НАБРОСОК ОТВЕТОВ НА ВОПРОСЫ К ЭКЗАМЕНУ ПО ПРОГРАМИРОВАНИЮ МГПК(МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ) 2021

1. Дайте определение понятию «Обработка информации». Дайте определение понятию «Алгоритм» и перечислите способы его описания. Дайте определение и охарактеризуйте назначение понятия «Системы программирования».

Обработка информации — вся совокупность операций (сбор, ввод, запись, преобразование, считывание, хранение, уничтожение, регистрация), осуществляемых с помощью технических и программных средств, включая обмен по каналам передачи данных.

Алгоритм – это точное предписание исполнителю (человеку или автоматическому устройству), определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату поставленной задачи. В точном предписании задаются указания о выполнении в установленном порядке некоторой системы операций и правила их применения к исходным данным для решения задачи.

К основным изобразительным средствам алгоритмов относятся следующие способы их записи:

– словесный;

– формульно-словесный;

– блок-схемный;

– на языке операторных схем;

– на псевдокоде;

– на алгоритмическом языке.

Системы программирования - это комплекс инструментальных программных средств, предназначенный для работы с программами на одном из языков программирования.

Системы программирования предоставляют сервисные возможности программистам для разработки их собственных компьютерных программ.

В настоящее время разработка любого системного и прикладного программного обеспечения осуществляется с помощью систем программирования, в состав которых в первую очередь входят:

· трансляторы с языков высокого уровня;

· средства редактирования, компоновки и загрузки программ;

· макроассемблеры (машинно-ориентированные языки);

· отладчики машинных программ.

Системы программирования, как правило, включают в себя:

· текстовый редактор (Edit), осуществляющий функции записи и редактирования исходного текста программы;

· загрузчик программ (Load), позволяющий выбрать из директория нужный текстовый файл программы;

· запускатель программ (Run), осуществляющий процесс выполнения программы;

· компилятор (Compile), предназначенный для компиляции или интерпретации исходного текста программы в машинный код с диагностикой синтаксических и семантических (логических) ошибок;

· отладчик (Debug), выполняющий сервисные функции по отладке и тестированию программы;

· диспетчер файлов (File), предоставляет возможность выполнять операции с файлами: сохранение, поиск, уничтожение и т.п.

Ядро системы программирования составляет язык. Существующие языки программирования можно разделить на две группы: процедурные и непроцедурные.

Процедурные (или алгоритмические) программы представляют из себя систему предписаний для решения конкретной задачи. Роль компьютера сводится к механическому выполнению этих предписаний.

Процедурные языки разделяют на языки низкого и высокого уровня.

Языки низкого уровня(машинно-ориентированные) позволяют создавать программы из машинных кодов, обычно в шестнадцатеричной форме. С ними трудно работать, но созданные с их помощью высококвалифицированным программистом программы занимают меньше места в памяти и работают быстрее. С помощью этих языков удобнее разрабатывать системные программы, драйверы (программы для управления устройствами компьютера), некоторые другие виды программ.

Программы на языках высокого уровня близки к естественному (английскому) языку и представляют набор заданных команд.

В общем случае язык – это заданный набор символов и правил, устанавливающих способы комбинации этих символов между собой для записи осмысленных текстов. Основой любого естественного или искусственного языка является алфавит, определяющий набор допустимых символов языка.

1. Перечислите и опишите типы алгоритмов

По типу используемого вычислительного процесса различают линейные(прямые), разветвляющиеся и циклические алгоритмы

Линейные алгоритмы описывают линейный вычислительный процесс, этапы которого выполняются однократно и последовательно один за другим.

Разветвляющийся алгоритм описывает вычислительный процесс, реализация которого происходит по одному из нескольких заранее предусмотренных направлений.

Направления по которым может следовать вычислительный процесс называются ветвями

Циклический алгоритм описывает вычислительный процесс, этапы которого повторяются многократно.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | |  | | |

1. Опишите разветвляющийся алгоритм.

Разветвляющийся алгоритм описывает вычислительный процесс, реализация которого происходит по одному из нескольких заранее предусмотренных направлений

Направления по которым может следовать вычислительный процесс называются ветвями

Во многих случаях требуется, чтобы при одних условиях выполнялась одна последовательность действий, а при других – другая.

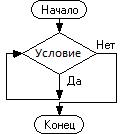
Если пошел дождь, то надо открыть зонт.

Если прозвенел будильник, то надо вставать.

Вычислительный процесс называется ветвящимся, если для его реализации предусмотрено несколько направлений (ветвей). Каждое отдельное направление процесса обработки данных является отдельной ветвью вычислений. Ветвление в программе — это выбор одной из нескольких последовательностей команд при выполнении программы. Выбор направления зависит от заранее определенного признака, который может относиться к исходным данным, к промежуточным или конечным результатам. Признак характеризует свойство данных и имеет два или более значений.  
Ветвящийся процесс, включающий в себя две ветви, называется простым, более двух ветвей — сложным. Сложный ветвящийся процесс можно представить с помощью простых ветвящихся процессов.   
Направление ветвления выбирается логической проверкой, в результате которой возможны два ответа: «да» — условие выполнено и «нет» — условие не выполнено.

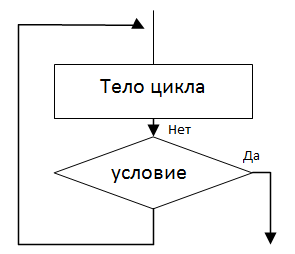
1. Опишите алгоритм цикла с предусловием.

Цикл с предусловием — цикл, который выполняется пока истинно некоторое условие, указанное перед его началом. Это условие проверяется до выполнения тела цикла, поэтому тело может быть не выполнено ни разу (если условие с самого начала ложно).  
Цикл называется итерационным(число повторений тела **цикла** заранее неизвестно), если число его повторений  не задается, а определяется в ходе выполнения цикла. В этом случае одно повторение цикла называется итерацией.

[](https://sites.google.com/site/elektronnyjucebnikmitrofanova/tema-1-algoritm/tema-1-3-vidy-algoritmov/cikl-s-postusloviem/New_FlowChart_2_Part_0.gif?attredirects=0)

1. Опишите алгоритм цикла с постусловием.

Цикл с постусловием — цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело всегда выполняется хотя бы один раз.

[](https://www.sites.google.com/site/elektronnyjucebnikmitrofanova/tema-1-algoritm/tema-1-3-vidy-algoritmov/cikl-s-predusloviem/%D0%A6%D0%B8%D0%BA%D0%BB%20%D1%81%20%D0%BF%D0%BE%D1%81%D1%82%D1%83%D1%81%D0%BB.png?attredirects=0)

1. Перечислите основные типы данных. Опишите операции над переменными основных типов данных. Перечислите и опишите операторы ввода и вывода.

В любой программе, обрабатывающей данные, в качестве операндов в выражениях используются переменные. Часто определяют переменную как пару “имя” - “значение”. Имени соответствует адрес участка памяти, выделенной переменной, а значением является содержимое этого участка.

Способ представления данных компьютером определяется их типом. Кроме того, тип данных определяет, какие действия разрешается выполнять над этими данными.

Ниже перечислены основные стандартные типы данных в pascal:

INTEGER – целочисленные данные в диапазоне от –32768 до 32767, в памяти занимают два байта;

REAL – вещественные числа в диапазоне от 2.9´10-39 (2.9E-39) до 1.7´1038 (1.7E38), занимают шесть байт;

CHAR – отдельный символ, один байт;

STRING – строка символов, количество символов в строке (длина строки) ограничивается числом N в квадратных скобках, занимает N+1 байт (если число N не указано, то максимальная длина строки равна 255 символов);

BOOLEAN – логический тип, имеет два значения: FALSE  (ложь) и TRUE (истина), один байт.

Заметим, что типы INTEGER, CHAR, и BOOLEAN относятся к порядковым типам (ordinal types).

Арифметические операции

 вычисление значений стандартных функций;

 умножение и деление;

 сложение и вычитание.

*Логические выражения* строятся из логических констант и переменных, операций отношения и логических операций. В операциях отношения могут участвовать арифметические и логические выражения, а также символьные данные. Результатом логического выражения является значение TRUE (истина) или FALSE (ложь).

*Операции отношения* : < (меньше), > (больше), = (равно), <= (меньше или равно), >= (больше или равно), < > (не равно). *Логические операции:*

Not --> НЕ - логическое отрицание; And --> И - логическое умножение; Or --> ИЛИ - логическое сложение. Xor --> исключительное ИЛИ. *В логических выражениях* операции выполняются слева направо с соблюдением следующего приоритета:

Not;

\*, /, Div, Mod, And, Shr, Shl ;

+, -, Or, Xor;

=, <>, <, >, <=, >=, in.

К переменным *символьного типа,* определенным, например, как: Var ch : char - кроме операции присваивания, применима функция Ord^h), которая возвращает порядковый номер символа в кодовой таблице.

Для **ввода** данных в языке Pascal предусмотрены READ и READLN а для их вывода WRITE и WRITELN

1. Опишите структуру программы на языке Паскаль.

Любая программа на языке программирования Pascal имеет три составляющие: 1) заголовок; 2) раздел описаний; 3) тело программы.

1) Заголовок – это слово Program , после которого идет название программы латиницей и точка с запятой в конце. Название не может начинаться с цифры, не должно совпадать с зарезервированными словами (begin, end, integer и т.п.) или с названиями переменных из раздела описаний (см. ниже), а также недопустимо использование каких бы то ни было символов (‘@’, ‘%’, ‘&’ и т.п.), кроме подчеркивания. Название желательно давать со смыслом, которое отображало бы суть самой программы. Но заголовок писать не обязательно.

2) Раздел описаний – Раздел где вы вводите константы или переменные – находится после слова Program но перед begin

3) Тело программы – блок операторов, в котором записываются команды для выполнения. Начинается этот блок словом begin («начало»), а заканчивается словом end. («конец») с точкой в конце.

1. Опишите условные операторы If и Case*.* Опишите конструкции с составным условным оператором и с использованием сложного условия.

Когда выполнение основной ветки программы доходит до условного оператора if - else, то в зависимости от результата логического выражения в его заголовке выполняются разные блоки кода. Если логическое выражение вернуло true, то выполняется один блок (в Паскале начинается со слова then), если false – то другой (начинается со слова else). После выполнения одного из вложенных блоков кода, ход программы возвращается в основную ветку. Другой вложенный блок не выполняется.

Бывают неполные формы условных операторов. В таком случае вложенный в if блок кода выполняется только в случае true логическом выражении заголовка. В случае false выполнение программы сразу передается в основной блок. Понятно, что ветка else в таком случае отсутствует.

Кроме оператора if в языке программирования Паскаль предусмотрен так называемый переключатель case. Его можно трактовать как некий вопрос, имеющий большое число ответов (а не только два, как это имеет место в операторе if-else). Однако в отличие от if, case имеет ряд принципиальных ограничений. Его формат следующий:

case селектор of

значение1: оператор1;

значение2: оператор2;

значение3: оператор3;

...

else операторN

end;

В заголовке оператора case вместо логического выражения фигурирует переменная, которую называют селектором. До этого в программе ей присваивается какое-либо значение. Эта переменная может иметь только перечисляемый тип (например, она не может быть вещественного типа). По ходу выполнения оператора case, значение переменной-селектора сравнивается с различными, описанными в нем альтернативами (метками-значениями). Как только совпадение будет найдено, то выполняется блок кода при данной метке и происходит выход в основную ветку программы. Значения-метки являются константами, которые может принимать селектор. Их тип и тип селектора должны быть совместимы по присваиванию.

Если совпадений не будет, то выполняется блок else. Если блок else отсутствует (он является не обязательным), то никакой блок кода в операторе case не выполняется.

1. Перечислите операторы организации циклов. Опишите цикл For.

Если число повторений тела цикла заранее известно, то используется **оператор цикла** for, который также часто называют оператором цикла с параметром.

Оператор for состоит из двух частей: тела цикла и заголовка, который предназначен для описания начального и конечного значений параметра цикла, а также варианта его изменения.

В зависимости от направления изменения параметра цикла (возрастание - to или убывание - downto) в языке Паскаль оператор цикла for может быть записан в одной из двух форм:

for параметр := нач\_знач to кон\_знач do

оператор;

или

for параметр := нач\_знач downto кон\_знач do

оператор;

1. Перечислите операторы организации циклов. Опишите цикл While.

Оператор цикла while содержит в себе выражение, которое управляет повторным выполнением оператора (который может быть составным оператором).

WHILE выражение DO

BEGIN

Внутренний оператор;

END;

Выражение, с помощью которого производится управление повторением оператора, должно иметь булевский тип. Вычисление его производится до того, как внутренний оператор будет выполнен. Внутренний оператор выполняется повторно до тех пор, пока выражение принимает значение True. Если выражение с самого начала принимает значение False, то оператор, содержащийся внутри оператора цикла while, не выполняется ни разу.

1. Перечислите операторы организации циклов. Опишите цикл Repeat.

В операторе цикла repeat выражение, которое управляет повторным выполнением последовательности операторов, содержится внутри оператора repeat.

REPEAT

Внутренний оператор;

UNTIL логическое выражение;

Результатом выражения должен быть результат булевского типа.

Операторы, заключенные между ключевыми словами repeat и until, выполняются последовательно до тех пор, пока результат выражения не примет значения True. Последовательность операторов выполняется по крайней мере один раз, поскольку вычисление выражения производится после каждого выполнения последовательности операторов.

1. Дайте определение понятию «Массив». Опишите способы объявления массивов. Опишите как осуществляется доступ к элементам массива

Массив — это пронумерованная последовательность величин одинакового типа, обозначаемая одним именем. Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом. Каждое из значений, составляющих массив, называется его компонентой (или элементом массива).

Массив данных в программе рассматривается как переменная структурированного типа. Массиву присваивается имя, посредством которого можно ссылаться как на массив данных в целом, так и на любую из его компонент.

Вообще, массив – однородный, упорядоченный структурированный тип данных с прямым доступом к элементам.

Переменные, представляющие компоненты массивов, называются переменными с индексами в отличие от простых переменных, представляющих в программе элементарные данные. Индекс в обозначении компонент массивов может быть константой, переменной или выражением порядкового типа (целочисленный, логический, символьный, перечислимый, диапазон).

Если за каждым элементом массива закреплен только один его порядковый номер, то такой массив называется линейным. Вообще количество индексов элементов массива определяет размерность массива. По этом признаку массивы делятся на одномерные (линейные), двумерные, трёхмерные и т.д.

Объявление массива выполняется с помощью ключевого слова ARRAY с указанием размерности массива и типа его элементов.

Например, массив в разделе описания переменных можно задать так:

Var Mass: array[1..10] of integer;

Для обращения к массиву используется его имя (также как для обычной переменной)

Для обращения к элементу массива используется имя массива и индекс элемента в квадратных скобках

1. Дайте определение понятию «Массив». Опишите алгоритм ввода и вывода одномерного массива.

Массив — это пронумерованная последовательность величин одинакового типа, обозначаемая одним именем. Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом. Каждое из значений, составляющих массив, называется его компонентой (или элементом массива).

Массив данных в программе рассматривается как переменная структурированного типа. Массиву присваивается имя, посредством которого можно ссылаться как на массив данных в целом, так и на любую из его компонент.

Вообще, массив – однородный, упорядоченный структурированный тип данных с прямым доступом к элементам.

Переменные, представляющие компоненты массивов, называются переменными с индексами в отличие от простых переменных, представляющих в программе элементарные данные. Индекс в обозначении компонент массивов может быть константой, переменной или выражением порядкового типа (целочисленный, логический, символьный, перечислимый, диапазон).

Если за каждым элементом массива закреплен только один его порядковый номер, то такой массив называется линейным. Вообще количество индексов элементов массива определяет размерность массива. По этом признаку массивы делятся на одномерные (линейные), двумерные, трёхмерные и т.д

Для ввода массива можно использовать любой цикл.

Вывод одномерного массива осуществляется аналогично.

В программе вместо операторов Read или Readln используются операторы Write или Writeln. Но просто заменить одни операторы на другие здесь недостаточно. Для того чтобы выводимые значения не сливались между собой, надо явным образом вставлять между ними разделитель – пробел или перевод строки. Приведем два возможных способа вывода массива:

For i := 1 To n Do Write (X[i],’ ‘);

For i := 1 To n Do Writeln (x[i]).

На первый взгляд второй способ может показаться более простым и удобным, но это далеко не всегда так. Результат работы такой программы зачастую неудобно, а то и просто невозможно анализировать. Ведь каждый элемент массива будет располагаться в отдельной строке, следовательно, мы не сможем увидеть более 25 элементов одновременно. Кроме того, очень часто массив требуется распечатать дважды, чтобы сравнить состояние массива до обработки и результат его обработки. В этом случае сравнение состояний массива гораздо удобнее проводить, если они распечатаны в двух соседних строках, а элементы выровнены по столбцам, то есть к варианту 1 должна быть добавлена еще и форматная печать (указано количество позиций, которое должно отводиться на печать одного элемента).

1. Дайте определение понятию «Двумерный массив». Опишите алгоритм ввода и вывода двумерного массива.

Двумерный массив - это одномерный массив, элементами которого являются одномерные массивы. Другими словами, это набор однотипных данных, имеющий общее имя, доступ к элементам которого осуществляется по двум индексам.

Существует несколько способов объявления двумерного массива Паскаля.

Мы уже умеем описывать одномерные массивы, элементы которых могут иметь любой тип, а, следовательно, и сами элементы могут быть массивами. Рассмотрим следующее описание типов и переменных:

Type   
Vector = array [1..5] of <тип\_элементов>;   
Matrix= array [1..10] of vector;   
Var m: matrix;

Мы объявили двумерный массив Паскаля m, состоящий из 10 строк, в каждой из которых 5 столбцов. При этом к каждой i -й строке можно обращаться m [ i ], а каждому j -му элементу внутри i -й строки – m [ i , j ].

Определение типов для двумерных массивов Паскаля можно задавать и в одной строке:

Type   
Matrix= array [1..5] of array [1..10] of < тип элементов >;   
или еще проще:   
type   
matrix = array [1..5, 1..10] of <тип элементов>;

Обращение к элементам двумерного массива имеет вид: M [ i , j ]. Это означает, что мы хотим получить элемент, расположенный в i -й строке и j -м столбце. Тут главное не перепутать строки со столбцами, а то мы можем снова получить обращение к несуществующему элементу. Например, обращение к элементу M [10, 5] имеет правильную форму записи, но может вызвать ошибку в работе программы.

1. Опишите процесс обработки данных в массиве. Дайте определение понятиям «Поиск» и «Сортировка». Опишите алгоритм поиска минимального/максимального элемента.

Рассмотрим часто встречающуюся задачу упорядочения членов числовой последовательности по какому-либо признаку.

Пример: упорядочить члены числовой последовательности по возрастанию.

Используем метод упорядочения, носящий имя метод "пузырька". Здесь име-

ет место аналогия с "всплыванием" пузырька в жидкости под действием силы

Архимеда. Будем сравнивать пары соседних элементов последовательно

справа налево и переставлять элементы в паре, если они стоят в порядке, не

соответствующем требуемому:

5,3,2,4,1 → 5,3,2,1,4 → 5,3,1,2,4 → 5,1,3,2,4 → 1,5,3,2,4

В начале просмотра присвоим некоторой логической переменной значение

true:

p:=true;

Если при просмотре пар была совершена хотя бы одна перестановка, изме-

ним значение логической переменной на противоположное:

p:=false;

Это означает, что последовательность еще не была упорядочена и просмотр

пар надо повторить. Цикл просмотров заканчивается, если после очередного

просмотра выполняется условие:

p=true;

Последовательность зададим в программе как массив типизированных кон-

стант из 10 элементов — целых чисел.

Обработка элементов двумерных массивов (матриц) обычно выполняется с

помощью двойного цикла. Один цикл задает индекс строки, другой — индекс

столбца.

При решении различных задач обработки информации часто возникает необ-

ходимость в использовании последовательностей символов. Такую последо-

вательность можно описать как массив символов, однако в Паскале для таких

целей имеется более удобный тип данных string[n] строка из n символов,

где n<=255. Отметим, что элемент строки с индексом 0 хранит заданную при

описании длину строки n, а элементы строки индексируются целыми числами

от 1 до n. Способы описания переменных — строк — аналогичны описанию

массивов.

Строковый тип определяется в разделе описания типов, переменные этого

типа — в разделе описания переменных:

type word : string[20];

var a,b,c : word;

Можно совместить описание строкового типа и соответствующих пере-

менных в разделе описания переменных:

var a,b,c : string[20];

d : string[30];

Можно определить строковую переменную и ее начальное значение как

констант-строку:

const l:string[11]='информатика';

Символы, составляющие строку, занумерованы слева направо; к ним можно

обращаться с помощью индексов, как к элементам одномерного массива.

Сортировка - это процесс упорядочивания наборов данных одного типа по возрастанию или убыванию значения какого-либо признака.

Имеется два вида алгоритмов сортировки: сортировка массивов, которые могут находиться как в операционной памяти, так и на диске в виде файла прямого доступа, и сортировка последовательных файлов, находящихся на дисках или магнитных лентах

Поиском называется алгоритм, который на входе воспринимает некоторое значение х и определяет запись, ключ которой совпадает со значением х. При этом на выходе такой алгоритм может выдать либо найденную запись, либо указатель на нее. Х называется аргументом поиска.

Алгоритм поиска минимального (максимального) элемента массива довольно очевиден: сначала делается предположение, что первый элемент массива является минимальным (максимальным), затем остальные элементы массива последовательно сравниваются с этим элементом. Если во время очередной проверки обнаруживается, что проверяемый элемент меньше (больше) принятого за минимальный (максимальный), то этот элемент становится минимальным (максимальным) и продолжается проверка оставшихся элементов.

1. Дайте определение понятиям «Подпрограммы», «Процедуры» и «Функции». Перечислите отличие процедуры от функции.

Подпрограмма – относительно самостоятельная часть программы, имеющая свое имя и выполняющая определенные действия.

Подпрограммы повышают надежность и наглядность программ, т.к. позволяют разрабатывать и отлаживать каждый блок программы независимо (например, разными людьми).

Структура подпрограммы почти полностью повторяет структуру всей программы и состоит из следующих частей:

заголовок подпрограммы

раздел описаний

тело подпрограммы

В Паскаль имеются два вида подпрограмм – процедуры и функции. Они отличаются назначеием и способом их использования. Процедуры служат для выполнения определенной последовательности действий, направленных на изменение программной обстановки (изменение значений переменных, ввод/вывод данных и т.п.). Функции для вычисления значения выражения.

Функция – подпрограмма, в которой выполняется некоторый набор действий, в результате чего имя функции получает некоторое единственное значение, передаваемое в вызываемую подпрограмму, при этом управление при выходе из функции возвращается в точку ее вызова.

Процедура – подпрограмма, в которой производится некоторый набор действий, при этом имени процедуры ничего не присваивается: процедура не имеет возвращаемого значения. Результаты в вызывающую программу передаются через механизм параметров или глобальные переменные. Управление возвращается за точку вызова процедуры.

1. Дайте определение понятию «Пользовательские подпрограммы». Опишите формальные, фактические входные и выходные параметры.

Формальные параметры — это идентификаторы входных данных для подпрограммы. Если формальные параметры получают конкретные значения, то они называются фактическими. Формальные параметры могут получить конкретные значения только в той программе, где производится обращение к данному модулю-подпрограмме. Тип и порядок записи фактических параметров должны быть такими же, как и формальных параметров. В противном случае результат работы программы будет непредсказуемым. Из этого следует, что фактические параметры используются при обращении к подпрограмме из основной, а формальные параметры — только в самом модуле.

Подпрограмма с параметрами используется для записи многократно повторяющихся действий при разных исходных данных.

При составлении подпрограмм с параметрами надо соблюдать следующие правила:

1) каждая подпрограмма имеет свое имя и список формальных параметров;

2) процедура из основной программы вызывается командой вызова, которая по форме ничем не отличается от вызова команды исполнителя. Результат присваивается одной или нескольким переменным, которые находятся в списке формальных параметров. Но результатом могут быть, конечно, не только значения переменных, но какое-либо действие, выполненное ЭВМ

Входные параметры – это переменные, в которые передаются значения переменных или констант основной программы в Процедуру, и не предназначены для возврата

Выходные параметры– это переменные (не константы), которые передаются основной программой в Процедуру, но их измененные в Процедуре значения, будут возвращены основной программе. При объявлении таких параметров указывается служебное слово Var.

1. Опишите символьные и строковые переменные. Перечислите и опишите процедуры и функции для работы с переменными символьного и строкового типа

Символьная  переменная — величина, значением которой является один символ (буква, цифра, знак).  
  Строковая переменная — величина, значением которой является последовательность символов. Значения символьных и строковых величин заключают в апострофы (одинарные кавычки). Например, 'А', '5', '+', 'Минск'.  
Длина строки — это количество символов в строке. Максимальная длина строки в Pascal ABC — 255 символов.

При работе со строками, как правило, возникает необходимость выполнять их копирование, вставку, удаление или поиск. Для эффективной реализации этих действий в Паскале предусмотрены стандартные процедуры и функции. Они кратко описаны ниже.

Функция Concat (s1, s2, ..., sn) возвращает строку, являющуюся слиянием строк s1, s2, ..., sn.

Функция Copy (s, start, len) возвращает подстроку длиной len, начинающуюся с позиции start строки s.

Процедура Delete (s, start, len) удаляет из строки s, начиная с позиции start, подстроку длиной len.

Процедура Insert (subs, s, start) вставляет в строку s подстроку subs, начиная с позиции start.

Функция Length (s) возвращает фактическую длину строки s, результат имеет тип byte.

Функция Pos (subs, s) ищет вхождение подстроки subs в строку s и возвращает номер первого символа subs в s или нуль, если subs не содержится в s.

Процедура **Str (x, s)** преобразует числовое значение x в строку s, при этом для x может быть задан формат, как в процедурах вывода write и writeln

Процедура **Val (s, x, errcode)** преобразует строку s в значение числовой переменной x, при этом строка s должна содержать символьное представление числа. В случае успешного преобразования переменная errcode равна нулю. Если же обнаружена ошибка, то errcode будет содержать номер позиции первого ошибочного символа, а значение x не определено.

1. Дайте определение понятию «Запись». Опишите порядок создания и использования записей

Запись – это структура данных, объединяющая элементы одного или различ- ных типов, называемые полями. Записи удобны для создания структурирован- ных баз данных с разнотипными элементами

Пример описания записи:

Type Car=Record

Number: integer; {Номер}

Marka: string[20]; {Марка автомобиля}

FIO: string[40]; {Фамилия, инициалы владельца}

Address: string[60] {Адрес владельца}

End;

Var Mashina: Car;

В данном примере была объявлена запись с именем Car, у которой имеется 4 поля: номер, название марки машины, ФИО владельца и его адрес.

Идентификатор поля должен быть уникален только в пределах записи, однако лучше его сделать уникальным в пределах всей программы. Объем памяти, необходимый для записи, складывается из длин полей. Значения полей записи могут быть использованы в выражениях. Обращение к значению поля осуществляется с помощью идентификатора переменной и идентификатора поля, разделенных точкой. Такая комбинация называется составным именем.

Например, доступ к полям записи Car осуществляется как: Mashina.Marka, Mashina.FIO, Mashina.Number. Составное имя можно использовать везде, где допустимо применение типа поля. Для присваивания полям значений используется оператор присваивания.

Пример присваивания полям записи Mashina:

Mashina. Number:=164536l;

Mashina.Marka:=’ГАЗ-24’;

Mashina.FIO:=’Иванов И.И’;

Mashina. Address:=’ул.Пушкина 12-30’;

Составные имена можно использовать в операторах ввода-вывода:

Read (Mashina. Number, Mashina. FIO, Mashina. Address);

Write(Mashina. Number:4, Mashina. FIO:12, Mashina. Address:25);

Допускается применение оператора присваивания и к записям в целом, если они имеют один и тот же тип. Например,

Mash:=Mashina;

После выполнения этого оператора значения полей записи Mash станут равны значениям соответствующих полей записи Mashina.

В ряде задач удобно пользоваться массивами из записей. Их можно описать следующим образом:

Type Car=Record

Number:Integer;

Marka:String[20];

FIO:String[40];

Address:String[60];

End;

Var Mashins: array [1..20] of Car;

Обращение к полям такой записи имеет громоздкий вид, для решения этой проблемы в языке Паскаль предназначен оператор With, который имеет следующий формат:

With <переменная типа запись> do <оператор>;

Один раз, указав переменную типа запись в операторе With, можно работать с именами полей как с обычными переменными.

1. Дайте определение понятию «Сортировка». Опишите основные методы сортировки.

Сортировка - это процесс упорядочивания наборов данных одного типа по возрастанию или убыванию значения какого-либо признака.

Имеется два вида алгоритмов сортировки: сортировка массивов, которые могут находиться как в операционной памяти, так и на диске в виде файла прямого доступа, и сортировка последовательных файлов, находящихся на дисках или магнитных лентах.

 Имеется три способа сортировки массивов:

сортировка обменом(метод пузырька);

сортировка выбором;

сортировка вставкой

Сортировка простыми обменами, сортировка пузырьком (англ. bubble sort) — один из квадратичных алгоритмов сортировки.

Алгоритм состоит в повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами. За каждый проход по массиву как минимум один элемент встает на свое место, поэтому необходимо совершить не более n−1 проходов, где n размер массива, чтобы отсортировать массив.

Сортировка выбором;

Просто и незатейливо — проходим по массиву в поисках максимального элемента. Найденный максимум меняем местами с последним элементом. Неотсортированная часть массива уменьшилась на один элемент (не включает последний элемент, куда мы переставили найденный максимум). К этой неотсортированной части применяем те же действия — находим максимум и ставим его на последнее место в неотсортированной части массива. И так продолжаем до тех пор, пока неотсортированная часть массива не уменьшится до одного элемента.

Сортировка вставкой

Сортировка вставками (Insertion Sort) — это простой алгоритм сортировки. Суть его заключается в том что, на каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем. Стоит отметить что массив из 1-го элемента считается отсортированным.  
  
Словесное описание алгоритма звучит довольно сложно, но на деле это самая простая в реализации сортировка. Каждый из нас, не зависимо от рода деятельности, применял алгоритм сортировки, просто не осознавал это:) Например когда сортировали купюры в кошельке — берем 100 рублей и смотрим — идут 10, 50 и 500 рублёвые купюры. Вот как раз между 50 и 500 и вставляем нашу сотню:) Или приведу пример из всех книжек — игра в карточного «Дурака». Когда мы тянем карту из колоды, смотрим на наши разложенные по возрастанию карты и в зависимости от достоинства вытянутой карты помещаем карту в соответствующее место.

1. Дайте определение понятию «Сортировка». Опишите метод пузырька

Сортировка - это процесс упорядочивания наборов данных одного типа по возрастанию или убыванию значения какого-либо признака.

Имеется два вида алгоритмов сортировки: сортировка массивов, которые могут находиться как в операционной памяти, так и на диске в виде файла прямого доступа, и сортировка последовательных файлов, находящихся на дисках или магнитных лентах.

Сортировка простыми обменами, сортировка пузырьком (англ. bubble sort) — один из квадратичных алгоритмов сортировки.

Алгоритм состоит в повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами. За каждый проход по массиву как минимум один элемент встает на свое место, поэтому необходимо совершить не более n−1 проходов, где n размер массива, чтобы отсортировать массив.

1. Дайте определение понятию «Файл». Перечислите виды файлов. Опишите процедуры работы с файлами

Файл — это определенное количество информации (программа или данные), имеющее имя и хранящееся в долговременной (внешней) памяти.

Работа с файлами реализуется средствами операционных систем. Многие операционные системы приравнивают к файлам и обрабатывают сходным образом и другие ресурсы

В подавляющем большинстве случаев файл хранит в себе какой-то определенный тип данных - текст, графическую информацию, про­граммный код и так далее (хотя бывают и некие «комбинированные» файлы, включающие, к примеру, картинку, текст и элемент программы). Поэтому можно говорить о том, что существует множество типов файлов, которые пользователь и компьютер должны безошибочно раз­личать. Например:

· Исполняемые файлы(программы).

· Файлы конфигурации, содержащие параметры, необходимые для работы программы.

· Файлы-библиотеки, содержащие описания типовых процедур, ко­торые могут использоваться сразу несколькими программами.

· Документы - т. е. некие совокупности информации, создаваемые пользователем (или компьютером) с помощью программ .

Есть и множество других типов файлов – все их не опишешь

Любые дисковые файлы становятся доступными программе после связывания их с файловой переменной, объявленной в программе. Все операции в программе производятся только с помощью связанной с ним файловой переменной.

Процедуры работы с файлами

Assign(f, FileName)

связывает файловую переменную f с физическим файлом, полное имя которого задано в строке FileName. Установленная связь будет действовать до конца работы программы, или до тех пор, пока не будет сделано переназначение.

После связи файловой переменной с дисковым именем файла в программе нужно указать направление передачи данных (открыть файл). В зависимости от этого направления говорят о чтении из файла или записи в файл.

Reset(f)

открывает для чтения файл, с которым связана файловая переменная f. После успешного выполнения процедуры Reset файл готов к чтению из него первого элемента. Процедура завершается с сообщением об ошибке, если указанный файл не найден.

Если f - типизированный файл, то процедурой reset он открывается для чтения и записи одновременно.

Rewrite(f)

открывает для записи файл, с которым связана файловая переменная f. После успешного выполнения этой процедуры файл готов к записи в него первого элемента. Если указанный файл уже существовал, то все данные из него уничтожаются.

Close(f)

закрывает открытый до этого файл с файловой переменной f. Вызов процедуры Close необходим при завершении работы с файлом. Если по какой-то причине процедура Close не будет выполнена, файл все-же будет создан на внешнем устройстве, но содержимое последнего буфера в него не будет перенесено.

EOF(f): boolean

возвращает значение TRUE, когда при чтении достигнут конец файла. Это означает, что уже прочитан последний элемент в файле или файл после открытия оказался пуст.

Rename(f, NewName)

позволяет переименовать физический файл на диске, связанный с файловой переменной f. Переименование возможно после закрытия файла.

Erase(f)

уничтожает физический файл на диске, который был связан с файловой переменной f. Файл к моменту вызова процедуры Erase должен быть закрыт.

IOResult

возвращает целое число, соответствующее коду последней ошибки ввода - вывода. При нормальном завершении операции функция вернет значение 0. Значение функции IOResult необходимо присваивать какой-либо переменной, так как при каждом вызове функция обнуляет свое значение. Функция IOResult работает только при выключенном режиме проверок ошибок ввода - вывода или с ключом компиляции {$I-}.

типизированный файл - это последовательность компонент любого заданного типа (кроме типа "файл"). Доступ к компонентам файла осуществляется по их порядковым номерам. Компоненты нумеруются, начиная с 0. После открытия файла указатель (номер текущей компоненты) стоит в его начале на нулевом компоненте. После каждого чтения или записи указатель сдвигается к следующему компоненту.

Запись в файл:

Write(f, список переменных);

Процедура записывает в файл f всю информацию из списка переменных.

Чтение из файла:

Read(f, список переменных);

Процедура читает из файла f компоненты в указанные переменные. Тип файловых компонент и переменных должны совпадать. Если будет сделана попытка чтения несуществующих компонент, то произойдет ошибочное завершение программы. Необходимо либо точно рассчитывать количество компонент, либо перед каждым чтением данных делать проверку их существования (функция eof, см. выше)

Смещение указателя файла:

Seek(f, n);

Процедура смещает указатель файла f на n-ную позицию. Нумерация в файле начинается с 0.

Определение количества компонент:

FileSize(f): longint;

Функция возвращает количество компонент в файле f.

Определение позиции указателя:

FilePos(f): longint;

Функция возвращает порядковый номер текущего компонента файла f.

Отсечение последних компонент файла:

Truncate(f);

Процедура отсекает конец файла, начиная с текущей позиции включительно.

Чтение из текстового файла:

Read(f, список переменных);

ReadLn(f, список переменных);

Процедуры читают информацию из файла f в переменные. Способ чтения зависит от типа переменных, стоящих в списке. В переменную char помещаются символы из файла. В числовую переменную: пропускаются символы-разделители, начальные пробелы и считывается значение числа до появления следующего разделителя. В переменную типа string помещается количество символов, равное длине строки, но только в том случае, если раньше не встретились символы конца строки или конца файла. Отличие ReadLn от Read в том, что в нем после прочтения данных пропускаются все оставшиеся символы в данной строке, включая метку конца строки. Если список переменных отсутствует, то процедура ReadLn(f) пропускает строку при чтении текстового файла.

Запись в текстовый файл:

Write(f, список переменных);

WriteLn(f, список переменных);

Открытие нетипизированного файла:

Reset(f, BufSize)

Rewrite(f, BufSize)

Параметр BufSize задает число байтов, считываемых из файла или записываемых в него за одно обращение. Минимальное значение BufSize - 1 байт, максимальное - 64 К байт. Если BufSize не указан, то по умолчанию он принимается равным 128.

Чтение данных из нетипизированного файла:

BlockRead(f, X, Count, QuantBlock);

1. Дайте определение понятию «Рекурсия». Перечислите виды рекурсии. Опишите рекурсивные процедуры и функции.

Рекурсия – это способ описания объектов или вычислительных процессов через самих себя

Рекурсивное программирование позволяет описать повторяющийся процесс без явного использования операторов цикла, например:

function Fact(n: integer): integer;

begin

if n=0 then Fact:=1

else Fact:= Fact (n-1) \* n

end;

Рекурсивное описание должно содержать хотя бы одну альтернативу, не использующую рекурсивный вызов, то есть явное определение для некоторых значений аргументов подпрограммы. Эта альтернатива – нерекурсивный случай – завершает последовательность рекурсивных вызовов. В противном случае процесс вычислений будет бесконечным.

Тогда рекурсивное к ней обращение позволяет вычислить сумму : S := Sum (m, Sum (n,x)); Оба типа рекурсии могут присутствовать одновременно. Пример такого сочетания – функция Аккермана:

Нерекурсивная реализация функции Аккермана потребует использования матрицы для хранения промежуточных значений

Рекурсия – очень мощный инструмент решения многих задач, но применять её следует только тогда, когда природа самой решаемой задачи рекурсивна.

Рекурсивный вызов может быть косвенным. В этом случае подпрограмма обращается к себе опосредованно, путем вызова другой подпрограммы, в которой содержится обращение к первой. Согласно правилам языка Паскаль каждый идентификатор перед употреблением должен быть описан. Поэтому в случае взаимного обращения подпрограмм друг к другу необходимо использовать опережающее описание

Опережающее описание заключается в том, что объявляется только заголовок подпрограммы, а ее тело заменяется зарезервированным словом forward. Следовательно, далее можно использовать обращение к подпрограмме, так как известны ее формальные параметры.

Тело подпрограммы начинается заголовком без указания ранее приведенного списка формальных параметров (и типа результата в случае функции).

1. Опишите объектно-ориентированное программирование. Дайте определения понятиям «Полиморфизм», «Наследование», «Инкапсуляция». Опишите понятие объекта, метода, свойства, события.

Объе́ктно-ориенти́рованное программи́рование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования

Инкапсуляция

Инкапсуляция —это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.

Наследование

Наследование —это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс —потомком, наследником или производным классом.

Полиморфизм

Полиморфизм —это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Объект

Сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса или копирования прототипа (например, после запуска результатов компиляции и связывания исходного кода на выполнение).

Метод

Ме́тод в объектно-ориентированном программировании — это функция или процедура, принадлежащая какому-то классу или объекту. Как и процедура в процедурном программировании, метод состоит из некоторого количества операторов для выполнения какого-то действия и имеет набор входных аргументов.

Свойство

Свойство — способ доступа к внутреннему состоянию объекта, имитирующий переменную некоторого типа. Обращение к свойству объекта выглядит так же, как и обращение к структурному полю (в структурном программировании), но, в действительности, реализовано через вызов функции.

События

Событие в объектно-ориентированном программировании — это сообщение, которое возникает в различных точках исполняемого кода при выполнении определённых условий.

События предназначены для того, чтобы иметь возможность предусмотреть реакцию программного обеспечения.

Для решения поставленной задачи создаются обработчики событий: как только программа попадает в заданное состояние, происходит событие, посылается сообщение, а обработчик перехватывает это сообщение. В общем случае в обработчик не передаётся ничего, либо передаётся ссылка на объект, инициировавший (породивший) обрабатываемое событие. В особых случаях в обработчик передаются значения некоторых переменных или ссылки на какие-то другие объекты, чтобы обработка данного события могла учесть контекст возникновения события.

Самое простое событие — это событие, сообщающее о начале или о завершении некоторой процедуры. Событие, по сути, сообщает об изменении состояния некоторого объекта. Наиболее наглядно события представлены в пользовательском интерфейсе, когда каждое действие пользователя порождает цепочку событий, которые, затем обрабатываются в приложении.

1. Опишите стандартные и пользовательские модули. Опишите структуру модуля.

Модули по отношению к основной части программы напоминают подпрограммы (процедуры и функции). Но по определению они являются самостоятельными программами, ресурсы которых могут быть задействованы в других программах. Кроме того описание модулей происходит вне вызывающего приложения, а в отдельном файле, поэтому модуль – это отдельно компилируемая программа. Файл скомпилированного модуля (именно такой нужен для использования) будет иметь расширение предусмотренное средой программирования (например, .tpu, .ppu, .pcu).

Модули создаются, как правило, для обеспечения компактности кода, о чем приходиться заботиться крупным проектам. Стоит также отметить, что использование модулей в каком-то смысле снимает ограничение на сегментацию памяти, так как код каждого модуля располагается в отдельном сегменте.

Структура модуля выглядит так:

Unit <имя модуля>;  
Interface  
<интерфейсная часть>  
Implementation  
<исполняемая часть>  
Begin  
<инициализация>  
End.

В состав среды Delphi входит набор модулей. Все модули можно разбить на две группы: системные модули и модули визуальных компонентов.

К системным модулям относятся System, SysUtils, ShareMem, Math. В них содержатся наиболее часто используемые в программах типы данных, константы, переменные, процедуры и функции. Модуль System — это сердце среды Delphi; содержащиеся в нем подпрограммы обеспечивают работу всех остальных модулей системы. Модуль System подсоединяется автоматически к каждой программе и его не надо указывать в операторе uses.

Модули визуальных компонентов (VCL — Visual Component Library) используются для визуальной разработки полнофункциональных GUI- приложений — приложений с графическим пользовательским интерфейсом (Graphical User Interface). Эти модули в совокупности представляют собой высокоуровневую объектно-ориентированную библиотеку со всевозможными элементами пользовательского интерфейса: кнопками, надписями, меню, панелями и т. д. Кроме того, модули этой библиотеки содержат простые и эффективные средства доступа к базам данных. Данные модули подключаются автоматически при помещении компонентов на форму, поэтому вам об этом заботиться не надо. Их список слишком велик, поэтому мы его не приводим.

1. Опишите основные компоненты среды Delphi. Охарактеризуйте понятия «Форма», «Палитра компонентов», «Инспектор объектов», «Редактор кода».

Рассмотрим наиболее часто используемые компоненты в Delphi

Frame ( Рамка) Наравне с формой служит контейнером для размещения других компонентов

MainMenu( Главное Меню) Главное меню программы. Способно создавать и обслуживать сложные иерархические меню

PopupMenu(Всплывающие Меню) Создание вспомогательных меню, появляющихся после нажатия правой кнопки мыши

Label( Метка) Размещение в окне надписей

Edit( Поле с редактированием) – Поле для ввода текста

Memo(Заметка) Многострочное поле для ввода, отображения или редактирования многострочного текста

Button(Кнопка) Командная кнопка. Обработчик события OnClick служит для выполнения определенных программистом действий при его выполнении

CheckBox(Переключатель). Состояние определяется логическим свойством Checked - позволяет пользователю управлять параметром с двумя состояниями — включено и отключено

RadioButton(Круговая кнопка). Служит для выбора одного из нескольких вариантов действий или чего ни будь другого – Может объединятся в группу для переключения

ComboBox(Комбинированый Список) - Сочетание выпадающего списка (раскрывающегося при щелчке мыши) и однострочного текстового поля, которое позволяет пользователю ввести значение вручную или выбрать из списка.

ListBox(Список) - Содержит список опций и контролирует текущий выбор

ScrollBar (Полоса управления) Вертикальная или горизонтальная полоса прокрутки по бокам компонента

GroupBox(Группа элементов) Группирует нескольких связанных по смыслу компонентов

RadioGroup Группа зависимых переключателей.

ActionList(Список действий) Служит для определения реакции программы на действия пользователя, свя-занные с выбором одного из группы однотипных управляющих элементов (опции меню).

Panel(Панель) Служит для объединения нескольких компонентов или как декоративный элемент интерфейса

**Форма**— это важнейший визуальный компонент. Формы представляют собой видимые окна Windows и являются основной частью практически любого приложения. Термины «форма» и «окно» — синонимы, т. е. обозначают одно и то же.

Для работы с формой предназначен компонент **Form** класса TForm. С создания формы начинается конструирование приложения. В форме размещаются визуальные компоненты, образующие интерфейсную часть приложения, и системные (невизуальные) компоненты. Таким образом, в системе Delphi форма является компонентом, который служит контейнером для всех других компонентов. В принципе можно создать и безоконное приложение, однако большинство приложений все же имеет видимое на экране окно, содержащее интерфейсную часть приложения.

Приложение может иметь несколько форм одна из которых считается главной и при запуске программы отображается первой.

Палитра компонентов — это витрина той библиотеки компонентов — VCL или CLX, с которой вы работаете. При создании приложений для Windows вы работаете с VCL. Но если вы создаете кросс-платформенное приложение, выполнив команду File | New | CLX Application, то палитра компонентов начинает отображать страницы CLX.

Инспектор Объектов (Object Inspector) обеспечивает простой и удобный интерфейс для изменения свойств объектов Delphi и управления событиями, на которые

реагирует объект

Редактор кода представляет из себя текстовый редактор кода с автоматическим выделением элементов программы важных для программиста таких как например

полужирным — ключевые слова языка программирования, цветом — константы, курсивом — комментарии.

Кроме того, редактор "на лету" проверяет программу на наличие синтаксических ошибок и в случае обнаружения ошибки подчеркивает ее красной волнистой линией (сообщение об обнаруженной ошибке отображается в окне Structure).

1. Опишите понятие «Компонент». Опишите работу с компонентом, задание и изменение свойств.

Компонент - это класс, интерфейс которого определяется свойствами (properties), методами (methods) и событиями (events). В визуальной среде разработки свойства компонента можно изменять во время проектирования.  
Интерфейс - механизм взаимодействия между двумя устройствами.  
Например, пользовательский интерфейс сайта управляет взаимодействием пользователя и сайта. Интерфейс библиотеки модулей определяет правила, по которым модули из этой библиотеки подключаются в программе. Интерфейс компонента имеет две области действия: программный интерфейс - использование компонента программистом в своем коде, и пользовательский интерфейс - работу пользователя с компонентом на этапе выполнения.  
  
Технически компонентом Delphi считается любой класс, порожденный из класса TComponent внутри иерархии классов VCL (Visual Component Library - библиотека визуальных компонентов).

Каждый компонент, который Вы помещаете на форму, имеет свое отражение в окне Инспектора Объектов (Object Inspector). Как Вы помните, Object Inspector имеет две "странички" - "Properties" (Свойства) и "Events" (События). Создание программы в Delphi сводится к "нанесению" компонент на форму (которая, кстати, также является компонентом) и настройке взаимодействия между ними путем:

изменения значения свойств этих компонент

написания адекватных реакций на события.

Как Вы уже успели, наверное, заметить, свойство является важным атрибутом компонента. Для пользователя (программиста) свойство выглядит как простое поле какой-либо структуры, содержащее некоторое значение. Однако, в отличие от "просто" поля, любое изменение значения некоторого свойства любого компонента сразу же приводит к изменению визуального представления этого компонента, поскольку свойство инкапсулирует в себе методы (действия), связанные с чтением и записью этого поля (которые, в свою очередь, включают в себя необходимую перерисовку). Свойства служат двум главным целям. Во-первых, они определяют внешний вид формы или компонента. А во-вторых, свойства определяют поведение формы или компонента.

Существует несколько типов свойств, в зависимости от их "природы", т.е. внутреннего устройства.

Простые свойства - это те, значения которых являются числами или строками. Например, свойства Left и Top принимают целые значения, определяющие положение левого верхнего угла компонента или формы. Свойства Caption и Name представляют собой строки и определяют заголовок и имя компонента или формы.

Перечислимые свойства - это те, которые могут принимать значения из предопределенного набора (списка). Простейший пример - это свойство типа Boolean, которое может принимать значения True или False.

Вложенные свойства - это те, которые поддерживают вложенные значения (или объекты). Object Inspector изображает знак "+" слева от названия таких свойств. Имеется два вида таких свойств: множества и комбинированные значения. Object Inspector изображает множества в квадратных скобках. Если множество пусто, оно отображается как []. Установки для вложенных свойств вида "множество" обычно имеют значения типа Boolean. Наиболее распространенным примером такого свойства является свойство Style с вложенным множеством булевых значений. Комбинированные значения отображаются в Инспекторе Объектов как коллекция некоторых величин, каждый со своим типом данных (рис 1). Некоторые свойства, например, Font, для изменения своих значений имеют возможность вызвать диалоговое окно. Для этого достаточно щелкнуть маленькую кнопку с тремя точками в правой части строки Инспектора Объектов, показывающей данное свойство.

Delphi позволяет легко манипулировать свойствами компонент как в режиме проектирования (design time), так и в режиме выполнения программы (run time).

1. Опишите основные свойства объектов.

Язык delphi использует концепцию обьекто-ориентированного программирования

Это означает, что функциональность приложения определяется набором взаимосвязанных задач, каждая из которых становится самостоятельным объектом. У объекта есть свойства (т.е. характеристики, или атрибуты) и методы, определяющие его поведение. В основе объектно-ориентированного программирования (ООП) лежит понятие класса.

Объекты в программе всегда являются экземплярами того или иного класса (подобно переменным определенного типа).

Сущность полиморфизма заключается в том, что методы различных классов могут иметь одинаковы имена, но различное содержание. Это достигается переопределением родительского метода в классе-потомке. В результате родитель и потомок в одинаковых ситуациях ведут себя по-разному.

Инкапсуляция

Инкапсуляция —это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.

Наследование

Наследование —это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс —потомком, наследником или производным классом.

Области видимости — это возможности доступа к составным частям объекта. В Delphi поля и методы могут относиться к четырем группам: «общие» (public), «личные» (private), «защищенные» (protected) и «опубликованные» (published).

1. Поля, свойства и методы, находящиеся в секции public, не имеют ограничений на видимость. Они доступны из других функций и методов объектов, как в данном модуле, так и во всех прочих, ссылающихся на него.

2. Поля, свойства и методы, находящиеся в секции private, доступны только в методах класса и в функциях, содержащихся в том же модуле, что и описываемый класс. Такая директива позволяет скрыть детали внутренней реализации класса от всех. Элементы из секции private можно изменять, и это не будет сказываться на программах, работающих с объектами этого класса. Обратиться к ним можно, только переписав содержащий их модуль.

3. Поля, свойства и методы, находящиеся в секции protected, доступны только внутри классов, являющихся потомками данного, в том числе и в других модулях. Такие элементы особенно необходимы дня разработчиков новых компонентов — потомков уже существующих.

4. Область видимости published имеет особое значение для интерфейса визуального проектирования. В этой секции должны быть собраны те свойства объекта, которые будут видны не только во время исполнения приложения, но и из среды разработки. Все свойства компонентов, доступные через Инспектор объектов, являются их опубликованными свойствами. Во время выполнения такие свойства общедоступны, как и public.

В Delphi объекты могут быть только динамическими! Любая переменная объектного типа есть указатель, причем для доступа к данным, на которые ссылается указатель объекта не нужно применять символ

Конструктор — это специальный метод, заголовок которого начинается зарезервированным словом constructor. Функция конструктора заключается в выделении памяти под экземпляр класса (объект) и в установлении связи между созданным объектом и специальной информацией о классе. В теле конструктора можно расположить любые операторы, которые необходимо выполнить при создании объекта, например, присвоить полям начальные значения. В Delphi конструкторов у класса может быть несколько. Общепринято называть конструктор Create.

Классическое правило объектно-ориентированного программирования утверждает, что для обеспечения надежности нежелателен прямой доступ к полям объекта: чтение и обновление их содержимого должно производиться посредством вызова соответствующих методов. Это правило и называется инкапсуляцией. В Delphi пользователь объекта может быть полностью отгорожен от полей при помощи свойств.

1. Опишите основные методы объектов

Методы — это процедуры и функции, которые принадлежат объекту. С помощью своих методов объект выполняет возложенные на него обязанности. Теперь нам предстоит узнать о методах объектов немного больше.  
  
   Как вы уже знаете, методы описываются внутри объявления объекта и бывают нескольких типов.

static (статические) — это простые процедуры и функции. Если при описании метода вы ничего не указали, то используется именно этот тип. Для компилятора это самый простой тип метода, потому что в потомках такой метод не может быть изменен, и поэтому заранее можно узнать адрес этого метода в памяти.

virtual (виртуальные) — такие методы могут быть переопределены в потомках объекта. Например, если у вас есть объект гараж и метод ворота, то в его потомке этот метод может быть заменен улучшенной версией. Для определения адреса Delphi строит таблицу виртуальных методов, которая позволяет во время выполнения программы определить адрес метода. В такой таблице хранятся все методы текущего объекта и его предка.

dynamic (динамические) — эти методы схожи с виртуальными, но для определения адреса используется другой способ. Для каждого объекта строится таблица только из его методов. Каждому методу назначается уникальный индекс. В данном случае экономится память, потому что не надо хранить адреса методов предков, но для поиска любого из них тратится намного больше времени.

message (сообщения) — такие методы реагируют на события операционной системы. Для большинства сообщений ОС Windows в Delphi уже есть специальные обработчики событий, но если вам нужно, чтобы метод реагировал на определенное событие, которого нет у компонента, необходимо определить его вручную.

abstract (абстрактный) — такой метод будет только объявлен в объекте, а реализации у него не будет. Если в объекте есть хотя бы один такой метод, то он считается абстрактным. Такой объект нельзя использовать.  
   Реализацию метода должны сделать потомки, и именно с потомками можно работать. Таким образом, вы можете в каком-либо объекте зарезервировать имя метода для потомков, чтобы они реализовали в нем какое-то действие.

   Если вы объявили метод как virtual или dynamic, то можно переопределить их действия в наследниках.

1. Опишите понятие «События». Опишите источники событий.

Под понятием события в Delphi подразумевается способность объекта вызывать какое-либо действие при исполнении программы. Проще говоря, это вызов программной функции, при определенном действии пользователя программы.

Одно событие может иметь только один обработчик

**Источником** **события** может выступать периферийный модуль, внешнее воздействие (например, нажатие кнопки), функция, ну и тому подобные вещи. Как правило, **источников** **событий** несколько и они асинхронны по отношению к друг другу.

1. Опишите компоненты страницы Standard

Standard – включает стандартные компоненты, обеспечивающие некоторые функции интерфейса пользователя. Компоненты на этой странице представляют стандартные Windows-элементы (кнопки, списки, меню и т.п.)

|  |
| --- |
| TMainMenu позволяет вам поместить главное меню в окно. Такое меню присутствует в большинство программах. При помещении TMainMenu на проектируемую форму, оно выглядит в виде значка. Для того, чтобы добавить, изменить, удалить пункты меню используйте свойство этого компонента Items в Object Inspector. При этом вызывается окно построителя пунктов меню. |
| TPopupMenu позволяет создавать всплывающее меню для многих других компонентов. Это меню появляется при нажатии на правую кнопку мышки на том компоненте, к которому оно привязано. Изменить пункты меню можно аналогично главному меню (свойство Items). Чтобы к конкретному компоненту привязать это всплывающее меню, его необходимо указать в свойстве PopupMenu этого компонента. |
| TCaption это обычный текст. Его можно редактировать как во время разработки, так и во время работы приложения. Текст можно изменить в свойстве Caption, шрифт меняется в свойстве Font. |
| TEdit предназначен для ввода пользователем некоторых данных. Обычно им пользуются для ввода цифровых или текстовых данных, пароля (свойство PasswordChar установите маску отображаемых символов, обычно это звездочка). Доступ к тексту - свойство Text. Ограничение количества введенных символов - свойство MaxLength (0 - число символов не ограничено). |
| TMemo это многострочный компонент для ввода/вывода данных. Может содержать относительно большой текст. Пример его использования вы можете увидеть в стандартном блокноте Windows. Для доступа к тексту используйте свойство Lines. Оно представляет набор строк Strings. Свойство WordWrap указывает нужно или нет производить автоматический перенос текста на новую строку, если она выходит за границы компонента. Этот компонент не работает с текстом формата RTF. Для этого применяйте компонент TRichEdit на вкладке Win32. |
| TButton это обычная кнопка в вашем приложении. Текст на кнопке задается свойством Caption. |
| TCheckBox представляет собой текстовую строку с возможностью ее установки в три положения. Свойство Checked true (истинно, установлена галочка), false (ложно, галочка снята). Возможно ее выделить серым в свойстве State в cbGrayed. Текст устанавливается в свойстве Caption. Такие компоненты чаще всего устанавлявают в окна настройки параметров программы, окна поиска. |
| TRadioButton применяется в списках свойств, где нужно выбрать только одну опцию из нескольких. Текст - свойство Caption, состояние - свойство Checked. Если компонентов TRadioButton на управляющей панели (TPanel, TGroupBox, TForm) несколько, то если пользователь установит флаг на одном, то на остальных он автоматически снимается. Это принцип группировки. Если у вас несколько таких групп, которые используют разные тематики в работе, то устанавливайте их в отдельные панели. |
| TListBox предназначен для отображения списка строк. От компонента TMemo он отличается только тем, что невозможно непосредственно редактировать текст, пользователь может только выбрать одну или несколько строк в этом списке. Доступ к строкам Lines. |
| TComboBox напоминает компонент TListBox, но только нужный пункт можно выбрать из ниспадающего списка. Строки хранятся в свойстве Lines. Установленный пункт из списка - свойство ItemIndex. Пример его использования - список шрифтов в текстовых редакторах. |
| TScrollBar это полоса прокрутки. Используется для прокручивания невидимой части в компоненте. |
| TGroupBox используется для оформления внешнего вида приложения и для группировки установленных в форму компонентов по смыслу. Все компоненты в форме можно разделить по функциональным назначениям и разместить в таких компонентах. Удобен при назначении порядка перехода по компонентам с помощью кнопки Tab. Заголовок этого компонента изменяется свойством Caption. |
| TRadioGroup по внешнему виду похож на компонент TGroupBox. Его отличие лиш в том, что в нем располагаются список компонентов TRadioButton. Список содержится в свойстве Items. Очень удобно использовать этот компонент т.к. он автоматически позиционирует список переключателей после изменения из списка. Свойство ItemIndex показывает какой переключатель сейчас установлен (-1 нет, 0 первый, 1 второй ...). |
| TPanel по функциональному назначению похож на компонент TGroupBox. |
| TActionList может содержать список как стандартных действий компонентов (вырезать, вставить, упорядочить значки для MDI-окон), так и есть возможность создать собственные действия. |

1. Опишите компоненты страницы Additional

Additional – на этой закладке располагаются дополнительные интерфейсные компоненты (для красочного оформления приложения, усовершенствованные компоненты из предыдущей страницы).

|  |
| --- |
| TBitBtn это таже кнопка TButton, но на ней можно разместить рисунок. Картинка загружается в кнопку с помощью свойства Glyph. Этой кнопке можно задать стандартный вид, изменив свойство Kind. Если вы используете а приложении кнопку без рисунка, то используйте компонент TButton. |
| TSpeedButton является частью компонента TSpeedBar. Это кнопка, на которой обычно располагается только картинка без текста. Границы этой кнопки можно спрятать, изменив свойство Flat на true. В таком случае этот компонент будет выделен только в том случае, когда указатель мыши находится над ним. Пример - кнопки на палитре компонентов и кнопки быстрого запуска команд меню. |
| TMaskEdit похож на компонент TEdit, но позволяет вводить данные определенного формата. Например телефон 555-55-55. Этот формат указывается в свойстве EditMask. |
| TStringGrid представляет собой таблицу, в которую могут заносится текстовые данные. Доступ к данных возможен только во время работы приложения через свойство Cell. |
| TDrawGrid таблица с занесением данных любого формата (текста, рисунков и т.п.). Доступ к ячейкам через свойство CellRect только во время работы приложения. |
| TImage может содержать рисунок большинства широко распространенных форматов. Рисунок устанавливается в свойстве Picture. Установленный рисунок можно центрировать (Center), вписать в рамки компонента увеличив или уменьшив (Stretch), сделать прозрачным относительно нижней левой точки рисунка (Transparent). |
| TShape позволяет программисту размещать в форму окружность, квадрат, линию, загружать изображения. Этот компонент можно применять в простейших графических редакторах. |
| TBevel применяется только для украшения внешнего вида окон. Может принимать вид приподнятых (опущенных) панелей из свойства Style. Может стать линией, квадратом, пунктирным квадратом с помощью свойства Shape. |
| TScrollBox полезен, когда вся видимая часть размещенных в него компонентов недоступна для пользователя. В таком случае в этом компоненте автоматически появляются бегунки прокрутки. |
| TCheckListBox по внешнему виду похож на компонент TListBox, но в его строках Items есть переключатели, аналогичные TCheckBox. Доступ к которым можно получить только во время работы приложения через свойство Checked[N], где N - индекс строки. Если установлено true, значит галочка стоит, если false - наоборот. |
| TSplitter позволяет изменять границы компонентов, к которым он прилегает. Может быть как вертикальным (Свойство Align в alLeft или alRight), так и горизонтальным (Align alTop или alBottom). |
| TStaticText функционирует подобно TLabel. Применяется в основном для текстов в диалоговых окнах, для вывода данных. Кроме того его текст можно отображать в рамке через свойство BorderStyle. |
| TControlBar применяется для современного оформления меню, кнопок быстрого доступа. В нем панели можно передвигать мышкой. Пример вы можете увидеть в главном окне Delphi. Там главное меню, кнопки, панель палитры компонентов расположены на компоненте TControlBar. |
| TChart позволяет красиво, в графической форме представить некоторые табличные данные. Данные прямо из этого компонента можно печатать, выводить в виде графики в файл. |

1. Опишите компоненты страницы Win 32

Win32 – обеспечивают доступ к 32-битным элементам Windows.

|  |
| --- |
| TTabControl это страничный компонент подобный картотеке или записной книжке. Этот компонент обеспечивает выбор нужной страницы из массива имеющихся. Список страниц находится в свойстве Tabs. |
| TPageControl это еще один страничный компонент. В отличие от предыдущего все размещаемые на нем страницы есть отдельными компонентами. Для создания страниц на этом компоненте щелкайте по нему правой кнопкой мышки. Все визуальные компоненты (видимые во время работы приложения), помещенные на определенную страницу становятся недоступными при перелистывании на другую. Пример можно увидеть в стандартном поиске файлов Windows. |
| TImageList может содержать в себе список хранящихся вместе изображений. Доступ к ним осуществляется через индекс. Для изменения списка хранимых изображений дважды щелкайте по этому компоненту. Все изображения в нем должны иметь одинаковый размер. Используйте свойства Height и Width для задания ширины и высоты изображения. Эти изображения в последствии через индекс можно размещать на TMainMenu, TPopupMenu, TPageControl, TToolBar, TTreeView, TListView и т.д. Необходимо помнить, что все загружаемые в этот компонент изображения, как и в любом другом компоненте с изображением, линкуются в исполняемый EXE файл. |
| TRichEdit в отличие от компонента TMemo может содержать форматированный текст RTF. Это текст с разными шрифтами, цветом. Пример вы увидите в текстовых редакторах MSWord или WordPad. |
| TTrackBar представляет собой многопозиционный бегунок. Его полезно использовать для задания данных из допустимого диапазона. Минимальный и максимальный диапазон данный устанавливается в свойстве Min и Max, а текущую позицию можно узнать или установить с помощью свойства Position. Этот бегунок можно увидеть в регуляторе громкости. |
| TProgressBar это простой индикатор процесса работы. Его можно увидеть в момент копирования файлов, форматирования диска. Минимальные и максимальные значения хранятся в свойстве Min и Max. Текущая позиция Position. Величина шага перемещения указателя - свойство Step. Само перемещение указателя прогресса осуществляется командой StepIt. Например, необходимо скопировать семь файлов. Изначально программно или на этапе проектировки задаем свойства Min=0, Max=7, Step=1, а после копирования каждого из файлов применяем StepIt. |
| TUpDown размещается с целью задания величин с последующим пошаговым изменением. Сама величина не отображается. Свойства схожи с компонентом TTrackBar. В свойстве Increment задается величина шага изменения значения. Если необходимо отображать результат тут же, то воспользуйтесь компонентом TSpinEdit на странице Samples. |
| THotKey позволяет задавать или отображать на экран комбинацию клавиш. Комбинация содержится в свойстве HotKey. |
| TAnimate представляет собой замечательный компонент оживления ваших приложений. С его помощью вы можете вставить анимированный видеофрагмент в вашу форму, прокрутить стандартную Windows анимацию. Файл загружается с указанного свойства FileName, а стандартную Windows анимацию устанавливают в свойстве CommonAVI. Для начала проигрывания не забудьте установить Active в true. |
| TDateTimePicker простой календарь и часы. Он может показывать время или календарь в зависимости от свойства Kind. Время и дату можно задавать прямо с клавиатуры, а дату можно еще указать из ниспадаюшего календарика. |
| TMonthCalendar это обычный календарь для выбора даты. Попробуйте его растянуть - можно даже увидеть весь год. Свойство установленной даты - Date. |
| TTreeView может показывать древовидную структуру данных. Например каталоги. Пользуйтесь свойством Items для изменения дерева. В дереве могут присутствовать изображения. Тогда к нему нужно привязать массив изображений TImageList и указывать для каждого элемента дерева индекс рисунка. |
| TListView широко известный компонент, применяемый в проводнике Windows. Может содержать большие, мелкие значки, список. Свойство Items содержит список выводимых элементов. |
| THeaderControl используйте его, чтобы отобразить заголовок колонок текста или чисел. |
| TStatusBar является нижней панелью, которая отображает информацию, как работает приложение. Практически повсюду применяется. В текстовых редакторах на такой панели можно узнать о состоянии кнопок Caps, Num, Scroll, узнать изменен или нет документ, на какой странице находится указатель и т.п. |
| TToolBar это хранилище кнопок, которые обычно размещаются в верхней части окна. Эта панель есть практически в любом приложении. Обычно на ней располагаются кнопки для быстрого доступа к пунктам главного меню. |
| TCoolBar представляет собой некий контейнер для компонентов. Там их легко позиционировать, перемещать. Такой элемент расположен в Internet Explorer. |
| TPageScroller cодержит в себе другие объекты, которые могут выходить за видимые пределы. Для этого пользователь использует стрелки и перемещает компонент в нужную сторону, тем самым освобождая невидимую часть. Пример вы можете увидеть на палитре компонентов на тех страницах, где видны не все значки компонентов. |

1. Опишите компоненты страницы System

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| System – набор компонентов для доступа к системным ресурсам, таким как OLE, DDE.   |  | | --- | | TTimer создает событие OnTimer, интервал которых задан с свойстве Interval. Этот интервал задается в миллисекундах и работает относительно точно. Тем не менее не рекомендуется его применение при точных вычислениях. С помощью свойства Enabled счетчик можно запустить (true) или остановить (false). | | TPaintBox компонент для рисования. Доступ к графике через Canvas. | | TMediaPlayer работает с изображением и звуком. Использует стандартный драйвер мультимедиа. Имеет вид готовой к работе панели управления устройством управления (кнопки Play, Stop и т.п.). | | TOLEContainer это компонент, содержащий OLE-объекты. Привязанные объекты во время проектировки приложения линкуются в исполняемый EXE файл. | | TDDEClientConv представляет собой связь с DDE сервером. Посылает данные или макрокоманды на сервер DDE после того, как связь была установлена. Это стандартные методы обмена данными между приложениями. Пример вы можете найти в стандартной поставке примеров Delphi. | | TDDEClientItem определяет обмен данными между DDE приложениями. Он содержит в себе данные, которые передаются. Используйте этот компонент вместе с TDDEClientConv, чтобы позволить приложению действовать как клиент DDE, и иметь возможность передать данные. | | TDDEServerConv представляет собой связь с DDE клиентом. | | TDDEServerItem представляет собой обмениваемые данные в DDE приложении. | |

1. Опишите компоненты страниц Dialog и Samples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dialogs – содержит стандартные диалоговые элементы Windows (окна открытия, сохранения файлов, окна выбора шрифта, принтера и т.п.)   |  | | --- | | TOpenDialog это диалог открытия файла. Этот и все другие диалоги активируются через функцию Execute. Если возвращаемое значение равно true, значит пользователь выбрал файл (принтер, шрифт и т.п.) и нажал на кнопку Ok в этом диалоговом окне. Тогда имя открываемого файла содержится в свойстве FileName. Обычно процедура вызова подобного диалогового окна выглядит следующим образом:  if OpenDialog.Execute then Memo.Lines.LoadFromFile(OpenDialog.FileName);  Здесь показан пример загрузки текстового файла в компонент Memo (страница Standart), выбранного с помощью диалога открытия файла OpenDialog.  Текст заголовка диалога в свойстве Title. Фильтр, который будет отфильтровывать файлы по расширению указывается в свойстве Filter. Его целесообразно установить сразу во время разработки проекта, поскольку при этом открывается удобная для занесения значений таблица.  Свойство InitialDir указывает программе каталог по умолчанию, который будет открыт в диалоговом окне. Если это свойство не задано, то берется каталог, из которого запущена программа. | | TSaveDialog - диалог сохранения файла. Свойства и работа подобны диалогу TOpenDialog. | | TOpenPictureDialog - диалог открытия рисунка. В отличие от TOpenDialog имеет внутренний предварительный просмотр рисунка в открываемом файле. Кроме того уже имеет некоторые заданные свойства фильтра Filter, который содержит все распознаваемые в этом диалоге форматы графики. | | TSavePictureDialog - диалог сохранения рисунка. | | TFontDialog - вызывает стандартный диалог выбора шрифта. Свойство Font содержит как изначальный (выбранный по умолчанию) шрифт, так и измененный, после вызова диалога.  FontDialog1.Font := Edit1.Font; if FontDialog.Execute then Edit.Font := FontDialog.Font;  Здесь показан пример изменения шрифта в компоненте Edit. Причем во время вызова диалога FontDialog в нем изначально будет установлен шрифт, заданный в компоненте Edit.  Свойство Device указывает типы отображаемых шрифтов. fdBoth - все, fdPrinter - принтерные, fdScreen - экранные.  MaxFontSize MinFomtSize устанавливает границы минимального и максимального размера шрифта. Цифра 0 - размер ограничивается только возможностями данного шрифта. | | TColorDialog - диалог выбора цвета. Цвет по умолчанию и измененное хранится в свойстве Color. | | TPrintDialog - диалог выбора принтера, выбора количества копий, страниц, свойства печати и т.п. | | TPrinterSetupDialog - диалог выбора принтера. | | TFindDialog - диалог поиска фрагмента текста в документе. | | TReplaceDialog - диалог замены текста в документе. | |

Обычно на странице Samples регистрируются свободно распространяемые компоненты, находящиеся в стадии разработки или тестирования. На данной странице представлены 12 образцов компонентов, семь из которых входят в стандартную поставку Delphi.

Компоненты, находящиеся на странице Samples



TGauge – индикатор процесса.

TColorGrid – таблица цветов.

TSpinButton – кнопки для дискретного увеличения и уменьшения значений в поле ввода.

TSpinEdit – поле с кнопками пошагового изменения значения ввода.

TDirectoryOutline – структура каталогов текущего диска.

TCalendar – табличный календарь.

TIBEventAlerter – компонент обработки сообщений сервера InterBase.

1. Перечислите компоненты меню. Опишите MainMenu.

Компонент **Delphi MainMenu** предназначен для добавления к программе главного меню, элемента, без которого не обходится ни одно из приложений для Windows.

1. Перечислите компоненты меню. Опишите PopupMenu.

Контекстное меню (**PopupMenu**), вызываемое в приложении Windows по щелчку правой кнопкой мыши, является стандартной и удобной возможностью многих программ. **Delphi компонент PopupMenu** предназначен для создания таких **контекстных меню**. Контекстное меню из-за способа своего появления называют ещё **всплывающим меню**.  
  
 **Компонент PopupMenu** почти не отличается от компонента MainMenu и также является невизуальным. Но в отличие от Главного меню, которое одно на Форме, всплывающие меню могут быть у каждого объекта, размещённого на Форме. Поэтому нужно разместить на Форме и запрограммировать столько **компонентов PopupMenu**, сколько контекстных меню предполагается использовать в программе. Кроме того, контекстное меню не имеет нескольких пунктов верхнего уровня, так как все его пункты располагаются в одном вертикальном столбце

1. Перечислите группы радиокнопок. Опишите компоненты RadioGroup и GroupBox.

Компонент RadioGroup применяется для формирования группы регулярно размещенных радиокнопок, из которых в любой момент времени может быть включена

только одна. Если по различным соображениям регулярное размещение радиокнопок нежелательно, то используются компоненты RadioButton, размещенные желательным образом в панели GroupBox

Radiogroup это панель, которая может содержать регулярно расположенные

столбцами и строками радиокнопки. Надпись в левом верхнем углу панели (см.

рис. 3.35) определяется свойством Caption. А надписи кнопок и их количество определяются свойством Items, имеющим тип TStrings

Компонент GroupBox представляет собой контейнер для размещения элементов управления или других компонентов-контейнеров, обладающий рамкой и надписью. Работа с компонентом. Компонент предназначен для объединения элементов управления, таких как RadioButton , CheckBox и т.д. В отличии от компонента Panel , компонент GroupBox не имеет широких возможностей задания различных стилей оформления, и используется для выделения на форме группы функционально объединенных компонентов.

1. Опишите графические возможности Delphi

Способы формирования изображений   
Существует два способа вывода графических объектов: - вывод заранее подготовленных изображений, не требует программирования и подходит для вывода статичных изображений. Он основан на использовании компонентов Image, Shape и TBevel - программный способ требует программирования, предоставляет возможности для динамического создания изображений и их анимации. Он основан на использовании программного свойства Canvas, присутствующего в форме и управляющих элементах

1. Основные компоненты для вывода изображений

В стандартную библиотеку визуальных компонент Delphi входит несколько объектов, с помощью которых можно создавать графические примитивы. Это - TImage (TDBImage), TShape, . Image позволяет поместить графическое изображение в любое место на форме. Shape позволяет поместить простейшие графические объекты на форме типа круг, квадрат и т. п. Разместить на форме данные компоненты используюя вкладку Additional палитры компонент. При проектировании следует помнить, что изображение, помещенное на форму во время дизайна, включается в файл. DPR и затем прикомпилируется к EXE файлу. Поэтому такой EXE файл может получиться достаточно большой.  
Создание и отображение статичных изображений   
Наиболее просто вывести иллюстрацию, которая находится в файле с помощью компонента TImage.

Отображение геометрических фигур.   
Чтобы показать простую геометрическую фигуру (прямоугольник, эллипс и т. д) можно воспользоваться компонентом Shape (Страница Additional). Вид геометрической фигуры задается свойством Shape, заполнение ее внутреннего пространства – составным свойством Brush (Bitmap, Color, Style), а обрисовка внешних границ – составным свойством Pen. Эти свойства имеют несколько значений. Перебирая значения свойства Shape можно получить все виды геометрических фигур, поддерживаемые компонентом Shape.  
  
Метод заполнения внутреннего пространства фигуры определяется значением свойства Brush. Style. Если оно равно bs.Solid, то фигура заливается цветом, заданным в свойстве Brush.Color. Если стиль кисти равен bs.Clear, то фигура рисуется прозрачной. Остальные значения стиля задают всевозможные варианты штриховки внутренней области. Цвет штриховки линий определяется значением свойства Brush.Color. Цвет граничных линий содержится в свойстве Pen.Color, а ее толщина – в свойстве Pen.Width.  
  
Многие компоненты в Delphi имеют свойство Canvas (канва, холст), представляющее собой область компонента, на которой можно рисовать или отображать готовые изображения. Это свойство имеют формы, графические компоненты Image,

PaintBox, BitMap и многие другие. Канва содержит свойства и методы, существенно упрощающие графику Delphi. Все сложные взаимодействия с системой спрятаны для пользователя, так что рисовать в Delphi может человек, совершенно не искушенный в машинной графике.

Каждая точка канвы имеет координаты X и Y. Система координат канвы, как

и везде в Delphi, имеет началом левый верхний угол канвы. Координата X возрастает при перемещении слева направо, а координата Y — при перемещении сверху

вниз.  
  
Brush -кисть, является объектом со своим набором свойств:   
Bitmap - картинка размером строго 8 x 8, используется для заполнения (заливки) области на экране.   
Color - цвет заливки.   
Style – предопределенный стиль заливки;   
Copy.Mode - свойство определяет, каким образом будет происходить копирование (метод Copy. Rect) изображения из другого места: один к одному, с инверсией изображения и др.   
Font - шрифт, которым выводится текст (метод Text.Out).   
Pen - карандаш, определяет вид линий является объектом с набором свойств:   
Mode - режим вывода: простая линия, с инвертированием, с выполнением исключающего и др.   
Style - стиль вывода: линия, пунктир и др.   
Width - толщина линии в точках   
Pen.Pos - текущая позиция карандаша, карандаш рекомендуется перемещать с помощью метода Move.To, а не прямой установкой данного свойства.   
Pixels - двухмерный массив элементов изображения (pixel), с его помощью Вы получаете доступ к каждой отдельной точке изображения

Методы для рисования простейшей графики - Arc, Chord, Line. To, Moveto, ellipse Polygon, Poly. Line, Rectangle, Round. Rect.   
При прорисовке линий в этих методах используются карандаш (Pen) канвы, а для заполнения внутренних областей - кисть (Brush). Методы для вывода картинок на канву - Draw и Stretch.   
Draw.   
В качестве параметров указываются прямоугольник и графический объект для вывода (это может быть TBitmap, TIcon или TMetafile).   
Stretch.Draw отличается тем, что растягивает или сжимает картинку так, чтобы она заполнила весь указанный прямоугольник (см. пример к данному уроку). Методы для вывода текста - Text.Out и Text.Rect. При выводе текста используется шрифт (Font) канвы. При использовании Text.Rect текст выводится только внутри указанного прямоугольника. Длину и высоту текста можно узнать с помощью функций Text. Width и Text. Height.

1. Опишите понятие «Канва» и ее возможности.

Многие компоненты в Delphi имеют свойство Canvas (канва, холст), представляющее собой область компонента, на которой можно рисовать или отображать готовые изображения. Это свойство имеют формы, графические компоненты Image,

PaintBox, BitMap и многие другие. Канва содержит свойства и методы, существенно упрощающие графику Delphi. Все сложные взаимодействия с системой спрятаны для пользователя, так что рисовать в Delphi может человек, совершенно не искушенный в машинной графике.

Каждая точка канвы имеет координаты X и Y. Система координат канвы, как

и везде в Delphi, имеет началом левый верхний угол канвы. Координата X возрастает при перемещении слева направо, а координата Y — при перемещении сверху

вниз.  
  
Brush -кисть, является объектом со своим набором свойств:   
Bitmap - картинка размером строго 8 x 8, используется для заполнения (заливки) области на экране.   
Color - цвет заливки.   
Style – предопределенный стиль заливки;   
Copy.Mode - свойство определяет, каким образом будет происходить копирование (метод Copy. Rect) изображения из другого места: один к одному, с инверсией изображения и др.   
Font - шрифт, которым выводится текст (метод Text.Out).   
Pen - карандаш, определяет вид линий является объектом с набором свойств:   
Mode - режим вывода: простая линия, с инвертированием, с выполнением исключающего и др. Style - стиль вывода: линия, пунктир и др.   
Width - толщина линии в точках   
Pen.Pos - текущая позиция карандаша, карандаш рекомендуется перемещать с помощью метода Move.To, а не прямой установкой данного свойства.   
Pixels - двухмерный массив элементов изображения (pixel), с его помощью Вы получаете доступ к каждой отдельной точке изображения

1. Опишите мультимедийные возможности Delphi. Опишите назначение и свойства компонентов Animate и MediaPlayer

Большинство современных программ, работающих в среде Windows, являются мультимедийными. Такие программы обеспечивают просмотр видеороликов и мультипликации, воспроизведение музыки, речи, звуковых эффектов. Типичными примерами мультимедийных программ являются игры и обучающие программы.

Delphi предоставляет в распоряжение программиста два компонента, которые позволяют разрабатывать мультимедийные программы:

Animate — Используется для воспроизведения немых

клипов AVI, подобных используемым в Windows изображениям копирования файлов и т.п.

MediaPlayer — позволяет решать более сложные задачи, например, воспроизводить видеоролики, звук, сопровождаемую звуком анимацию.

Таблица. Свойства компонента Animate

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Свойство | Определяет |  |
|  | Name | Имя компонента. Используется для доступа к свойствам компонента и управлением его поведением |  |
|  | FileName | Имя AVI-файла в котором находится анимация, отображаемая при помощи компонента |  |
|  | StartFrame | Номер кадра, с которого начинается отображение анимации |  |
|  | stopFrame | Номер кадра, на котором заканчивается отображение анимации |  |
|  | Activate | Признак активизации процесса отображения кадров анимации |  |
|  | Color | Цвет фона компонента (цвет "экрана"), на котором воспроизводится анимация |  |
|  | Transparent | Режим использования "прозрачного" цвета при отображении анимации |  |
|  | Repetitions | Количество повторов отображения анимации |  |

Таблица. Кнопки компонента MediaPlayer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | Кнопка | Обозначение | Действие |  |
|  | Воспроизведение | btPlay | Воспроизведение звука или видео |  |
|  | Пауза | btPause | Приостановка воспроизведения |  |
|  | Стоп | btStop | Остановка воспроизведения |  |
|  | Следующий | btNext | Переход к следующему кадру |  |
|  | Предыдущий | btPrev | Переход к предыдущему кадру |  |
|  | Шаг | btStep | Переход к следующему звуковому фрагменту, например, к следующей песне на CD |  |
|  | Назад | btBack | Переход к предыдущему звуковому фрагменту, например, к предыдущей песне на CD |  |
|  | Запись | btRecord | Запись |  |
|  | Открыть/Закрыть | btEject | Открытие или закрытие CD-дисковода компьютера |  |
|  |  |  |  |  |

Таблица. Свойства компонента MediaPiayer

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Свойство | Описание |  |
|  | Name DeviceType  FileName AutoOpen Display  VisibleButtons | Имя компонента. Используется для доступа к свойствам компонента и управлением работой плеера  Тип устройства. Определяет конкретное устройство, которое представляет собой компонент MediaPiayer. Тип устройства задается именованной константой: dtAutoSelect — тип устройства определяется автоматически; dtVaweAudio — проигрыватель звука; dtAVivideo — видеопроигрыватель; dtCDAudio — CD-проигрыватель  Имя файла, в котором находится воспроизводимый звуковой фрагмент или видеоролик  Признак автоматического открытия сразу после запуска программы, файла видеоролика или звукового фрагмента  Определяет компонент, на поверхности которого воспроизводится видеоролик (обычно в качестве экрана для отображения видео используют компонент Panel)  Составное свойство. Определяет видимые кнопки компонента. Позволяет сделать невидимыми некоторые кнопки |  |

1. Опишите мультимедийные возможности Delphi

Большинство современных программ, работающих в среде Windows, являются мультимедийными. Такие программы обеспечивают просмотр видеороликов и мультипликации, воспроизведение музыки, речи, звуковых эффектов. Типичными примерами мультимедийных программ являются игры и обучающие программы.

Delphi предоставляет в распоряжение программиста два компонента, которые позволяют разрабатывать мультимедийные программы:

Animate — Используется для воспроизведения немых

клипов AVI, подобных используемым в Windows изображениям копирования файлов и т.п.

MediaPlayer — позволяет решать более сложные задачи, например, воспроизводить видеоролики, звук, сопровождаемую звуком анимацию.

Компонент Animate, значок которого находится на вкладке Win32 , позволяет воспроизводить простую анимацию, кадры которой находятся в AVl-файле.

Компонент MediaPlayer, значок которого находится на вкладке System, позволяет воспроизводить видеоролики, звук и сопровождаемую звуком анимацию.

1. Опишите создание анимации и мультипликации в Delphi

В Delphi алгоритм создания анимации состоит из нескольких шагов

Прорисовка обьекта

Отсчет интервала времени

Очистка экрана

Прорисовка изображения заново с изменением положения или формы

1. Опишете процесс создания таблиц и баз данных средствами Delphi

Прежде, чем начать строить приложения, работающие с базами данных, надо

иметь сами базы данных. Delphi поставляется с примерами, имеющими немало баз

данных, которыми можно воспользоваться для обучения. Базы данных Pers и Dep,

фрагменты которых были приведены выше в табл. 9.1 и 9.3 и которые будут использоваться в примерах данной и последующих глав книги, вы можете найти на

диске, приложенном к этой книге. Но можно и самим создать необходимые базы

данных. Причем не обязательно для этого использовать стандартные СУБД. Вместе с BDE и Delphi поставляется программа Database Desktop (файл DBD32.EXE

для 32-разрядных Delphi), которая позволяет создавать таблицы баз данных некоторых СУБД, задавать и изменять их структуру.

Обычно вызов Database Desktop включен в главное меню Delphi в раздел Tools.

Если это не сделано, то полезно включить его туда с помощью команды

Tools | Configure Tools. Вызовите Database Desktop.

Давайте создадим с помощью Database Desktop таблицу базу данных СУБД

Paradox 7. В Paradox 7 база данных — это каталог, в котором лежат таблицы -

файлы с расширением db. Поэтому прежде надо создать соответствующий каталог

с помощью любой программы Windows, например, с помощью «Проводника\*. Далее

выполните команду Database Desktop File | New. Вам откроется меню с разделами:

QBE Query

SQL File

Тable

Визуальный построитель запросов и запись этих запросов в файл.

Создание запроса на SQL и запись его в файл.

Создание новой таблицы.

Выберите Table. Откроется небольшое диалоговое окно . В нем из выпадающего списка вы можете выбрать СУБД, для которой хотите создать таблицу. Можете посмотреть в нем, таблицы каких СУБД могут создаваться с помощью Database Desktop. Выберите Paradox 7. Вы увидите окно Database Desktop.

В этом окне вы можете задать структуру таблицы (поля и их типы), создать вторичные индексы, ввести диапазоны допустимых значений полей, значения по умолчанию и ввести много иной полезной информации о создаваемой таблице.

1. Опишите компоненты для работы с базами данных. Опишите технологию BDE

Для работы с базами данных используются три группы компонентов:

Множества данных (data sets)

Визуальные компоненты баз данных (data-aware controls)

Источники данных (data sources).

Компонент **Table** является самым фундаментальным множеством данных в Delphi. Он обеспечивает доступ к таблице практически любой СУБД. С помощью данного компонента можно организовать произвольный доступ к любой записи или их подмножеству, поиск записей по заданному критерию.

Компонент **Query** является самым гибким множеством данных в Delphi. Он инкапсулирует в себе запрос к базе данных на языке SQL, результат действия которого есть логическая таблица, формируемая их строк и столбцов всех физических таблиц базы данных. В сравнении с компонентом **Table** компонент **Query** обладает тем преимуществом, что он может работать сразу с несколькими таблицами, предоставляя доступ ко всем реляционно связанным данным. Более того, данный компонент позволяет создавать “гетерогенные множества”, объединяя на серверах таблицы с разнотипными СУБД.

Визуальные компоненты баз данных – это управляющие элементы пользовательского интерфейса для просмотра и редактирования данных. Большинство из них дублируют известные управляющие элементы диалоговых окон. Визуальные компоненты баз данных отличаются от обычных управляющих элементов тем, что сами знают о том, как брать данные из полей таблицы и как их туда помещать. Для этого они связываются с соответствующими источниками данных с помощью своих свойств.

Источники данных – это невизуальные компоненты **DataSource**, выполняющие роль трубопроводов между множествами данных и визуальными компонентами данных.

BDE (сокр. от англ. Borland Database Engine — «движок баз данных Borland») — 32-битный движок баз данных под Microsoft Windows для доступа к базам данных из Borland Delphi, C++ Builder, IntraBuilder, Paradox for Windows и Visual dBASE for Windows. В RAD Studio не поддерживается начиная с версии XE 7. Однако, при необходимости, может быть установлен отдельно

1. Дайте определение понятию «Динамические структуры данных». Опишите принцип организации динамических структур при помощи указателей.

Динамические структуры данных – это структуры данных, память под которые выделяется и освобождается по мере необходимости.

Указатель — это всего лишь специальная переменная. В отличие от обычных переменных, которые хранят значение определенного типа, указатель хранит ад­рес ячейки памяти. Название "указатель" обусловлено тем, что указатель не со­держит конкретного значения, а указывает на ячейку памяти, в которой хранится 1 нужное значение.

В Delphi существует два типа указателей: типизированные и нетипизированные (общие). Типизированные указатели можно использовать только с перемен­ными конкретного типа, в то время как нетипизированные указатели могут указы­вать на любые данные.

1. Дайте определение понятию «Стек». Опишите операции со стеком.

Стек представляет собой динамическую структуру данных, основанную на линейном однонаправленном списке. У стека для взаимодействия доступен только один конец структуры ─ вершина стека (рисунок 4). И включение нового элемента в стек и выборка последнего ранее включенного идет через вершину стека. Таким образом, на обслуживание попадает первым элемент, поступивший последним. Говорят, что стек ─ это структура с дисциплиной обслуживания LIFO(Last In, First Out) ─ «последним пришёл, первым ушёл».

Основные операции над стеками:

создание стека;

включение нового элемента;

проверка стека на пустоту;

извлечение элемента

1. Дайте определение понятию «Очередь». Опишите операции с очередью.

**Очередь** — структура данных, у которой дисциплина элементов организована по принципу, который называется «первый вошел — первый вышел» или FIFO(First In — First Out).

Основные операции над очередью:

создание очереди;

включение нового элемента;

проверка очереди на пустоту;

извлечение элемента.

1. Дайте определение понятию «Линейный однонаправленный список». Опишите операции с линейным однонаправленным списком.

Линейный список –это динамическая структура данных. У данных с динамической структурой с течением времени изменяется сама структура, а не только количество элементов, как у файлов или последовательностей. Линейный список характеризуется тем, что каждый его элемент связан с предшествующим (рисунок 1); известно, какой элемент находится в начале списка, какой в конце, а также, какой элемент стоит перед текущим; переходить от текущего элемента к следующему можно только с помощью указанных связей между соседними элементами

Основные операции над линейными однонаправленными списками:

создание списка;

включение нового элемента;

уничтожение списка;

исключение элемента;

полный перебор элементов списка;

поиск элемента;

упорядочение (сортировка) списка.

1. Дайте определение понятию «Дерево». Опишите обходы деревьев: прямой обратный, симметричный.

**Дерево** – это совокупность элементов, называемых узлами (один из которых определен как корень), и отношений («родительских»), образующих иерархическую структуру узлов. Вообще говоря, древовидная структура задает для элементов дерева (узлов) отношение «ветвления», которое во многом напоминает строение обычного дерева.

Типы обходов дерева

 прямой обход- сначала посещается корень, затем узлы поддерева

 Симметричный обход- сначала посещаются все узлы поддерева t1, затем корень n, затем последовательно в симметричном порядке все узлы поддеревьев t1,…,tk

 Обход в обратном порядке- сначала посещаются в обратном порядке все узлы поддерева t1, затем t2и т.д., последним посещается корень n.

1. Дайте определение понятию «Бинарное дерево». Опишите особенности обходов.

Бинарное дерево - это иерархическая динамическая структура данных, состоящая из узлов и соединяющих их дуг. В каждый узел, кроме одного, ведет ровно одна дуга.

1. Дайте определение понятию «Бинарное дерево». Опишите реализацию бинарных деревьев.

Бинарное дерево - это иерархическая динамическая структура данных, состоящая из узлов и соединяющих их дуг. В каждый узел, кроме одного, ведет ровно одна дуга.

1. Дайте определение понятию «Хеширование». Опишите функции хеширования

Хеширование (иногда хэширование, англ. hashing) - преобразование входного массива данных произвольной длины в выходную строку фиксированной длины.

Перечислим и кратко охарактеризуем наиболее часто используемые функции хэширования

**SHA** - Secure Hash Algorithm (1992)

160-разрядный хэш-код (дайджест). НЕ устойчив к коллизиям.

512-битовые блоки.

**SHA-1**  - Secure Hash Algorithm 1 (1995)

Модификация SHA. Исправлены недостатки. Решает проблему коллизий.

**SHA-2** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Secure Hash Algorithm Version 2* — безопасный алгоритм хеширования, версия 2) – семейство однонаправленных [хеш-функций](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F), включающее в себя алгоритмы *SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256* и *SHA-512/224*.

SHA-3 является важным криптографическим алгоритмом для обеспечения информационной безопасности, а также целостности данных в цифровых операциях. Недавние безопасные хеш-алгоритмы, включая MD5, RIPEMD, SHA-0, SHA-1 и SHA-2 устарели и были признаны восприимчивыми к атакам различного рода.

SHA-3 (Keccak) – алгоритм хеширования переменной разрядности, разработанный группой во главе с Йоаном Дайменом в 2012 году. 5 августа 2015 года алгоритм утверждён и опубликован в качестве стандарта FIPS 202. Keccak был выбран в качестве официального алгоритма для SHA-3 в 2012 году. Keccak основан на конструкции Sponge (Губка), которая является новым способом проектирования хеш-функций, помимо стратегии итерационного метода Меркла — Дамгора, реализуемой в MD(x).

**MAC -** Message Authentication Code - код аутентификации (проверки подлинности) сообщения.

Это зависящая от ключа однонаправленная хэш-функция. Простейшим способом преобразования хэш-функции в MAC является шифрование хэш-значения симметричным алгоритмом.

**HMAC**

Один из вариантов добавления секретного ключа в уже существующий алгоритм хэширования. Функция хэширования в этом алгоритме интерпретируется как «черный ящик», то есть функция хэширования реализована как отдельный модуль и ее можно менять.

Алгоритм HMAC [представлен в документе RFC 2104] принят как обязательный в протоколе IPSec и используется в ряде других протоколов Internet (TLS, SET и другие)

Широко используются на практике также функции, разработанные Роном Ривестом:

**MD2 -** Message Digest #2

Низкоскоростной, но очень надежный алгоритм, создающий 128-разрядные дайджесты данных любого объема.

**MD4** - Message Digest #4 (1990)

Более скоростной, но менее надежный алгоритм, создающий 128-разрядные дайджесты данных любого объема. 512-битовые блоки. Есть дефекты.

**MD5** - Message Digest #5 (1992)

Версия MD4 с повышенной надежностью, преимущества также и в скорости. 128-разрядные дайджесты данных любого объема.

Неустойчив к коллизиям!

Обратим внимание, что алгоритмы SHA надежнее алгоритмов MDx, так как вырабатывают более длинный хэш-код (160 бит против 128 бит), что снижает вероятность того, что разные входные последовательности будут преобразованы в одно значение хэш-кода.

1. Опишите принципы организации хеш-таблиц, алгоритмы выполнения операций с ними.

**Хеш-таблица** – это структура данных, реализующая интерфейс ассоциативного массива, то есть она позволяет хранить пары вида "ключ- значение" и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу. Хеш-таблица является массивом, формируемым в определенном порядке хеш-функцией.

Принято считать, что хорошей, с точки зрения практического применения, является такая хеш-функция, которая удовлетворяет следующим условиям:

функция должна быть простой с вычислительной точки зрения;

функция должна распределять ключи в хеш-таблице наиболее равномерно;

функция не должна отображать какую-либо связь между значениями ключей в связь между значениями адресов;

функция должна минимизировать число коллизий – то есть ситуаций, когда разным ключам соответствует одно значение хеш-функции (ключи в этом случае называются *синонимами* ).

Хеш-таблицы должны соответствовать следующим *свойствам*.

Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Получающееся хеш-значение является индексом в исходном массиве.

Количество хранимых элементов массива, деленное на число возможных значений хеш-функции, называется *коэффициентом заполнения хеш-таблицы* ( *load factor* ) и является важным параметром, от которого зависит среднее время выполнения операций.

Операции поиска, вставки и удаления должны выполняться в среднем за время O(1). Однако при такой оценке не учитываются возможные аппаратные затраты на перестройку индекса хеш-таблицы, связанную с увеличением значения размера массива и добавлением в хеш-таблицу новой пары.

Механизм разрешения коллизий является важной составляющей любой хеш-таблицы.

1. Опишите основные понятия комбинаторики. Опишите алгоритмы генерации перестановок, сочетаний, размещений.

**Комбинаторика** - раздел олимпиадного **программирования** (а так же и раздел математики), где ставится вопрос о подсчете числа вариантов выбора элементов из некоторого, как правило, конечного множества согласно заданным правилам.

*Перестановкой* из N элементов называется упорядоченный набор из N различных чисел от 1 до N. Количество различных перестановок порядка N равно N! = 1\*2\*3 ... \* (N-1) \* N. Заметим, что 0!=1. Для факториала справедлива следующая рекурентная запись: N! = (N-1)!\*N

Под *числом размещений* понимают количество вариантов, которыми можно записать в ряд подпоследовательность из K элементов некоторой перестановки из N элементов. При этом последовательности из одинаковых элементов, но с различным их порядком следования считаются различными. Количество таких комбинаций расчитывается по формуле: ANK = N!/(N-K)!.

Под *числом сочетаний* понимают количество вариантов, которыми можно выбрать K элементов из некоторого множества, состоящего из N элементов. При этом последовательности из одинаковых элементов, но с различным их порядком следования считаются равными. Количество таких комбинаций расчитывается по формуле: CNK = ANK/K! = N!/(K!\*(N-K)!).

1. Дайте определение понятию «Граф». Перечислите виды графов. Опишите способы представления ориентированных и неориентированных графов.

**Граф** — это конечная совокупность вершин, некоторые из которых соединены ребрами, т.е. это совокупность точек, называемых вершинами, и линий, соединяющих некоторые из вершин, называемых ребрами или дугами в зависимости от вида **графа**.

1. Опишите метод нахождения каркаса графа минимального веса по алгоритму Р. Прима

**Алгоритм** **Прима** (Prim). Будем строить **каркас** **графа** следующим образом. Пометим все вершины **графа**, кроме одной (произвольной), как неиспользованные. Пока есть неиспользованные вершины, выбираем ребро наименьшего **веса**, соединяющее использованную вершину с неиспользованной, и добавляем его в **каркас**, делая эту неиспользованную вершину использованной.

1. Опишите метод нахождения каркаса графа минимального веса по алгоритму Дж. Краскала

**Алгоритм** **Краскала** - это **алгоритм** **минимального** остовного дерева, что принимает **граф** в качестве входных данных и **находит** подмножество ребер этого **графа**, который формирует дерево, включающее в себя каждую вершину, а также имеет **минимальную** сумму **весов** среди всех деревьев, которые могут быть сформированы из **графа**.

1. Опишите алгоритм Дейкстры для нахождения кратчайшего пути в графе

**Алгори́тм Де́йкстры** (англ. *Dijkstra’s algorithm*) — алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях, например, его используют протоколы маршрутизации OSPF и IS-IS.

1. Объясните понятия: поток, поток максимально величины, разрез, минимальный разрез. Опишите алгоритм Форда-Фалкерсона нахождения максимального потока

**Поток** — это по сути последовательность инструкций, которые выполняются параллельно с другими **потоками**. Каждая программа создает по меньшей мере один **поток**: основной, который запускает функцию main().

1. Объясните сущность понятия динамического программирования. Опишите реализацию методов динамического программирования

Основным принципом, положенным в основу метода динамического программирования, является принцип оптимальности, суть которого заключается в том, что каждое последующее решение строится оптимальным образом независимо от решений, полученных на всех предыдущих этапах, кроме последнего. Чтобы реализовать этот принцип, необходимо в исходной задаче определить: этапы решения (иначе — подзадачи, на которые она декомпозируется); управляемые переменные (или варианты решений) на каждом этапе; информацию для решения задачи на каждом этапе; рекуррентные вычислительные процедуры, связывающие соседние этапы. Проще говоря, в методе динамического программирования искусственно создаются условия для независимой оптимизации на отдельном этапе по результатам только предыдущего, причём с гарантией того, что полученное решение будет находиться в области допустимых решений. Последовательность этапов может устанавливаться произвольно, за исключением случаев порождения подзадач в установленном хронологическом порядке. Первое прохождение этапов может быть как прямым, так и обратным, а методы оптимизации на отдельных этапах могут отличаться друг от друга.

1. Дайте определение понятию «Файл». Перечислите типы файлов. Опишите процедуры и функции применимые для всех типов файлов.

Файл — это определенное количество информации (программа или данные), имеющее имя и хранящееся в долговременной (внешней) памяти.

Основные **типы** **файлов**: регулярные (обычные) **файлы** и директории (справочники, каталоги ). Обычные **файлы** содержат пользовательскую информацию. Директории - системные **файлы**, поддерживающие структуру файловой системы

Процедуры работы с файлами

Assign(f, FileName)

связывает файловую переменную f с физическим файлом, полное имя которого задано в строке FileName. Установленная связь будет действовать до конца работы программы, или до тех пор, пока не будет сделано переназначение.

После связи файловой переменной с дисковым именем файла в программе нужно указать направление передачи данных (открыть файл). В зависимости от этого направления говорят о чтении из файла или записи в файл.

Reset(f)

открывает для чтения файл, с которым связана файловая переменная f. После успешного выполнения процедуры Reset файл готов к чтению из него первого элемента. Процедура завершается с сообщением об ошибке, если указанный файл не найден.

Если f - типизированный файл, то процедурой reset он открывается для чтения и записи одновременно.

Rewrite(f)

открывает для записи файл, с которым связана файловая переменная f. После успешного выполнения этой процедуры файл готов к записи в него первого элемента. Если указанный файл уже существовал, то все данные из него уничтожаются.

Close(f)

закрывает открытый до этого файл с файловой переменной f. Вызов процедуры Close необходим при завершении работы с файлом. Если по какой-то причине процедура Close не будет выполнена, файл все-же будет создан на внешнем устройстве, но содержимое последнего буфера в него не будет перенесено.

EOF(f): boolean

возвращает значение TRUE, когда при чтении достигнут конец файла. Это означает, что уже прочитан последний элемент в файле или файл после открытия оказался пуст.

Rename(f, NewName)

позволяет переименовать физический файл на диске, связанный с файловой переменной f. Переименование возможно после закрытия файла.

Erase(f)

уничтожает физический файл на диске, который был связан с файловой переменной f. Файл к моменту вызова процедуры Erase должен быть закрыт.

IOResult

возвращает целое число, соответствующее коду последней ошибки ввода - вывода. При нормальном завершении операции функция вернет значение 0. Значение функции IOResult необходимо присваивать какой-либо переменной, так как при каждом вызове функция обнуляет свое значение. Функция IOResult работает только при выключенном режиме проверок ошибок ввода - вывода или с ключом компиляции {$I-}.

типизированный файл - это последовательность компонент любого заданного типа (кроме типа "файл"). Доступ к компонентам файла осуществляется по их порядковым номерам. Компоненты нумеруются, начиная с 0. После открытия файла указатель (номер текущей компоненты) стоит в его начале на нулевом компоненте. После каждого чтения или записи указатель сдвигается к следующему компоненту.

Запись в файл:

Write(f, список переменных);

Процедура записывает в файл f всю информацию из списка переменных.

Чтение из файла:

Read(f, список переменных);

Процедура читает из файла f компоненты в указанные переменные. Тип файловых компонент и переменных должны совпадать. Если будет сделана попытка чтения несуществующих компонент, то произойдет ошибочное завершение программы. Необходимо либо точно рассчитывать количество компонент, либо перед каждым чтением данных делать проверку их существования (функция eof, см. выше)

Смещение указателя файла:

Seek(f, n);

Процедура смещает указатель файла f на n-ную позицию. Нумерация в файле начинается с 0.

Определение количества компонент:

FileSize(f): longint;

Функция возвращает количество компонент в файле f.

Определение позиции указателя:

FilePos(f): longint;

Функция возвращает порядковый номер текущего компонента файла f.

Отсечение последних компонент файла:

Truncate(f);

Процедура отсекает конец файла, начиная с текущей позиции включительно.

Чтение из текстового файла:

Read(f, список переменных);

ReadLn(f, список переменных);

Процедуры читают информацию из файла f в переменные. Способ чтения зависит от типа переменных, стоящих в списке. В переменную char помещаются символы из файла. В числовую переменную: пропускаются символы-разделители, начальные пробелы и считывается значение числа до появления следующего разделителя. В переменную типа string помещается количество символов, равное длине строки, но только в том случае, если раньше не встретились символы конца строки или конца файла. Отличие ReadLn от Read в том, что в нем после прочтения данных пропускаются все оставшиеся символы в данной строке, включая метку конца строки. Если список переменных отсутствует, то процедура ReadLn(f) пропускает строку при чтении текстового файла.

Запись в текстовый файл:

Write(f, список переменных);

WriteLn(f, список переменных);

Открытие нетипизированного файла:

Reset(f, BufSize)

Rewrite(f, BufSize)

Параметр BufSize задает число байтов, считываемых из файла или записываемых в него за одно обращение. Минимальное значение BufSize - 1 байт, максимальное - 64 К байт. Если BufSize не указан, то по умолчанию он принимается равным 128.

Чтение данных из нетипизированного файла:

BlockRead(f, X, Count, QuantBlock);