6 ФОРМЫ ПРЕДСТАВЛЕНИЯ ЛОГИЧЕСКОГО ПРОЕКТА МОДУЛЯ (ПРОГРАММЫ)

6.1 Введение

Модуль - это последовательность логически связанных фрагментов, оформленных как отдельная часть программы.

Спектр форм представления логического проекта модуля в соответствии с увеличением трудности его создания приведен ниже:

- текстовое описание;
- структурированный естественный язык (псевдокод);
- таблица решений;
- графическое представление;
- язык программирования.

6.1.1 Псевдокод

На стадии проектирования логики модуля - для повышения читабельности - удобно записывать алгоритмы на неформальном блочно-структурированном языке, который допускает описание производимых действий на естественном языке.

Поэтому псевдокод создается в виде стилизованного естественного языка, конструкции которого близки к конструкциям блочно-структурированного языка программирования. Таким образом, псевдокод позволяет формально изображать логику программы, не заботясь при этом о синтаксических тонкостях конкретного языка программирования.

Псевдокод не следует понимать как один из языков программирования.

При создании псевдокода можно отталкиваться от естественного языка или языка программирования.

6.1.2 Таблицы решений

Если действия, выполняемые программой, зависят от сложно взаимосвязанных условий, то описать логику такой программы проще с помощью таблицы решений (ТР).

Обычно таблица решений состоит из 4-х частей: перечня условий, комбинаций условий, возможных действий и предпринимаемых действий.

Перечень условий	Наборы условий
Возможные действия	Предпринимаемые действия

Если последовательность действий не важна, то в соответствующем столбце раздела *предпринимаемые действия* ставятся “+” или “x”.

Обычно условие представляется в форме вопросительного предложения и требует ответа “да-нет”. Пример гипотетической таблицы решений приведен ниже.

УСЛОВИЯ	1	2	3	4	5	6	7	8
C1	Д	Д	Д	Д	Н	Н	Н	Н
C2	Д	Д	Н	Н	Д	Д	Н	Н
C3	Д	Н	Д	Н	Д	Н	Д	Н
ДЕЙСТВИЯ								
D1	1	1	1	1	1	1	1	
D2	2	2	2	2	2	2	2	
D3								2
D4								1

6.2 Графическая форма представления логического проекта модуля

6.2.1 Схемы алгоритмов и программ

Использование графики для отображения алгоритма подтверждает утверждение, что образы выразительнее слов. Графические схемы остаются хорошей иллюстрацией, облегчающей процесс программирования. Кроме того, они являются эффективным посредником при общении программистов с непрограммистами.

Схемы полезны только тогда, когда они имеют достаточно обобщенный характер. Слишком большое число деталей затруднит составление и восприятие схем.

Как и псевдокод, графические схемы можно применять на любом уровне абстракции.

Графические символы и правила выполнения схем алгоритмов и программ стандартизированы в ГОСТ 19.701-90 (Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения).

Этот ГОСТ относится к системе ЕСПД. Стандарты ЕСПД начинаются с 19.xxx-год.

В этом стандарте определены символы, предназначенные для использования в документации по обработке данных, и приведено руководство по условным обозначениям для применения их в:

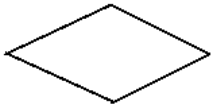
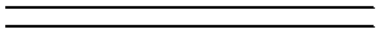
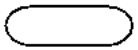
1)схемах данных;
2)схемах программ;
3)схемах работы системы;
4)схемах взаимодействия программ;
5)схемах ресурсов системы.

Стандарт не распространяется на форму записей и обозначений, помещаемых внутри символа или рядом с ними и служащих для уточнения выполняемых ими функций.

Схемы программ отображают последовательность операций в программе.

Символы

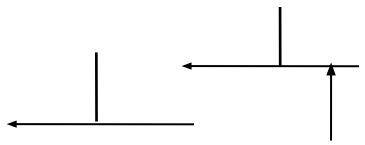
В таблице представлены основные символы, используемые в схемах алгоритмов и программ.

Наименование	**Обозначение**	**Функция**
Процесс		Отображает обработку данных любого вида (выполнение операции или группы операций, в результате которых изменяется значение, форма представления или расположение данных).
Решение		Выбирает направления выполнения алгоритма или программы в зависимости от условий, определенных внутри этого символа.
Параллельные действия		Отображает синхронизацию двух или более параллельных операций.
Предопределенный процесс		Отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).
Терминатор		Начало, конец, прерывание процесса обработки данных или выполнения программы.
Линия потока		Отображает поток данных или управления.
Комментарий		Используется для добавления комментариев в целях объяснения или примечаний.
Соединитель		Используется для обрыва линии потока и продолжения ее в другом месте.

Правила применения символов и выполнения схем

Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация.

Линии – это прямые, соединяющие символы на схеме и указывающие потоки управления. Направления линии потока сверху вниз и слева направо приняты за основные и, если линии потока не имеют изломов, стрелками можно не обозначать. В остальных случаях направление линии потока обозначать стрелкой обязательно.



Линия потока подводится, как правило, к центру символа. Следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой.

Две и более входящие линии могут объединяться в одну исходящую линию. Если две или более линии объединяются в одну линию, место объединения должно быть смещено.

Надписи на схемах

Внутри символов, как правило, даются записи о выполняемых операциях. Записи могут содержать сведения о нескольких операциях. При этом записи внутри символа должны быть представлены так, чтобы их можно было читать слева направо и сверху вниз, независимо от направления потока.

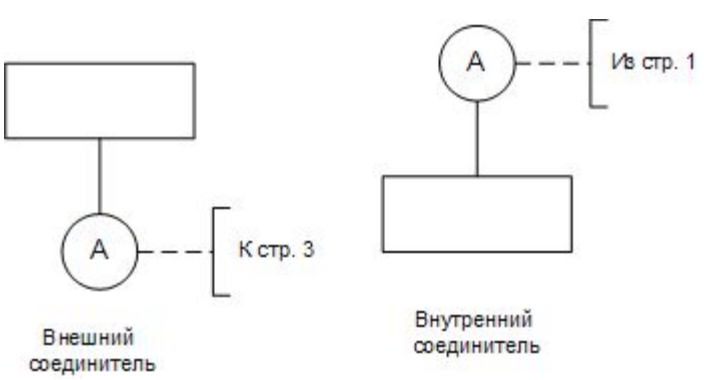
Комментарий применяется, если пояснение не помещается внутри символа (для пояснения характера параметров, особенностей процесса, линий потока, идентификаторов, формализовано записанных операций и др.). Комментарий помещают в свободном месте схемы на том же листе и соединяют с поясняемым символом.

В схемах может использоваться идентификатор символов, который определяет символ для использования в справочных целях в других элементах документации (например, в листинге программы) и при описании схемы в тексте или при устном выступлении. Идентификатор символа должен располагаться слева над символом.

Применение соединителей

При выполнении схем алгоритмов и программ возможны обрывы линий потока. Обрывы имеют место в случае, если:

- участок схемы, где должна быть проведена линия, соединяющая символы, насыщен надписями или другими изображениями;
- связываемые линией символы расположены на значительном расстоянии друг от друга;
- связываемые линией символы находятся на разных листах.

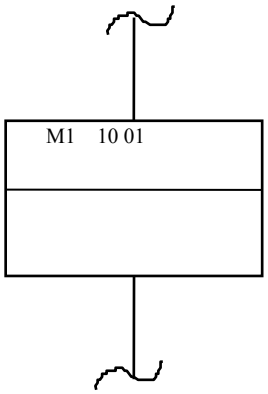


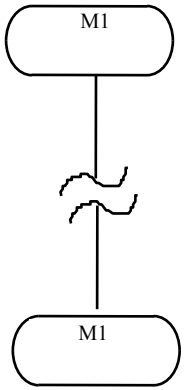
Соединитель в начале разрыва называется внешним соединителем, а соединитель в конце разрыва – внутреннем соединителем. Внутри этого символа указывается один и тот же идентификатор (буква или цифра). Ссылки к страницам могут быть приведены совместно с символом комментария для их соединителей.

Детализация схем алгоритмов и программ

Для относительно сложной программы составляют укрупненную схему, а ее детализация отображается в виде самостоятельной схемы, которая начинается и заканчивается символом указателя конца.

Детализируемая часть программы на укрупненной схеме изображается одним символом, внутри которого, в его верхней части проводят горизонтальную линию.





Над горизонтальной линией помещается идентификатор детализации программы и адресная ссылка, где размещен символ терминатор, обозначающий начало детализации.

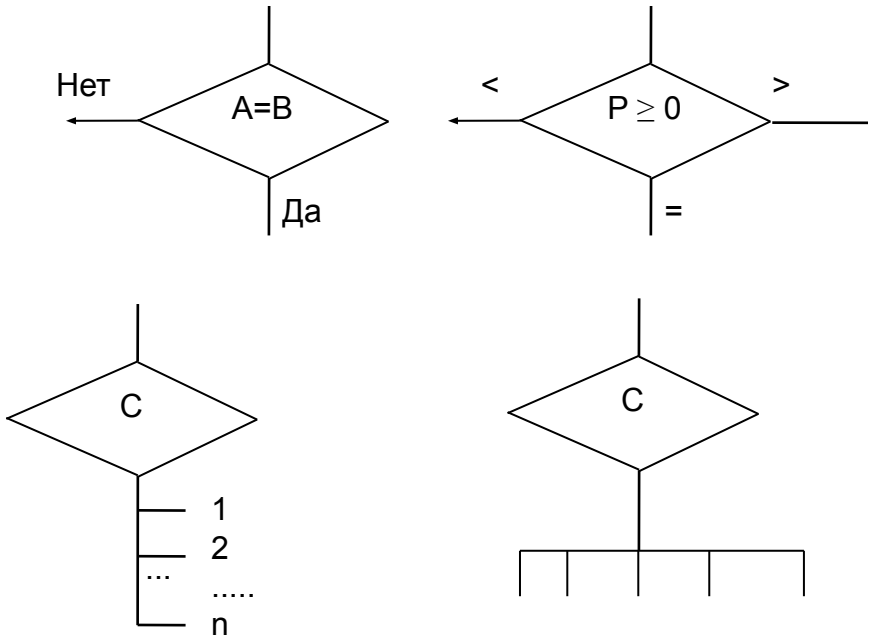
Внутри этого символа указывается идентификатор данной детализации.

Лист 10

Возможные варианты отображения решения

При числе вариантов не более трех признак условия решения (например: Да, Нет, =, >, <) проставляют над каждой выходящей линией потока или справа от линии потока. При числе вариантов более 3-х условие варианта проставляется в у соответствующей линии.

ординату зоны размещения таблицы или символа “Комментарий”, в которых приводят адреса всех вариантов.



ПРИМЕР: Схема подпрограммы Transmit

Передача массива во внутреннем ОЗУ через последовательный порт в режиме UART.

Адрес массива передается чрез регистр R0;

Размер массива передается через регистр R7.

Числовые значения слева над символами – это идентификаторы символов, которые служат для указания символов при описании схемы подпрограммы Transmit.