

1. Понятие алгоритма. Свойства алгоритма и виды реализации.

Алгоритм – четкая последовательность однозначных действий, приводящая к какому-либо результату.

Свойства: Детерминированность (определенность) – каждый шаг является однозначным

Результативность – приводит к какому-либо результату

Массовость – подходит к классу задач

Дискретность – представляет последовательность действий

Виды реализации: запись на алгоритмическом языке, блок-схема, псевдокод.

2. Основные этапы решения задач на ЭВМ.

1. Постановка задачи, 2. Формализация (перевод на язык математики) 3. Алгоритмизация (этап построения алгоритма), 4. Программирование (этап написания программы на выбранном языке программирования), 5. Отладка и тестирование (Отладка – процесс выявления и устранения ошибок в коде программы, Тестирование – проверка работоспособности программы на подобранном примере), 6. Передача в эксплуатацию, 7. Сопровождение (устранение ошибок в процессе эксплуатации)

3. Классификация языков программирования.

Язык программирования – способ записи программы для решения различных задач на ЭВМ в понятной для компьютера форме.

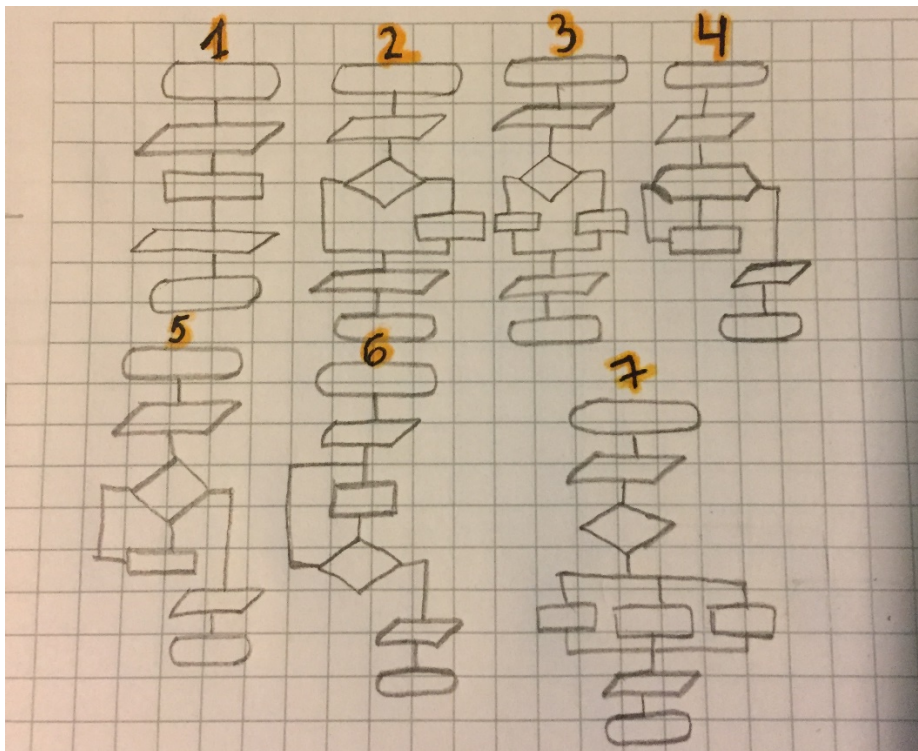
Процессор компьютера служит для восприятия языка машинных команд. В 50-е года появились первые средства автоматизации программирования – языки автокода или ассемблеры. Для восприятия компьютером программы на языке автокода требовался специальный переводчик – транслятор. Языки автокода и ассемблера являются *машинно-зависимыми* языками программирования (низкого уровня). То есть они настроены на структуру машинных команд одного компьютера. Первыми *машинно-независимыми* языками (высокого уровня) были Фортран, Алгол и Кобол. В 65-м году в Дартмутском университете был разработан Бейсик. В 71-м году появился Паскаль, как учебный язык структурного программирования. Язык программирования Си создавался как инструментальный язык для разработки ОС, трансляторов и т.д. Язык ЛИСП основан на рекурсиях, а Пролог реализует логическое программирование.

4. Понятие «данные» и «величины».

Данные – совокупность величин, с которыми работает компьютер. Они могут быть: входные, результирующие (выходные) и промежуточные. Всякая величина занимает свое определенное место в памяти ЭВМ. Любая величина имеет три основных свойства: 1. имя, 2. тип, 3. значение.

5. Базовые алгоритмические структуры.

Они бывают: 1 Линейные, 2-3 С неполным\полным ветвлением, 4-6 Циклические (полный цикл, с пост\пред условием, 7 Множественный выбор



6. Основные элементы программы и алфавит языка Pascal.

В паскале используются метки, константы, типы, переменные, подпрограммы.

Алфавит:

Латинские буквы от A до Z (прописные и строчные)

Цифры от 0 до 9

Специальные символы + - * / > < =; # ' , . : { } [] () и их комбинации: ":", "..", "<>", "<=", ">=", "{ }"

7. Структура программы на языке Pascal.

program <имя программы>;

label <раздел меток>;

const <раздел констант>;

type <раздел типов>;

var <раздел переменных>;

procedure (function) <раздел подпрограмм>;

begin

<программа>

end.

8. Основные типы данных в Pascal.

Типы данных делятся на: структурированные (массивы, множества, строки, записи, файлы), указательные и простые, которые в свою очередь делятся на порядковые (целые, перечисляемые, логические, интервальные, символьные) и вещественные.

9. Арифметические и логические операции в Pascal.

Арифметические операции: сложение (+), вычитание (-), умножение(*), вещественное деление(/), деление нацело (div), остаток от деления (mod).

Логические: логическое сложение\дизъюнкция (or), логическое умножение\конъюнкция (and), логическое отрицание (not), исключающее или (xor).

10. Процедуры ввода и вывода в Pascal.

Ввод данных – процедура передачи информации из внешних устройств в оперативную память. Вывод данных – передача данных из оперативной памяти во внешние устройства.

read (список ввода) \ readln (список ввода)

write (список вывод) \ writeln (список вывода)

11. Форматный вывод в Pascal.

I:P – выводится значение I в крайние правые позиции в поле шириной P - write(I:6)

R:P:Q – в крайние правые позиции выводится десятичное представление числа R с фиксированной точкой, после десятичной точки выводится Q цифр (от 0 до 24) – write (R:8:4)

12. Функции, связывающие различные типы данных в Pascal.

Ord (x) – дает порядковый номер значения x вещественного типа

Pred (x) – дает предыдущее по отношению к x значение вещественного типа

Succ (x) – дает следующее значение в типе

Chr (x) – дает символ с порядковым номером x

Odd (x) – выдает true, если x нечетное, и false, если x четное

13. Условный оператор и оператор множественного выбора в Pascal.

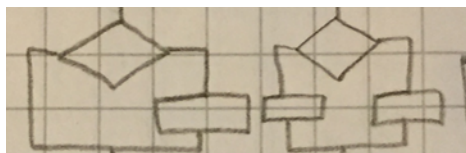
Условный оператор:

if (условие) then

действие;

if (условие) then

действие



else действие;

Множественный выбор:

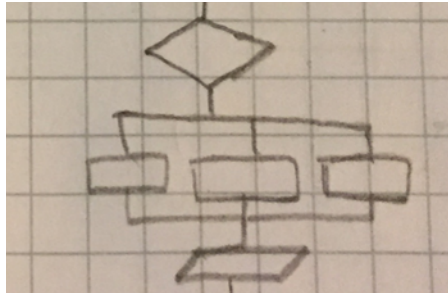
case переменная of

значение 1: действие;

значение n: действие

else действие;

end;



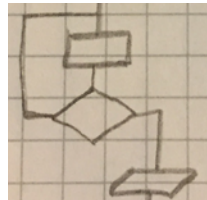
14. Виды циклов. Операторы цикла в Pascal.

С пост-условием:

repeat

тело цикла

until (условие)



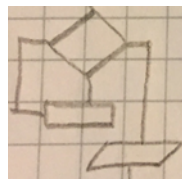
С пред-условием:

while (условие) do

begin

тело цикла

end;



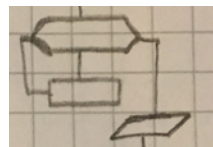
Цикл со счетчиком:

for переменная:=начальное значение to конечное значение do

begin

тело цикла

end;



15. Процедуры и функции в Pascal. Их отличия.

Существует два вида подпрограмм: процедуры и функции. Главное их отличие в том, что процедура не возвращает значение в главную программу, а функция через свое имя возвращает значение в главную программу.

Процедура:

procedure имя (параметры);

begin

тело процедуры;

end;

Функция:

function имя(параметры):тип;

begin

тело функции;

имя функции := значение;

end;

16. Понятие локальных и глобальных переменных.

Локальная переменная – переменная, которая доступна в определенном блоке программы, где она объявлена.

Глобальная переменная доступна во всей программе.

В си:

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
int i;                - глобальная
```

```
{int s=0;             - локальная
```

17. Работа с одномерными массивами в Pascal.

Массив – совокупность однотипных элементов, каждый элемент которой имеет свой индекс. В паскале массивы объявляются в разделе var.

var имя массива: array [начальное значение индекса..
конечное значение индекса] of тип;

Например: var a:array[1..30] of integer;

Ввод и вывод одномерного массива:

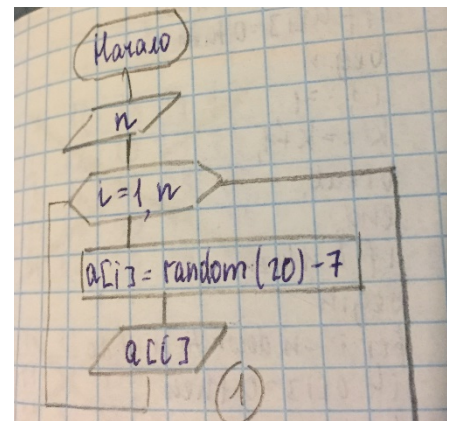
```
for i :=1 to 30 do
```

```
begin
```

```
a[i]:=random(20)-7;
```

```
writeln ( a [ i]);
```

```
end;
```



18. Работа с двумерными массивами в Pascal.

Массив – совокупность однотипных элементов, каждый элемент которой имеет свой индекс, а в двумерном массиве два индекса (один означает номер строки, другой – номер столбца). В паскале массивы объявляются в разделе var.

var имя массива: array [начальное значение индекса.. конечное значение индекса, начальное значение индекса.. конечное значение индекса] of тип;

Например: var a:array[1..30, 1..15] of integer;

Ввод и вывод двумерного массива:

for i :=1 to 30 do

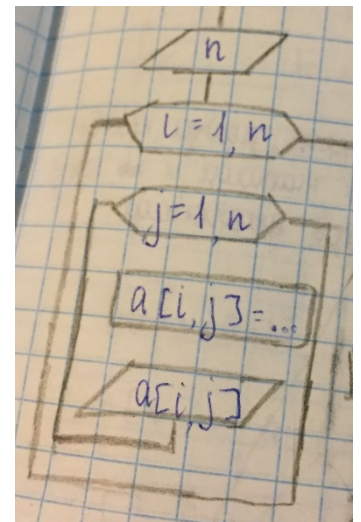
for j :=1 to 15 do

begin

readln (a[i,j]);

writeln (a[i,j]);

end;



19. Работа со строками в Pascal. Основные функции.

Строка – последовательность (массив) символов. Максимальная длина – 255 символов.

Var имя переменной:string;

var s:string;

s1:string[20];

Основные функции:

copy(s,poz,n) – выделяет из строки s подстроку длиной n символов, начиная с позиции poz

concat(s1,s2..sn); - конкатенация или сцепление, соединяет строки в одну

length (s) – определяет длину строки s

pos(s1,s2) – первое появление в строке s2 подстроки s1

delete (s,poz,n) – удаляет n символов из строки s, начиная с позиции poz

insert (s1,s2,poz) – вставляет строку s1 в строку s2 с позиции poz

20. Наличие и отсутствие оператора var в параметрах функции/процедуры.

При описании оператора с var в параметрах функции/процедуры происходит инициализация локальных переменных в функции/процедуре, но при этом значение вышесказанным переменным не присваивается.

При описании оператора без var в параметрах функции/процедуры, при её вызове в программе, происходит передача значений переменных в функцию. В последующем их значения относительно основной программы не меняются.

21. Понятие рекурсии.

Рекурсия – это способ организации вспомогательного алгоритма – подпрограммы, при котором эта программа (процедура или функция) в ходе выполнения обращается сама к себе.

Пример функции на паскале:

```
function fact (n:integer):extended;  
begin  
  if n<=1 then  
    fact:=1  
  else fact:=fact(n-1)*n;  
end;
```

22. Работа с файлами в Pascal.

Файл – поименованная область памяти.

Файловая переменная – структурированный тип, представляющий собой совокупность однотипных элементов, число которых заранее не определено.

```
assign (файловая переменная, папка);  
assign (f, 'Number.dat');  
close (f);
```

23. Комбинированный тип данных в Pascal.

Комбинированный тип данных — это структурированный тип, состоящий из фиксированного числа компонентов (полей) разного типа.

```
type имя базы = record  
  поле1:тип;  
  поле2:тип;  
end;  
  
type student = record  
  fio: string [50];  
  data: char;  
end;
```

24. Работа с графикой в Pascal. Основные функции.

```
Подключить модуль: uses Graph;  
Линия: Line(x1,y1,x2,y2);  
Установка цвета: SetColor(цвет);  
Прямоугольник: Rectangle(x1,y1,x2,y2);  
Круг: Circle(x,y,r);  
Сектор: PieSlice(x,y:integer,a,b,R);
```

Установка размера окна: `SetWindowSize (800,600);`

25. Язык программирования Си. Синтаксис языка, структура программы.

Типы данных: целый – `int`, `unsigned int`; вещественный – `float`, `double`; символьный – `char`; логический – `bool`.

Описание переменных: тип переменная (`int a; int b=4`)

Структура программы: `#include <библиотека.h>`

..

`void main()`

{описание переменных

программа}

26. Типы данных в Си. Преобразование типов.

Типы данных: целый – `int`, `unsigned int`; вещественный – `float`, `double`; символьный – `char`; логический – `bool`.

При операциях с разными типами данных тип с наименьшим диапазоном преобразуется в тип с большим. `Float – double`; `char – int`; `double – long double`.

27. Операции и выражения в Си. Правила записи арифметических операций в Си.

`a = 0;` `&&` - И `||` - ИЛИ `!=` - НЕ

28. Стандартные библиотеки ввода/вывода в Си. Функции `scanf` и `printf`.

1. `math.h` – синусы, косинусы и т.д.

2. `stdio.h` – библиотека форматного ввода\вывода (`%i,%d` – целый тип; `%f` – вещественный)

`printf`(“строка форматов”, список выводимых величин)

`printf`("n= %f", f);

`scanf_s`("%i", &n);

3. `iostream.h` – библиотека потокового ввода\вывода

`cin >> Переменная; cout << переменная;`

4. `iomanip.h` – форматный потоковый ввод\вывод

`setw(число); (cout << setw(5)<<a;)`

5. `stdlib.h` – для формирования случайных последовательностей

29. Функции препроцессора Си. Макроопределения.

Макроопределения напрямую связаны с макросами с Си. Макросы объявляются по следующему синтаксису:

`#define имя_макроса текст замещения`

Пример:

`#define pi 3.1415`

`y=pi*3-20`

Действие макроса распространяется на часть программы, ограниченную конструкцией `#define` и концом программы, для не особо типичных случаев возможно использование `#undef` в качестве конца матрицы.

Более мощные возможности препроцессора связаны с использованием макросов с параметрами. Их называют макроопределениями. Макроопределения похожи на функции, а их параметры на формальные параметры функции. В результате препроцессорной обработки, все макроопределения в исходном коде заменяются текстами замещения.

Например, пусть задано макроопределение с площадью круга и радиусом r :

Как видим мы:

```
#define s(r) (3.1415*r*r)
```

```
x=S(a)-S(b);
```

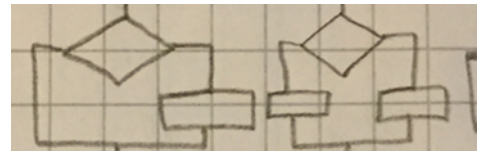
Как видит это дело компилятор:

```
x=(3.1415*a*a)-( 3.1415*b*b)
```

30. Алгоритмические конструкции ветвления, их реализация в Си.

```
if (условие) действие1;
```

```
else действие2;
```

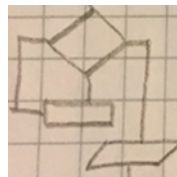


31. Циклические конструкции в Си.

С пред-условием:

```
While (условие)
```

```
{Тело цикла}
```

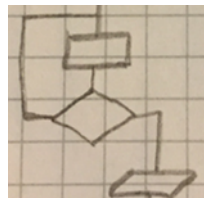


С пост-условием:

```
do
```

```
{Тело цикла}
```

```
while(условие);
```

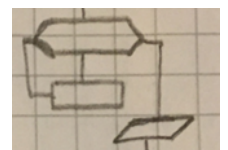


Цикл со счетчиком:

```
for (начало присвоения; условие окончания; модификация параметров)
```

```
for (i=0; i<3;i++)
```

```
for ( ; ; ) – бесконечный цикл
```



32. Особенности работы со строками в Си.

Объявление: `char s[10];`

`gets(строка)` – считывает всю строку

`'\0'` – символ конца строки

Используемые библиотеки: `string.h`, `ctype.h`, `stdlib.h`

`strcpy(s1,s2)` – копирует `s2` в `s1`;

`strcat(s1,s2,...)` – конкатенация;

strchr(s,ch) – поиск по номеру символа;
strrchr(s,ch) – последнее вхождение s в ch;
strlen(s) –длина строки;
strncpy(s1,s2,n) – копирует n от s2 в s1;
strncat(s1,s2,n) – присоединение n символов из s2 в s1;
strstr(s1,s2) – возвращает указатель строки s2 в строку s1;
strcmp(s1,s2) – сравнивает s1 и s2;
strncmp(s1,s2,n) – сравнивает первые n символов;
atoi(s) – преобразует, состоящую из цифр строку s, в целое число;
atol(s) – преобразует строку в длинное целое число (long);
atof(s) – преобразует строку в вещественный тип;
tolower(ch) – преобразует символ ch к строчному формату;
toupper(ch) - преобразует символ ch к прописному формату;
fscanf(f;”%s”,&s) – считывает строку;
fgets(f,s) – считывает строку из файла;
puts(значение) – вывод комментария;

33. Массивы данных. Описание и инициализация одномерного массива в Си.

Массив – совокупность однотипных элементов, каждый элемент которой имеет свой индекс.

Тип имя массива [размерность] (int a[10], b[5][6]) Для формирования случайного массива необходимо подключить библиотеку stdlib.h

Ввод одномерного массива:

```
#include <iomanip.h>
#include <iostream.h>

void main()
{int *a,i,n;
cout << “n=”;
cin>>n;
a = new int [n];
for (i=0;i<n;i++)
{a[i]=(i+2)-1;
cout << setw(6) << a[i];}
```

34. Двумерные массивы в Си.

Массив – совокупность однотипных элементов, каждый элемент которой имеет свой индекс, а в двумерном массиве два индекса (один означает номер строки, другой – номер столбца).

Тип имя массива [размерность] (int a[10], b[5][6]) Для формирования случайного массива необходимо подключить библиотеку `stdlib.h`

Ввод двумерного массива:

```
#include <iomanip.h>
#include <iostream.h>
#include <stdlib.h>

void main()
{int **b,i,n,m,j;
randomize;
cout << "n=";
cin>>n;
cout << "\n m=";
cin >>m;
b = new int [n];
for (i=0;i<n;i++)
{b[i] = new int[m];
cout<<"\n";
for (j=0;j<m;j++)
{b[i][j] = random(25)-10;
cout << setw(6) << b[i][j];}}
```

35. Адресная арифметика в си.

Если тебе этот вопрос попался, то тебе не повезло, прости :с

36. Использование указателей при работе с массивами данных.

Указатель – переменная, значением которой является адрес.

тип *имя указателя; (int *b;)

Для того, чтобы определить адрес, по которому записаны переменные, необходимо перед ее именем указать амперсен. Чтобы по адресу(указателю) определить значение, которое находит в ячейке памяти, перед символом ставим *.

Использование в массивах расписано в вопросе 33-34.

38. Динамические массивы.

Отличие динамического массива от статического в том, что на момент компиляции программы размерность динамического массива неизвестна, а в статическом массиве его размер должен быть известен до компиляции. Для выделения области памяти используется оператор `new`, а для освобождения - `delete`. Для того, чтобы динамически выделить память для переменной одного из базовых типов, объявляют указатель назначения соответствующего типа и в качестве значения этому указателю присваивают результат вызова оператора `new`.

Тип*указатель;

Указатель = new тип переменной;

Значением переменной указателя является адрес ячейки, выделенной для записи значения переменной.

Delete [] указатель;

43. Работа с файлами в Си.

Чтобы начать работу с файлами в Си, необходимо описать файловую переменную, которая предназначена для хранения адреса страницы, содержащей информацию о файле. Чтобы занести адрес указанной страницы в файловую переменную и записать в таблицу информацию необходимую для работы с файлом - (физическое имя файла, путь к файлу, режим доступа к файлу)

FILE *f,*f1;

Файловая переменная = fopen (<полное физическое имя файла>, <режим доступа>);

Режим доступа определяет каким образом предполагается использовать файл

"w" - создаёт новый файл и открывает его для записи

Если файл уже существует, то он стирается со всем своим содержимым и создаётся новый

"r" - открывает файл для чтения и указатель устанавливается в начало файла

"a" - открывает файл для добавления записи в его конец

Закрытие файла - fclose(файловая переменная);

Функция feof(файловая переменная) возвращает единицу, если указатель позиции в файле установлен на его конец, ноль - в противном случае.

Для чтения из файла используется функция - fscanf (файловая переменная, строка форматов выводимого значения, список выводимых переменных)

44. Рекурсия в Си.

Рекурсия – это способ организации вспомогательного алгоритма – подпрограммы, при котором эта программа (процедура или функция) в ходе выполнения обращается сама к себе.

45. Структуры языка Си.

Структура - группа переменных, объединённых одним именем. Элементы структуры часто называют Её полями.

```
struct имя
```

```
{тип 1 поле 1 ;
```

```
Тип 2 поле 2, поле 3;
```

```
..} список переменных;
```

Описание структуры - некий шаблон, по которому в последствии создаются переменные.

Обращение к полю структуры: имя структуры.поле

46. Модульное программирование в Си.

mas.cpp – название файла, под которым мы сохраняем модуль.

```
#include <iostream>
```

```
#include <stdlib.h>
```

```
void vvodm()
```

```
{int a[10],n,c;
```

```
cout<<"n=";
```

```
cin>>n;
```

```
for (i=0;i<n;i++)
```

```
{a[i]=rand()%20-7;
```

```
cout<<" "<<a[i];}
```

Открываем исходный код и пишем программу, в которую подключаем модуль.

```
#include "mas.cpp"
```

```
void main()
```

```
{vvodm();}
```

47. Методы сортировки данных.

Сортировка выбором:

```
void selatsort (int *a, int n);
```

```
{int buf,i,j,pos;
```

```
for (i=0;i<n-1;i++)
```

```
{pos = i;
```

```
buf = a[i];
```

```
for (j=i+1;j<n;j++)
```

```
if (a[j]<buf)
```

```
{pos = j;
buf = a[j];}
a[pos] = a[i];
a[i] = buf;}}
```

Сортировка методом вставок:

```
void insertsort (int *a, int n)
{int i,j,buf;
for (i=1,i<n;i++)
{buf = a[i];
for (j = i-1;j>=0&& a[j]>buf;j--)
a[j+1] = a[j];
a[j+1] = buf;}}
```

Сортировка пузырьком:

```
Void bubblesort (int *a, int n)
{int i,j,buf;
For (i=0;i<n-1;i++)
For (j=i+1;j<n;j++)
If (a[i]>a[j])
{buf = a[i];
A[i]=a[j];
A[j]=buf;}}
```

49. Понятие «объект», «семейство», «класс».

Объект – совокупность отдельных информационных элементов и функций, которые им оперируют.

Объект ООП – совокупность переменных состояния и связанных с ними методов (операций). Эти методы определяют как объект взаимодействует с окружающим миром. (Например: форма (userform), рабочий лист (worksheet), диаграмма (chart))

Семейство – объект, содержащий несколько других объектов, как правило, одного и того же типа.

Например, объект workbooks (рабочие книги) содержит все рабочие объекты workbook (рабочая книга). Каждый элемент семейства нумеруется и может быть идентифицирован либо по номеру, либо по имени.

Класс – множество объектов, имеющих общую структуру и общее поведение.

Класс определяет имя объекта, его свойства и действия, выполняемые над объектом. В свою очередь каждый объект является экземпляром класса.

50. Понятие «свойство», «метод», «событие».

Метод – действие, выполняемое над объектом.

Свойство – атрибут объекта, определяющий его характеристики.

Событие – действие, распознаваемое объектом (например, щелчок мыши).















Процедуры обработки событий – специальный вид процедур, генерирующий отклик на события.

51. Объект «пользовательская форма». Основные элементы управления.

1. Вставка, форма (Insert, Form)

2. Вставить UserForm (Insert UserForm)

Панель элементов ToolBox

Элемент управления	Имя объекта	Кнопка
поле	TextBox	
надпись	Label	
поле со списком	ComboBox	
список	ListBox	
флажок	CheckBox	
переключатель	OptionBox	
рамка	OptionButton	
выключатель	Frame	
полоса прокрутки	ScrollBar	
счётчик	SplinButton	
изображение	Image	
набор вкладок	TabStrip	
набор страниц	MultiPage	
кнопка	CommandButton	

52. Общие свойства элементов управления.

Name	Имя объекта (оно всегда уникальное)
Top, Left	Изменение положения объекта на форме
Visible	Видимость объекта (True/False)
Enabled	Доступ к объекту (True/False)
Font	Шрифт объекта
BackColor	Цвет заднего фона объекта
ForeColor	Цвет шрифта
Height, width	Высота и ширина объекта
BorderColor	Цвет границы объекта
BorderStyle	Стиль границы

53. Дополнительные элементы управления.

Дополнительные элементы управления являются самостоятельными объектами, обладающими как общими для всех элементов управления свойствами и методами, так и присущими только им свойствами и методами. Для добавления дополнительных элементов управления на панель элементов необходимо:

- Выбрать команду Сервис + Дополнительные элементы (Tools + Additional Controls);
- В появившемся на экране окне Дополнительные элементы (Additional Controls) в списке Доступные элементы (Available Controls) установить флажок напротив добавляемого элемента;
- Нажать кнопку ОК.

Удаление ненужного элемента управления из панели элементов происходит аналогично добавлению, только флажок снимают.

Среди дополнительных элементов управления очень полезным является элемент управления Calendar (календарь). Этот объект предоставляет средство для организации удобного интерфейса по вводу дат. Элемент управления конструируется в форме с помощью кнопки Календарь (Calendar).

Его основные свойства:

Day	Возвращает выбранный день
DayFont, DayFontColor	Устанавливают шрифт и цвет шрифта для названий дней недели
FirstDay	Первый день недели. Допустимые значения от воскресенья (Sunday) до субботы (Saturday)
Month	Возвращает выбранный месяц
MonthLenght	Допустимые значения: длинный (Long) (отображаются полные названия месяца) и короткий (Short) (отображаются только первые три буквы из названия месяца)
ShowDays	Допустимые значения: True (отображаются названия дней недели) и False (в противном случае)
ShowData-selected	Допустимые значения: True (отображается выбранная дата в верхней части календаря) и False (в противном случае)
Value	Возвращает выбранную дату
Year	Возвращает выбранный год

54. Понятие «макроса» и «macroRecorder».

Макрос – это программа на языке VBA

MacroRecorder – это транслятор, создающий макрос на языке VBA, которая является результатом перевода на язык VBA действий пользователя

Запись макроса с помощью MacroRecorder'а можно осуществить при помощи перехода на вкладку Вид =>Макросы=>Запись макроса

55. Типы данных в VBA.

Целый: byte (0..255), integer (-32768..32767), long

Вещественный: single, double, currency

Строковый: string (переменная), string (постоянная)

Дата: date

Логический: Boolean

Любой указатель объекта: object

Любое числовое значение вплоть до границ диапазона значений double: variant

Любое числовое значение вплоть до границ диапазона значений string: variant

56. Синтаксис описания переменных и алфавит языка VBA.

Описание переменных: Dim имя_переменной As тип_данных

Например, Dim x As Double

Длина переменной не более 255 символов, не может содержать %,.,&!,#,@,\$, должна начинаться с буквы, номера должны быть уникальны в своей области. Также не следует использовать имена, совпадающие с ключевыми словами VBA и именами встроенных функций и процедур.

57. Операции в VBA.

Существует три основных типа операций: арифметические, логические и операции отношения.

58. Математические функции в VBA.

atn – арктангенс

tan – тангенс

log – натуральный логарифм

rnd – генератор случайных чисел

int, fix – округление чисел

Остальные такие же как в Паскале и Си.

59. Функции проверки типов в VBA.

Isarray – является ли массивом

IsDate – является ли датой

IsEmpty – было ли описано через Dim

IsError – является ли кодом ошибки

IsNumeric – является ли числом

IsNull – является ли пустым значением

IsObject – является ли переменная объектом

60. Функции преобразования форматов в VBA.

Val – число типа variant

Str – строка типа variant

Cbool - Boolean

Cbyte - Byte

Ccur - Currency

Cdate - Date

Cdbl - Double

Cint - Integer

Cstr - String

Cvar - Variant

Clng - Long

61. Функции обработки строк в VBA.

В VBA есть отдельный тип String, объявление типичное:

```
Dim var as String
```

Соединение строк происходит тоже просто:

```
"Example1"+" Example2"
```

Для добавления другого типа в строковую переменную необходимо преобразование этого типа:

```
Dim meow as String
```

```
Dim kot as Integer
```

```
kot = 3
```

```
meow = "МЯУ МЯУ "
```

```
meow = meow+cstr(kot)
```

```
'В выводе переменной meow будет "МЯУ МЯУ 3"
```

Символы пробела и enter'a являются соответственно chr(9) и chr(13)

Пример:

meow = meow+chr(13)+"ПОМОГИТЕ Я КОТ"
'Выведет "ПОМОГИТЕ Я КОТ" с новой строки

Существует достаточно большое кол-во функций работы со строками, но наиболее частыми в употреблении являются следующие:

- Len(строка) - Возвращает целое число, показывающее число знаков в строке (очень удобна при работе в цикле по строке)
- Asc(символ) - Возвращает значение типа Integer, представляющее код знака, соответствующий знаку.
- Chr(значение в int) - функция обратная Asc, возвращает символ по его коду в формате строки.
- InStr(строка) - Возвращает целое число, указывающее начальную позицию первого вхождения одной строки в другую.
- Replace(строка, подстрока1, подстрока2) - Возвращает строку, в которой указанная подстрока заданное число раз заменена другой подстрокой. Пример:
Dim aString As String = Replace(TestString, "o", "i")
- StrComp(строка1, строка2) - Возвращает -1, 0 или 1 в зависимости от результата сравнения строк.
- UCase(строка) - Возвращает строку или знак, содержащий указанную строку, преобразованную в верхний регистр.
- LCase(строка) - Возвращает строку или символ, преобразованные в нижний регистр.

62. Функции времени и даты в VBA.

Функция	Действие
Date()	Возвращает текущую системную дату
Time()	Возвращает текущее системное время
Now()	Возвращает текущие дату и время
Hour()	Возвращает текущий час
Minute()	Возвращает текущую минуту
Second()	Возвращает текущую секунду
Day()	Возвращает текущий день
Month()	Возвращает текущий месяц
Year()	Возвращает текущий год (вдруг забыл?)

63. Встроенные диалоговые окна в VBA.

Существует два основных вида встроенных диалоговых окон в VBA:

- Окно ввода информации (InputBox)
- Окно сообщений (MsgBox)

Общий синтаксис выражения в InputBox:

InputBox (*prompt* [, *title*] [, *default*] [, *xpos*] [, *ypos*] [, *helpfile*] [, *context*])

Prompt – строковое выражение, отображаемое как сообщение в диалоговом окне

Title – заголовок в диалоговом окне

Default – значение по умолчанию

Xpos, Ypos – параметры места появления диалогового окна на экране

Helpfile – файл справки (путь к нему)

Context – номер раздела в файле справки

Общий синтаксис выражения в MsgBox:

MsgBox (*prompt* [, *buttons*] [, *title*] [, *helpfile*] [, *context*])

Buttons – позволяет настроить число и тип отображаемых кнопок, тип значка, основную кнопку

64. Условный оператор и оператор множественного выбора в VBA.

If (условие) then оператор1 else оператор2

Если поле then и\или else содержит более одного оператора, то в конце этой конструкции ставится end if.

65. Операторы цикла в VBA.

Полный цикл: for счетчик = начало To конец [step шаг] инструкции

next счетчик

По умолчанию шаг равен одному.

Пред-условие: while (условие)

Инструкции

Wend

Или do while (условие)

Инструкции

Loop

Пост-условие: do инструкции

Loop while (условие)

66. Процедуры и функции в VBA.

Функция: [public/private] sub имя (аргументы)

Инструкции

End Sub

Процедура: [public/private] sub имя (аргументы)

Инструкции

End sub

Об отличии процедур и функции см. пункт 15.

67. Технология ADO. Принципы работы.

Технология ADO - технология, которая позволяет с помощью некоторых конструкций установить связь с внешней базой данных и впоследствии работать с ней, не открывая приложения, в котором эта база данных реализована.

Объектная модель ADO состоит из следующих объектов высокого уровня и семейств объектов:

- Connection (представляет подключение к удаленному источнику данных)
- Recordset (представляет набор строк, полученный от источника данных)
- Command (используется для выполнения команд и SQL запросов с параметрами)
- Record (может представлять одну запись объекта Recordset или же иерархическую структуру, состоящую из текстовых данных)
- Fields (представляет столбцы таблицы базы данных)
- Parameters (представляет набор параметров SQL-инструкции)
- Properties (представляет набор свойств объекта)

В начале модуля идет блок описания переменных. С помощью переменной "cn" осуществляется связь с БД, а с помощью переменных "rs" и "rsp" осуществляется выгрузка данных из БД

```
Public cn as ADODB.Connection
Public rs as ADODB.Recordset
Public rsp as ADODB.Recordset
Public fl as Integer
```

Процедура инициализации формы - в ней происходит подключение к базе данных (создание связи, выбор драйвера, указание пути к БД, активизация данной связи) и выгрузка существующих данных из таблицы при помощи SQL-запроса.

```
Private Sub UserForm_Initialize()
    Set cn = New ADODB.Connection
    cn.Provider = "Microsoft.ACE.OLEDB.12.0"
    cn.ConnectionString = "C:\путь.mdb"
    cn.Open
    Set rs = New ADODB.Recordset
    rs.CursorType = adOpenKeyset
    rs.LockType = adLockOptimistic
    rs.Source = "SELECT Дисциплина.* FROM Дисциплина order by Дисц.Код_пред"
    Set rs.ActiveConnection = cn
    rs.Open
```