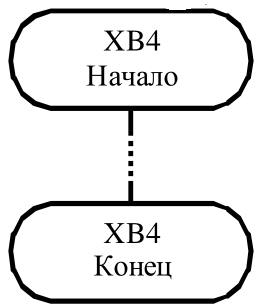
Для графического представления алгоритма нужно использовать стандарт ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения», который является переводом международного стандарта ISO 5807:1985. Используемые символы указаны в таблице 3.

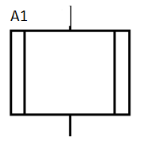
Необходимо нарисовать от 2 до 4 схем алгоритмов. Обязательной является схема основного алгоритма (основное меню программы). Так как разработка алгоритма происходит до написания кода, внутри символов рекомендуется использовать естественный язык и математические формулы, а не код на языке программирования C.

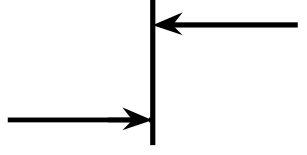
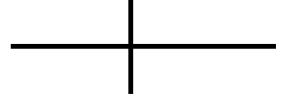
Таблица 3 — Символы, используемые в схемах программ

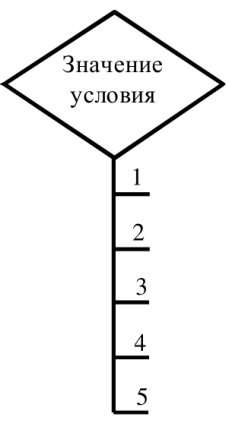
|  |  |  |
| --- | --- | --- |
| Символ | Название | Назначение |
|  | Процесс | Символ отображает функцию обработки данных любого вида |
|  | Данные | Символ отображает данные, носитель данных не определен |
|  | Предопре­делённый процесс | Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в подпрограмме (модуле) |
|  | Решение | Символ отображает 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 рядом с линиями, отображающими эти пути. |
|  | Границы цикла | 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. |
|  | Терминатор | Символ отображает вход из внешней среды и выход во внешнюю среду (начало или конец схемы программы) |
|  | 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 один и тот же уникальный идентификатор. |
|  | Коммен­тарий | Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. |
|  | Символ с полосой | Символ указывает, что в этом же комплекте документации есть более подробное представление данной части алгоритма. В терминаторах такого подробного представления нужно указать идентификатор символа с полосой (рисунок 1а). |

Не требуется точное соответствие между количеством символов в алгоритме и операторов в программе на языке С — один символ может представлять от 1 до 10 строк кода на языке С. Также структура алгоритма на этом этапе может отличаться от разделения операторов по функциям на этапе реализации программы, так как могут появиться новые обстоятельства, например, удаление дублирования кода или соблюдение рекомендаций по длине функции.

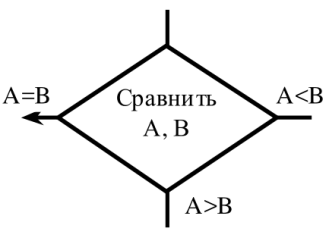
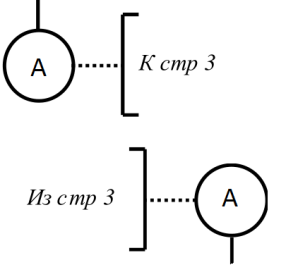








а) б) в) г)



д) е) ж)

Рисунок 1 — Использование идентификаторов и линий в схемах алгоритмов

При рисовании символов рекомендуется делать их фиксированной ширины и располагать равномерно. Если необходимо ссылаться на отдельные символы из других документов, то такие символы нужно снабжать идентификаторами, расположенными слева над символом (рисунок 1б).

Потоки управления в схемах показываются линиями. Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Такие направления входа и выхода потока управления считаются стандартными. Линии должны быть направлены к центру символа. Если поток управления имеет направление, отличное от стандартного, стрелки должны указывать это направление. Также стрелки используются в случаях, когда необходимо внести большую ясность в схему (например, при соединениях линий). Если две или более линии объединяются в одну линию, место объединения должно быть смещено (рисунок 1г).

В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются.

Для минимизации излишних пересечений или слишком длинных линий в схемах линии рекомендуется разрывать и на концах разрыва использовать соединители с одинаковым идентификатором.

Если схема состоит из нескольких страниц, то для соединителей на разных страницах рекомендуется указать ссылки к страницам с помощью символа комментария (рисунок 1д).

Несколько выходов из символа следует показывать:

1) несколькими линиями от данного символа к другим символам (рисунок 1е);

2) одной линией от данного символа, которая затем разветвляется в соответствующее число линий (рисунок 1ж).

Примеры схем алгоритмов показаны на рисунках 2 и 3.

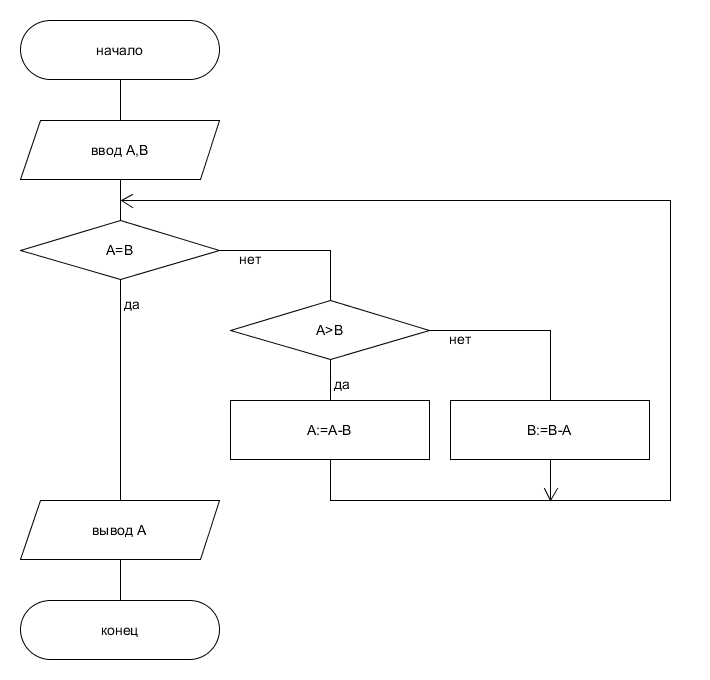


Рисунок 2 — Алгоритм Эвклида для нахождения НОД

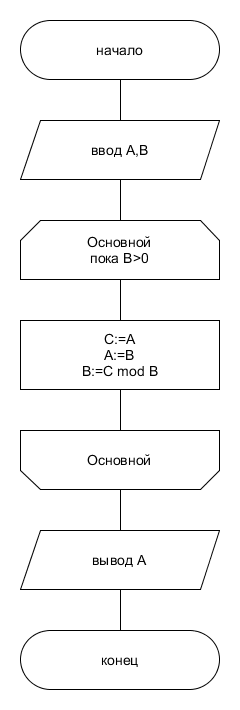
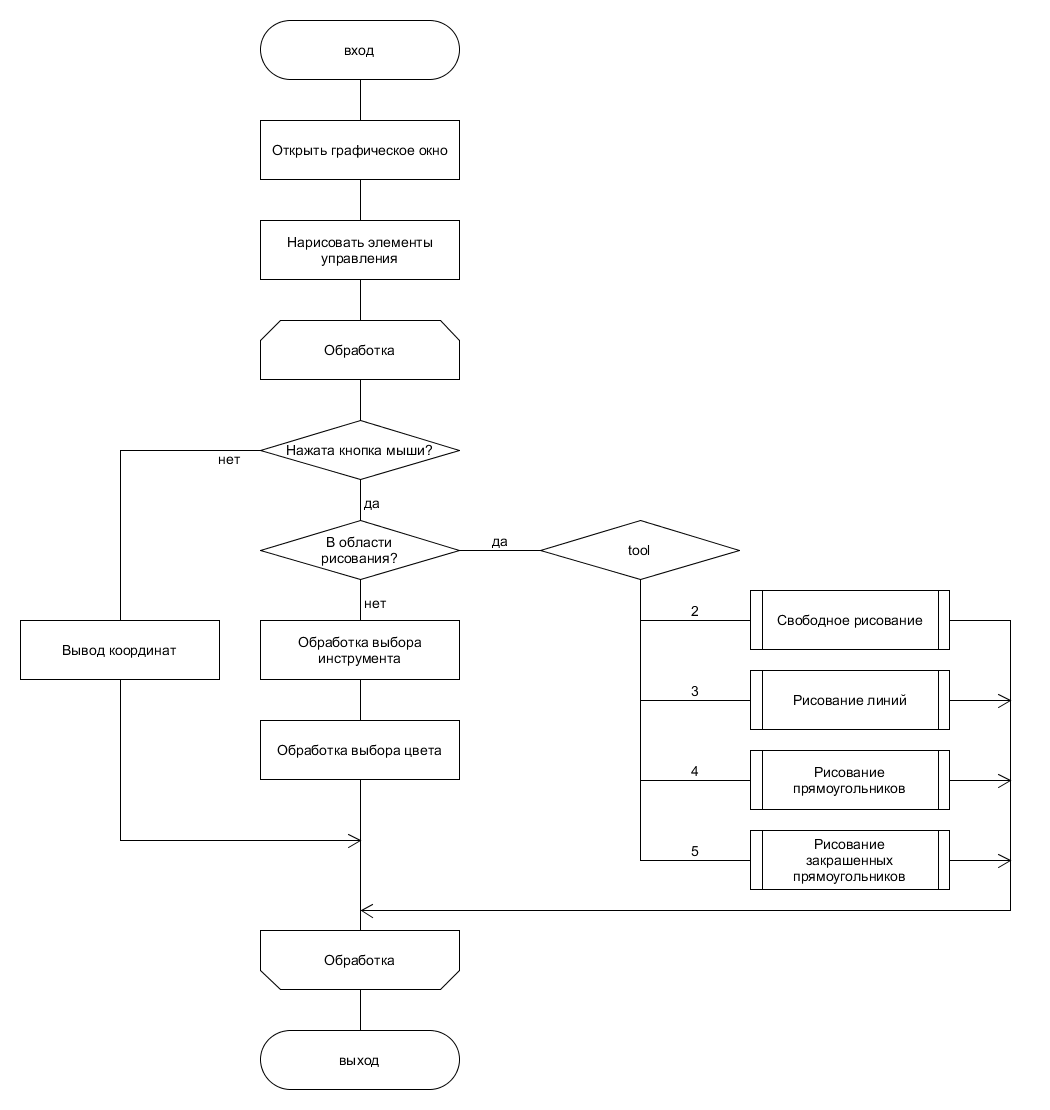


Рисунок 3 — Улучшенный алгоритм Эвклида и символы цикла

Для рисования схем используется специальная версия UMLet в среде MinIDE. Большинство online-инструментов использует либо предыдущий стандарт 1973 года, либо некоторую комбинацию с другими графическими нотациями (SDL, workflow, диаграмма деятельности UML). В большинстве инструментов отсутствует символ окончания цикла (хотя в помощи указывается, что он должен получаться путем переворачивания).

Выбор символов в UMLet выполняется из палитры «Flowchart GOST» двойным щелчком. Текст в символе изменяется в окне редактора. Аналогично из палитры можно выбирать подходящую линию. У линии можно указать характеристики тип стрелки и вид (сплошная или пунктирная) после «lt=», метки в начале и конце линии после «m1=», «m2=». Как установить другие характеристики, можно узнать с помощью Ctrl-пробел в окне редактора.

Схема основного алгоритма редактора показана на рисунке 3. Алгоритм предварительного рисования линий и прямоугольников при нажатой кнопке мыши показан на рисунке 4. Если алгоритм возвращает 1, то происходит рисование линии или прямоугольника в соответствии с текущим цветом и цветом заливки.

Рисунок 3 — Основной алгоритм программы

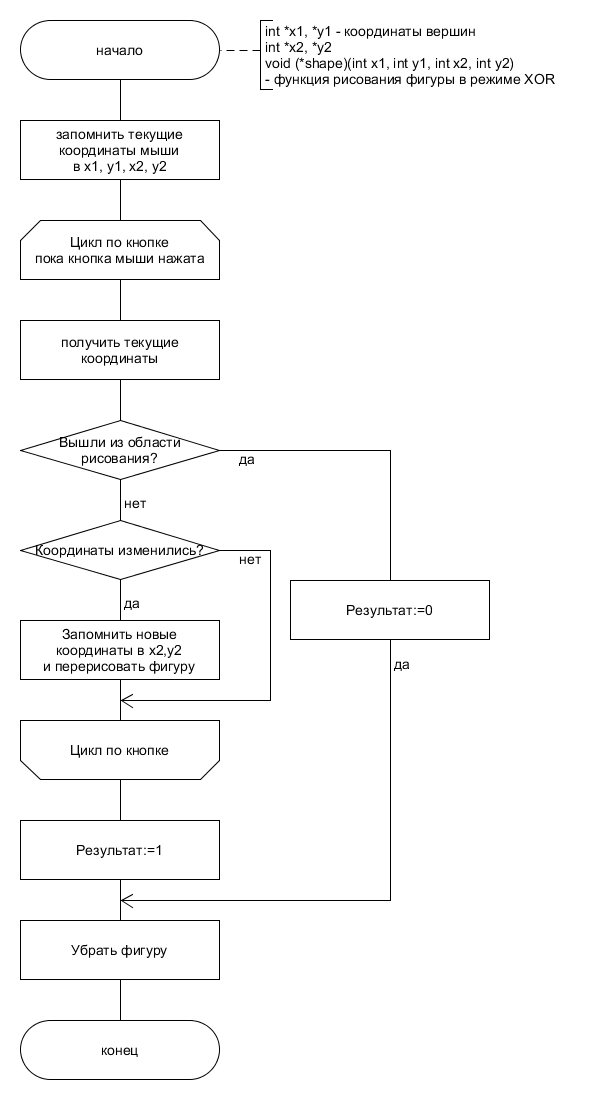


Рисунок 4 — Алгоритм предварительного рисования линий и прямоугольников