

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

ПЛАН ЛЕКЦИИ

- Этапы разработки программ
- Понятие и свойства алгоритма
- Формы представления алгоритма
- Трансляторы
- Типы данных
- Алгоритмические структуры
- Процедуры и функции
- Структурное программирование
- Объектно-ориентированное программирование

ЭТАПЫ РАЗРАБОТКИ ПРОГРАММ

- 1. Постановка задачи**
- 2. Анализ задачи и моделирование**
- 3. Выбор языка программирования**
- 4. Разработка алгоритма**
- 5. Проектирования общей структуры программы**
- 6. Кодирование или составление программы**
- 7. Отладка и тестирование программ**
- 8. Анализ полученных результатов**
- 9. Передача программы в эксплуатацию**
- 10. Сопровождение программы**

ПРИМЕР

■ Задание N 21.

Укажите правильный порядок следования этапов компьютерного моделирования:

- а) планирование и проведение компьютерных экспериментов,
- б) создание алгоритма и написание программы,
- в) разработка концептуальной модели, выявление основных элементов системы и их взаимосвязей,
- г) формализация, переход к модели,
- д) постановка задачи, определение объекта моделирования,
- е) анализ и интерпретация результатов.

■ Варианты ответа:

- в); д); б); г); а); е)
- д); в); г); б); а); е)
- а); б); г); д); в); е)
- д); г); б); в); а); е)

ПОНЯТИЕ АЛГОРИТМА

Алгоритм – это точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату.

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время.

СВОЙСТВА АЛГОРИТМОВ:

- ▣ **Дискретность** (прерывность, раздельность) — Выполнение алгоритма разбивается на последовательность элементарных действий – шагов. Каждое действие должно быть закончено исполнителем прежде, чем он перейдет к выполнению следующего действия.
- ▣ **Детерминированность** – это определенность (т.е. общепонятность и точность). Запись алгоритма должна быть такой, чтобы, выполнив очередную команду, исполнитель точно знал, какую команду надо выполнять следующей.
- ▣ **Результативность (конечность)** – алгоритм должен приводить к решению задачи за конечное число шагов.
- ▣ **Корректность** – алгоритм должен задавать правильное решение задачи.
- ▣ **Массовость** – с помощью одного и того же алгоритма можно решать однотипные задачи и делать это неоднократно. Свойство массовости значительно увеличивает практическую ценность алгоритмов.

ФОРМЫ ПРЕДСТАВЛЕНИЯ АЛГОРИТМА:

- на естественном языке (словесная запись алгоритма);
- словесно-математическая;
- в виде блок-схем (графическая форма);
- на языке псевдокодов (псевдокод – система обозначений и правил, предназначенная для единообразной записи алгоритмов), возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций. ;
- на языке программирования.




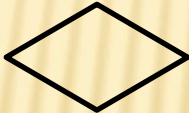


ПОНЯТИЕ БЛОК-СХЕМЫ

*Блок-схема - графическое представление алгоритма, когда отдельные действия изображаются в виде геометрических фигур – **блоков**.*

Внутри блоков указывается информация о действиях, подлежащих выполнению.

Связь между блоками изображают с помощью ***линий связи*** (*линий потока*), обозначающих передачу управления.

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ:

Название блока	Обозначение блока	Назначение блока
Терминатор		Начало/Конец
Процесс		Вычислительный процесс (Обработка данных)
Данные		Ввод/Вывод данных
Решение		Ветвление, выбор, проверка условия
Подготовка		Заголовок счетного цикла
Предопределенный процесс		Обращение к процедуре

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Компьютерная **программа** — последовательность инструкций, предназначенная для исполнения устройством управления вычислительной машины.

Программой называют алгоритм решения задачи, рассчитанный на исполнение его компьютером и записанный с помощью предложений используемого языка программирования.

Все языки программирования делятся **машинно-зависимые** и **машинно-независимые**.

Машинно-зависимые языки зависят от типа компьютера. Каждый компьютер имеет свой собственный язык программирования – машинный язык – и может исполнять программы, записанные только на этом языке.

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Если язык программирования ориентирован на конкретный тип процессора, то он называется *языком программирования низкого уровня* (например, язык ассемблер).

Машинно-независимые языки не зависят от типа компьютера, на котором они используются. Машинно-независимые языки в отличие от машинно-зависимых называются **языками высокого уровня**.

Алгоритмический язык – это определенный набор символов и специальных слов, которые в соответствии со строгими правилами записи команд (синтаксиса языка) описывают алгоритм решения задачи (например, *Algol, Cobol, Fortran, Basic, Pascal, C++*).

ТРАНСЛЯТОРЫ

Процесс равносильного преобразования алгоритма, заданного на языке программирования, в алгоритм на машинном языке называется *трансляцией*

Трансляторы – это специальные программы, которые переводят программы, написанные на языке высокого уровня, на язык машинных кодов.

Транслятор выполняет следующие основные функции:

- ✓ производит *синтаксический анализ* исходного текста программы;
- ✓ производит непосредственную *трансляцию* исходной программы на язык машинных кодов;
- ✓ распределяет память для полученной программы.

ТРАНСЛЯТОРЫ

Различают компиляторы и интерпретаторы

Компилятор преобразует исходную программу на любом языке высокого уровня в некоторую стандартную форму на машинном языке, называемую **объектным модулем**. Объектный модуль формируется лишь при отсутствии синтаксических ошибок.

Интерпретатор преобразует отдельные предложения исходного языка в машинный код и немедленно их исполняет. Интерпретатор не создает объектный модуль.

Компоновщик – программа, которая собирает результирующую программу из объектных модулей и стандартных библиотечных модулей и формирует исполняемый файл. *Исполняемый файл* – это файл, который может быть обработан или выполнен компьютером без предварительной трансляции.

ТЕСТ

■ Задание N 29.

Утверждение «Языковой процессор, который построчно анализирует исходную программу и одновременно выполняет предписанные действия, а не формирует на машинном языке скомпилированную программу, которая выполняется впоследствии» справедливо для ...

■ Варианты ответа:

- компилятора
- транслятора
- синтаксического анализатора
- интерпретатора

ДААННЫЕ

Данные – общее название всего того, чем оперирует компьютер. Любые цепочки символов, над которыми можно выполнять операции, являются данными.

Данное характеризуется **тремя** атрибутами:

- **именем;**
- **типом;**
- **значением.**

Выбор типа данных влияет на:

- *длину внутреннего представления, т.е. объем используемой памяти;*
- *точность вычислений и результата;*
- *возможность выполнения определенных действий над этими данными;*
- *скорость выполнения программы.*

ДАННЫЕ

Константами называются данные, которые нельзя переопределить, т. е. изменить в процессе исполнения алгоритма. Они имеют фиксированные тип и значение.

Данные, значения которых порождаются в процессе исполнения алгоритма, называются **переменными**.

Над данными можно выполнять ту или иную последовательность операций, описываемую **выражением**.

Данные, входящие в выражение, называются **операндами**.

Различают следующие типы выражений: арифметические, логические (в частности, сравнения, или отношения), символьные и т. д.

Результатом вычисления выражения является **значение**. Его тип зависит от типа операндов и от вида операций, выполняемых над операндами.

КОМАНДА ПРИСВАИВАНИЯ

Команда

$X:=Y$

называется командой ***присваивания***.

Здесь

X — имя (простой) переменной, элемента массива (переменной с индексами)

Y — *выражение*.

Примеры:

$n:=0$

$s:=s+1$

$y:=a^2+5$

$z:=x \bmod 2$

ЛИНЕЙНЫЕ АЛГОРИТМЫ

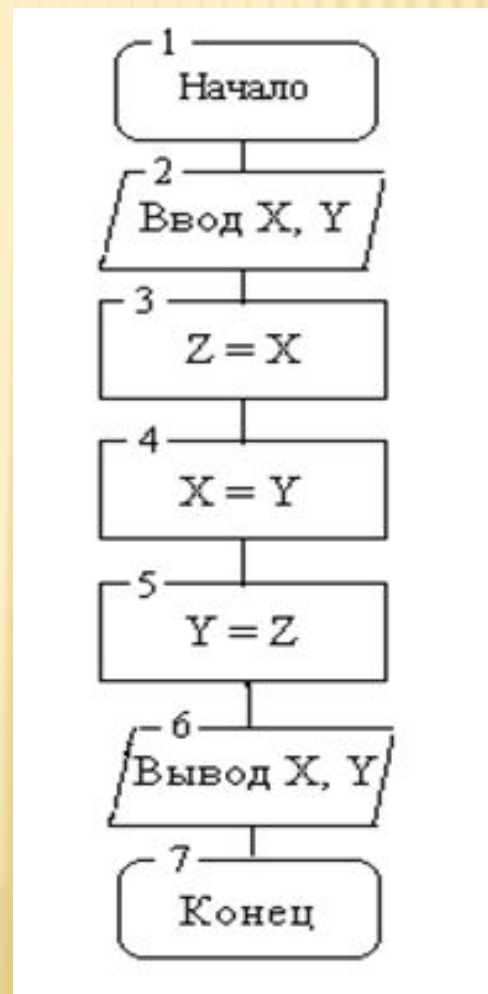
Алгоритмической структурой называется стандартный способ соединения отдельных шагов алгоритма для выполнения типичной последовательности действий.

В теории алгоритмов доказывається, что любой алгоритм может быть представлен в виде комбинации трех алгоритмических структур: следования, развилки и цикла.

Линейный алгоритм (следование) состоит из упорядоченной последовательности действий, не зависящей от значений исходных данных, при этом каждая команда выполняется только один раз строго после той команды, которая ей предшествует.

ПРИМЕР

Составить алгоритм, в результате выполнения которого переменные X и Y обменяются своими значениями.



ТЕСТ

■ Задание N 27.

Запись выражения $y = Ax^2 + Bx + C$ на алгоритмическом языке (возведение в степень обозначим через \wedge) имеет вид...

■ Варианты ответа:

- $y := A*x^2 + B*x + C$
- $y := (A*x)^2 + B*x + C$
- $y := Ax^2 + Bx + C$
- $y := Ax^2 + Bx + C$

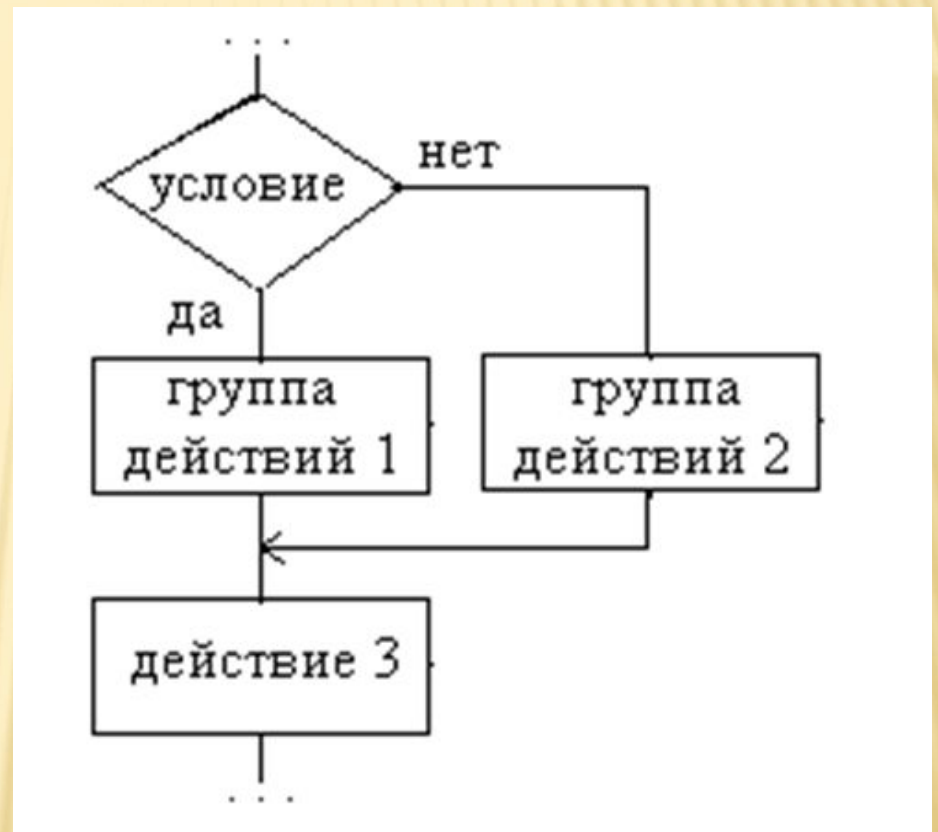
РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

Разветвляющимися называются алгоритмы, в которых в зависимости от значения какого-то выражения или от выполнения некоторого логического условия дальнейшие действия могут производиться по одному из нескольких направлений.

«ОБХОД»



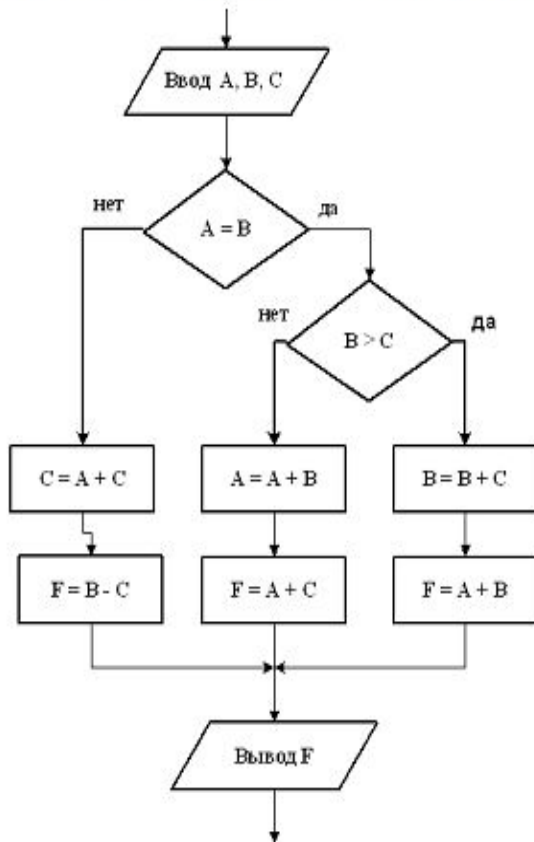
«РАЗВЕТВЛЕНИЕ»



ТЕСТ

Задание N 28.

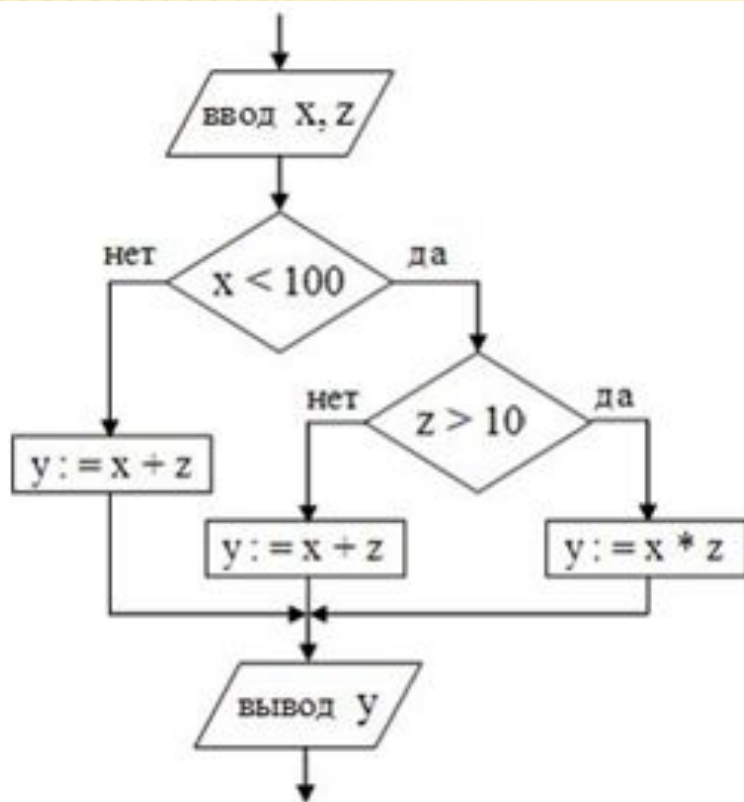
Дана блок-схема алгоритма. Если начальные значения переменных A , B и C равны 3, 3 и 1 соответственно, то значение переменной F будет равно ...



Варианты ответа:

- 7
- 8
- 6
- 1

ПРИМЕР

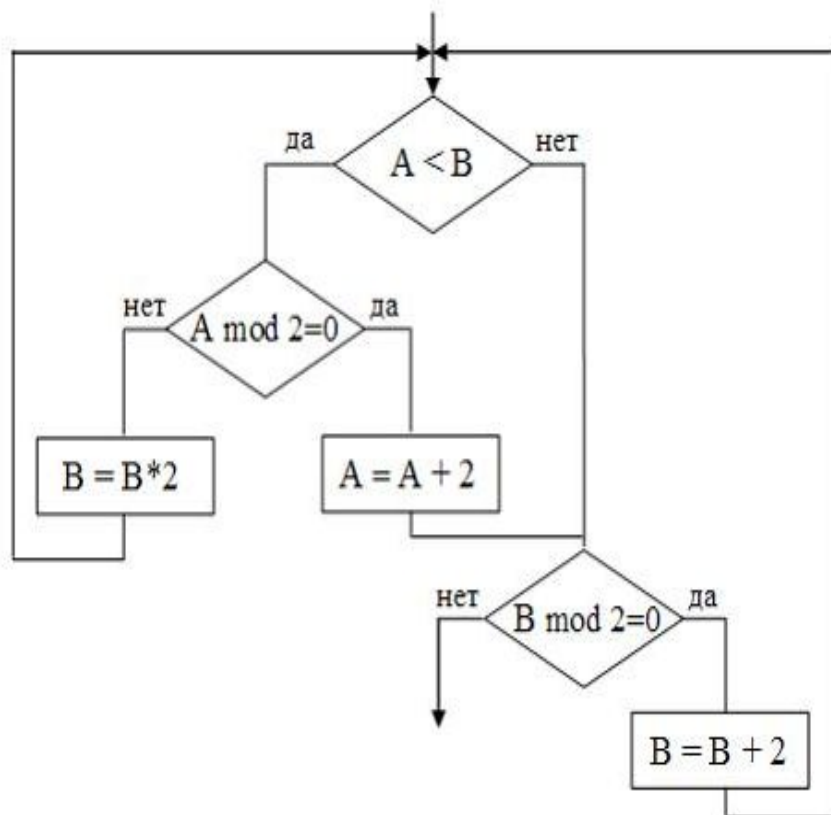


○ ВВОД x,z
если (x<100)
то если (z>10)
то y:=x*z
иначе y:=x+z
все
иначе y:=x+z
все
ВЫВОД y

ТЕСТ

Задание N 29.

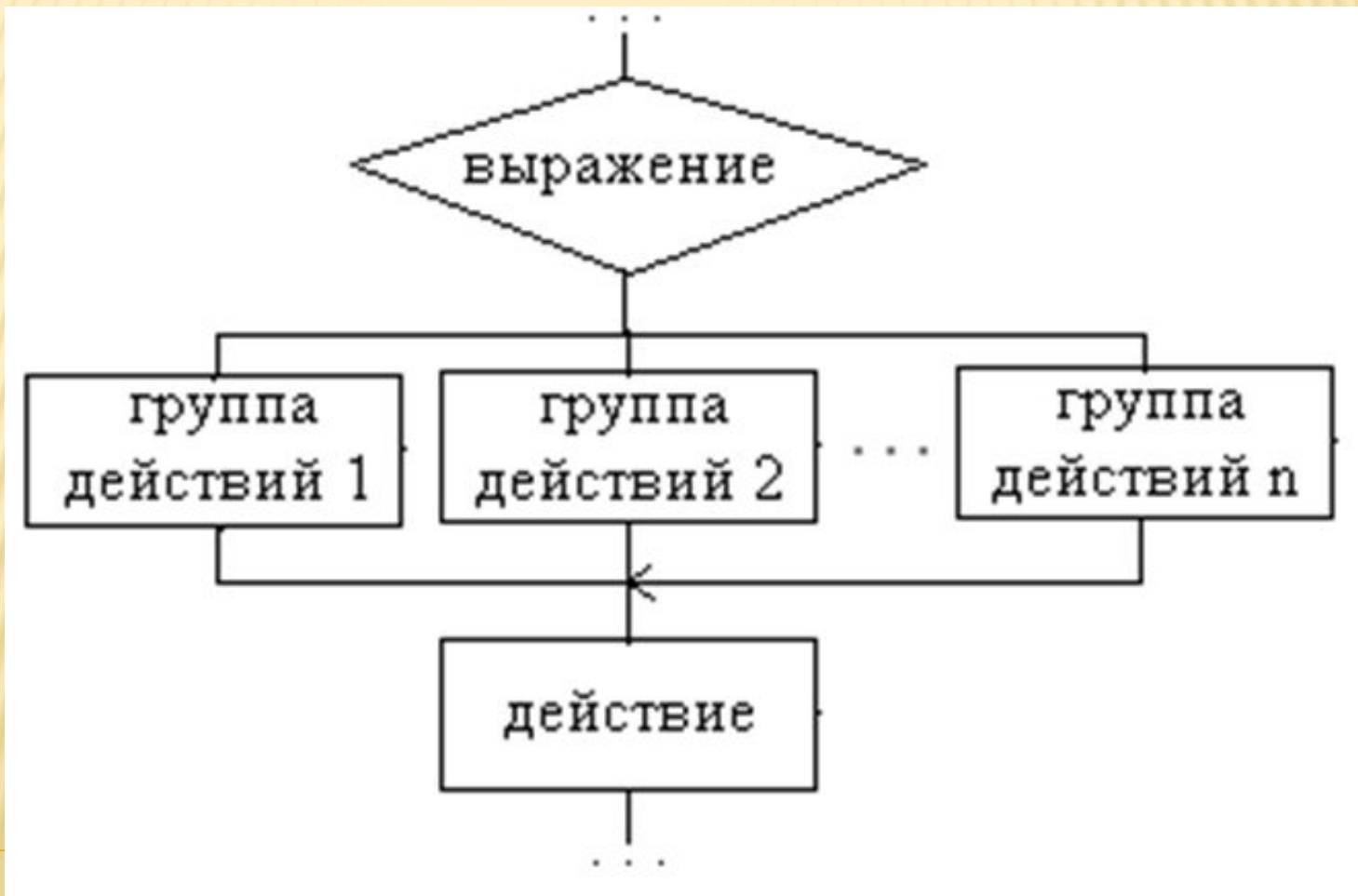
Определите, при каких начальных значениях переменных A и B алгоритм, представленный следующей блок-схемой, закончит работу (mod – функция, вычисляющая остаток от деления нацело первого аргумента на второй).



Варианты ответа:

- $A=1, B=6$
- $A=4, B=2$
- $A=5, B=3$
- $A=3, B=5$

МНОЖЕСТВЕННЫЙ ВЫБОР



ТЕСТ

Задание N 32.

Значение переменной m после выполнения фрагмента алгоритма (операция $\text{mod}(x, y)$ – получение остатка целочисленного деления x на y)

$k := 70$

выбор

при $\text{mod}(k, 12) = 5:$ $m := k;$

при $\text{mod}(k, 12) < 5:$ $m := 2;$

при $\text{mod}(k, 12) = 13:$ $m := 3;$

иначе $m := 1;$

все

будет равно ...

Варианты ответа:

3

2

1

70

ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

Цикл — разновидность управляющей конструкции, предназначенная для организации многократного исполнения набора инструкций.

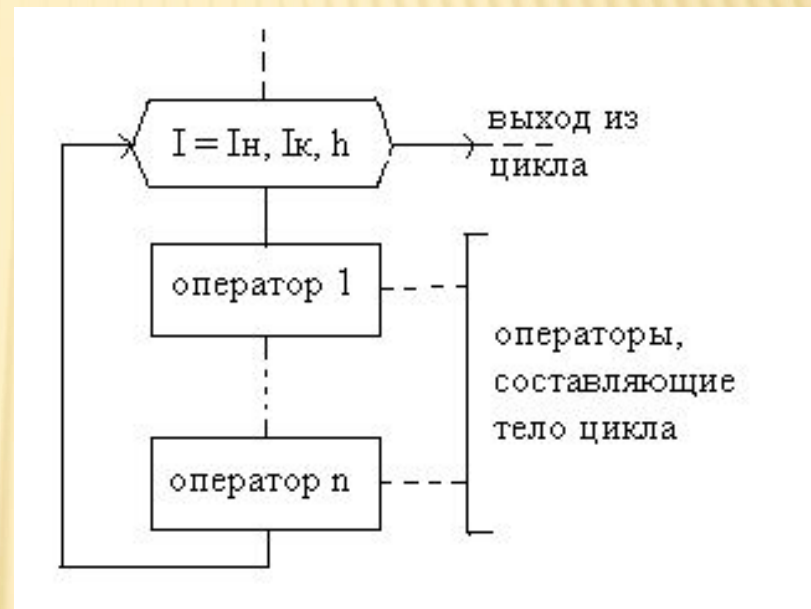
Различают три типа циклов:

- циклы со счетчиком (или с известным числом повторений);
- циклы с предусловием;
- циклы с постусловием.

В любом цикле должна быть **переменная**, которая управляет выходом из цикла, т.е. определяет число повторений цикла.

Последовательность действий, которая должна выполняться на каждом *шаге цикла* (т.е. при каждом повторении цикла), называется **телом цикла** или **рабочей частью цикла**.

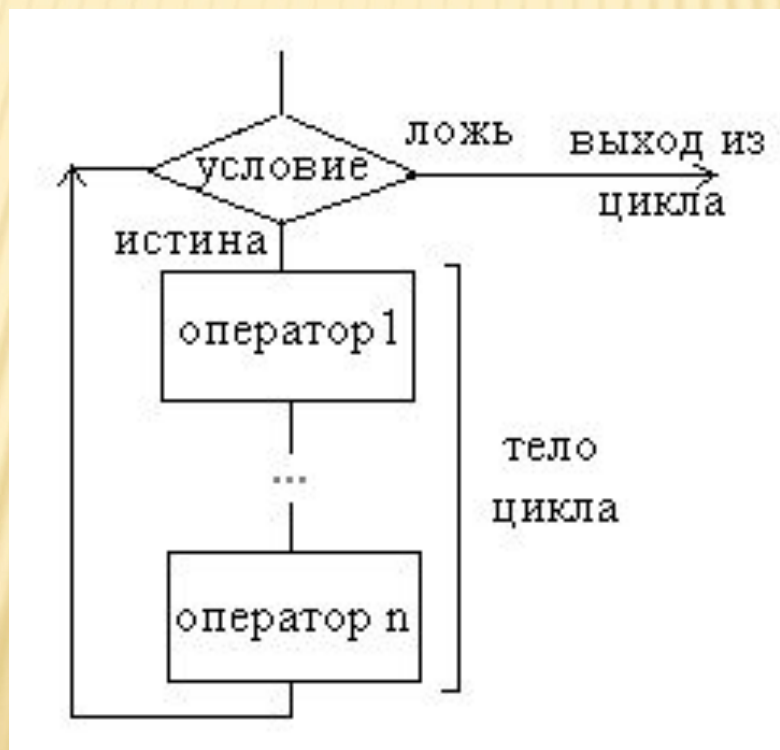
ЦИКЛЫ СО СЧЕТЧИКОМ



В циклах со счетчиком число повторений цикла является фиксированным числом. Переменная, которая считает количество повторений (шагов) цикла, называется **счетчиком цикла** (параметром цикла, управляющей переменной цикла).

ЦИКЛЫ С ПРЕДУСЛОВИЕМ

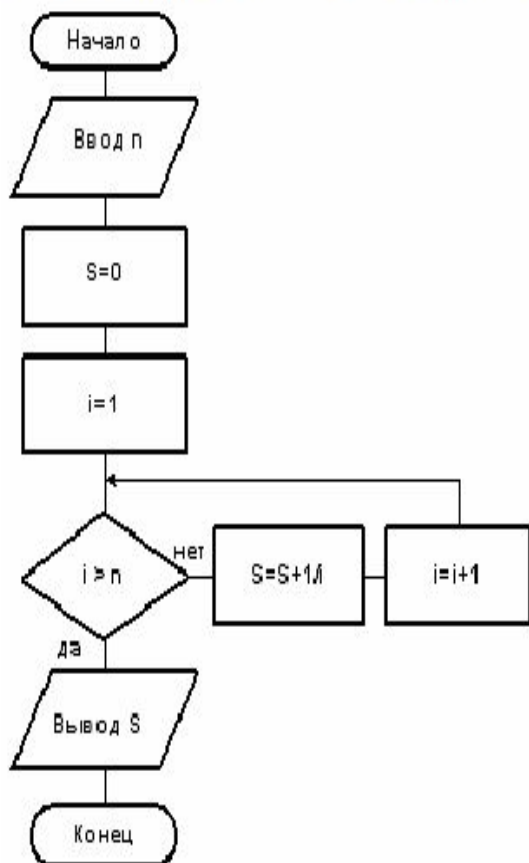
Циклы с предусловием – это такие циклы, в которых до начала выполнения тела цикла проверяется условие выполнения следующего шага цикла



ТЕСТ

Задание N 30.

Значение переменной S после выполнения алгоритма при $n = 4$ будет равно ...



Варианты ответа:

- 0
- $\frac{25}{12}$
- $\frac{11}{6}$
- 4

ТЕСТ

Задание N 33.

В представленном фрагменте программы

```
b := 12;    d := 46
```

```
НЦ ПОКА   d >= b
```

```
    d := d - b
```

```
КЦ
```

тело цикла выполнится _____ раз(а).

Варианты ответа:

- бесконечное число
- 0
- 4
- 3

ЦИКЛЫ С ПОСТУСЛОВИЕМ

Циклы с постусловием – это такие циклы, в которых условие выполнения следующего шага цикла проверяется после выполнения тела цикла



ПРОЦЕДУРЫ И ФУНКЦИИ

Подпрограммы – это программные модули, выполняющие законченное логическое действие, а по окончании работы передающие управление обратно в головную программу. Эти модули можно использовать и в других программах.

Подпрограмма-процедура — это независимая именованная часть программы, которую после однократного описания можно многократно вызвать по имени из последующих частей программы для выполнения определенных действий.

Процедура выполняет последовательность определенных действий и при этом может не возвращать в головную программу какое-либо значение.

Подпрограмма-функция в результате своей работы возвращает в программу некоторое значение.

Имя функции может входить в выражение, как операнд, а процедура **не может** использоваться как операнд в выражении.

ПРОЦЕДУРЫ И ФУНКЦИИ

Данные для обработки подпрограммы-функции и подпрограммы-процедуры получают из вызвавшей их программы.

Подпрограмма получает данные из основной программы:

- **неявно**, т.е. с использованием *глобальных переменных и констант*;
- **явно** – через так называемые *параметры (фактические параметры)*.

ОБЛАСТЬ ДЕЙСТВИЯ ДАННЫХ

Ресурсы основной программы, которые доступны в любой подпрограмме, вызываемой из этой основной программы, получили название **глобальных**.

Подпрограммы в собственном разделе описаний могут объявлять необходимые для работы алгоритма рабочие переменные. Ресурсы, объявленные в разделе описаний подпрограммы, называются **локальными**, и они недоступны в программе, из которой вызывается эта подпрограмма.

Если в подпрограмме объявлена локальная переменная, имя которой совпадает с именем глобальной переменной, то эта глобальная переменная в подпрограмме «перекрывается» (т.е. становится недоступной в подпрограмме).

ПЕРЕДАЧА ДАННЫХ ЧЕРЕЗ ПАРАМЕТРЫ

Для обеспечения контролируемой передачи параметров в подпрограмму и возврата результатов из неё используется механизм *параметров*. В этом случае при описании подпрограммы в ее заголовке описывают список параметров. Эти параметры называют **формальными**, так как для их размещения память не отводится.

При обращении к подпрограмме для каждого формального параметра должно быть указано **фактическое значение** – литерал, константа или переменная того же типа, что и соответствующий формальный параметр.

Порядок, количество и тип формальных и соответствующих фактических параметров должны совпадать.

Результаты подпрограмм должны возвращаться в вызвавшую их программу.

ПЕРЕДАЧА ДАННЫХ ЧЕРЕЗ ПАРАМЕТРЫ

Чтобы отличать параметры подпрограммы, описанные в её заголовке и теле, от параметров, указываемых при вызове подпрограммы, первые принято называть *формальными параметрами*, вторые — *фактическими* параметрами.

При вызове подпрограммы фактические параметры, указанные в команде вызова, становятся значениями соответствующих формальных параметров, чем и обеспечивается передача данных в подпрограмму.

РЕКУРСИВНЫЕ АЛГОРИТМЫ

Рекурсивный алгоритм – алгоритм, использующий в качестве вспомогательного самого себя.

Подпрограмма, обращающаяся сама к себе как к подпрограмме, называется ***рекурсией***. (Рекурсия – «возвращение»).

В рекурсивных подпрограммах выделяют две серии шагов:

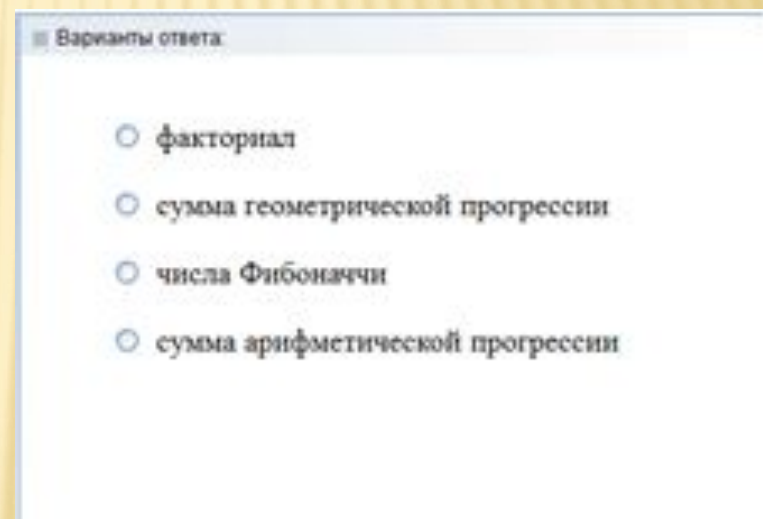
- шаги ***рекурсивного погружения*** подпрограммы в себя до тех пор, пока выбранный параметр не достигнет ***граничного значения***;
- шаги ***рекурсивного выхода*** до тех пор, пока выбранный параметр не достигнет ***начального значения*** .

ТЕСТ

Задана программа, реализующая рекурсивный алгоритм. В программе вычисляется (-ются) ...

```
ВВЕСТИ n
rec=reks(n)
ВЫВЕСТИ rec
END
```

```
ФУНКЦИЯ reks(n)
ЕСЛИ n<0 ТО
  reks=0
ИНАЧЕ
  reks=N+reks(N-1)
КОНЕЦ БЛОКА ЕСЛИ
КОНЕЦ ФУНКЦИИ
```



СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

Структурный подход предполагает использование нескольких основных структур (следование, разветвление, цикл), комбинация которых дает все многообразие алгоритмов и программ.

Данный подход позволяет представлять алгоритмы в виде блочной структуры без нагромождений операторов перехода **GOTO**.

МЕТОД НИСХОДЯЩЕГО ПРОЕКТИРОВАНИЯ

Метод нисходящего проектирования предполагает последовательное разложение общей функции обработки данных на простые функциональные элементы методом «сверху - вниз», от целого к части.

МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ

При модульном подходе сначала определяются *состав* и *подчиненность* функций, а затем – набор *программных модулей*, реализующих эти функции. Однотипные функции реализуются одними и теми же модулями. Функция верхнего уровня обеспечивается главным модулем, который управляет выполнением нижестоящих функций, реализуемых подчиненными модулями.

СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

Структурное программирование основано на *модульной структуре* программного продукта и *типовых управляющих структурах* алгоритмов обработки данных различных программных модулей.

В любой типовой структуре блок, кроме условного (который имеет два выхода), имеет только один вход и один выход. Виды основных управляющих структур – *последовательность, альтернатива (условие выбора), цикл.*

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Большим шагом вперед в развитии технологий программирования было появление объектно-ориентированного программирования.

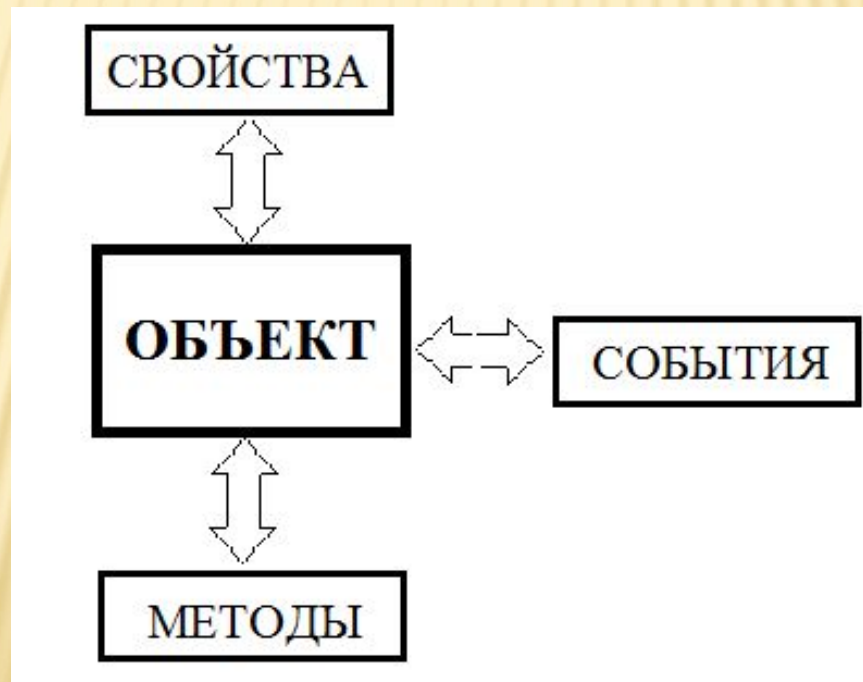
В концепции объектно-ориентированного программирования новый подход заключается в объединении данных и алгоритмов (функций), относящихся к одному типу объектов, в единое описание **классов объектов** (инкапсуляция).

***Класс** – это объединение данных и обрабатывающих их процедур и функций.*

Класс задает свойства и поведение объектов класса – **экземпляров класса**. Класс – это множество объектов с идентичными свойствами. Каждый объект принадлежит некоторому классу.

ПОНЯТИЕ ОБЪЕКТА

Объект — совокупность свойств, методов, событий.
В объектно-ориентированном программировании
понятию объекта соответствует схема:



СВОЙСТВА, МЕТОДЫ, СОБЫТИЯ

Свойствами описываются характеристики объектов (размер, цвет, видимость и т.д.).

Методы – это действия, совершаемые над объектами (ячейку можно почистить – метод Clear, форму показать – метод Show).

События - действия, распознаваемые объектом (щелчок мышью, выход из программы), для которого можно запрограммировать отклик (т.е. реакцию объекта на произошедшее событие).

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Объектно-ориентированный **подход** **к**
программированию:

- программа представляет собой описание объектов, их свойств, совокупностей, отношений между ними, способов их взаимодействия и операций над объектами (методов);
- важное свойство подхода – поддержка механизма обработки *событий*, которые изменяют атрибуты объектов и моделируют их взаимодействие.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Объектно-ориентированное программирование снабжает программные объекты встроенными характеристиками:

Инкапсуляция – это понятие означает, что в качестве единого целого, называемого объектом, рассматриваются некоторая структура данных, определяющая его свойства, и некоторая группа функций (методов).

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Наследование – механизм порождения новых классов, когда порождаемый класс *наследует* данные и методы порождающего класса. Наследование позволяет одним объектам приобретать атрибуты и поведение других. Группы более низкого уровня наследуют характеристики групп более высоких уровней. Кроме того, наследование позволяет модифицировать поведение объектов-потомков, не меняя поведения объектов исходного родительского класса.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Полиморфизм – множественность форм, которые может принимать правило с одним и тем же именем.
Полиморфизм методов – когда метод с одним именем может исполняться по-разному для порождаемого и порождающего классов.

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ

Интегри́рованная среда́ разрабо́тки, ИСР (англ. *IDE*, *Integrated development environment* или *integrated debugging environment*) — система программных средств, используемая программистами для разработки программного обеспечения (ПО).

Частный случай ИСР — среды визуальной разработки, которые включают в себя возможность визуального редактирования интерфейса программы (используемый в VBA).