ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ИРКУТСКОЙ ОБЛАСТИ
«ЧЕРЕМХОВСКИЙ ГОРНОТЕХНИЧЕСКИЙ КОЛЛЕДЖ им. М.И. ЩАДОВА»

РАССМОТРЕНО
на заседании ЦК
«Информатики и ВТ»
Протокол №5
«09» января 2024 г.
Председатель: Чипиштанова Д.В.

УТВЕРЖДАЮ
Зам. директора по УР
О.В. Папанова
«22» февраля 2024 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения

самостоятельных работ студентов

по учебной дисциплине

ОП.04 Основы алгоритмизации и программирования

программы подготовки специалистов среднего звена

09.02.07 Информационные системы и программирование

Разработал:
Н.С.Коровина

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 54CE6FF6F2E941EC6571CFEA2EAFB5E4
Владелец: Сычев Сергей Николаевич
Действителен: с 23.08.2024 до 16.11.2025

2024 г

1. ПЕРЕЕНЬ САМОСТОЯТЕЛЬНЫХ РАБОТ

№ п/п	Тема самостоятельной работы	Количество часов	Оценка и контроль
1	Основные принципы алгоритмизации и программирования	4	защита
2	Операторы языка программирования	2	защита
3	Операторы языка программирования	2	защита
4	Операторы языка программирования	2	защита
5	Процедуры и функции	2	защита
6	Структуризация в программировании	2	защита
7	Указатели	2	защита
8	Основные принципы объектно-ориентированного программирования (ООП)	2	защита
9	Интегрированная среда разработчика	2	защита
10	Визуальное событийно-управляемое программирование	2	защита
11	Разработка оконного приложения	2	защита
12	Этапы разработки приложений	2	защита

2. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНЫХ РАБОТ
САМОСТОЯТЕЛЬНАЯ РАБОТА №1

Тема: Основные принципы алгоритмизации и программирования

Цель: научиться разрабатывать блок-схем алгоритмов.

Методические указания:

Задание 1. Составить алгоритм и решить задачу по варианту:

ПЕРЕЧЕНЬ ВАРИАНТОВ

№ Варианта	№ Задачи	Первая буква фамилии
1	**1,4**	А, К, Ф, Ж, С, Я, Д, О, Щ
2	**2,5**	Б, Л, Х, Ё, Р, Ю, Г, Н, Ш, И, У, Й
3	**3,6**	В, М, Ч, З, Т, Ц, Е, П, Э

Задача 1. Даны два круга с общим центром и радиусами R_1 и R_2 ($R_1 > R_2$). Найти площади этих кругов S_1 и S_2, а также площадь S_3 кольца, внешний радиус которого равен R_1, а внутренний радиус равен R_2: $S_1 = \pi \cdot (R_1)_2$, $S_2 = \pi \cdot (R_2)_2$, $S_3 = S_1 – S_2$.

Задача 2. Даны три точки A, B, C на числовой оси. Точка C расположена между точками A и B. Найти произведение длин отрезков AC и BC.

Задача 3. Найти расстояние между двумя точками с заданными координатами (x_1, y_1) и (x_2, y_2) на плоскости.

Задача 4. Решить линейное уравнение $A \cdot x + B = 0$, заданное своими коэффициентами A и B (коэффициент A не равен 0).

Задача 5. Дана масса M в килограммах. Используя операцию деления нацело, найти количество полных тонн в ней (1 тонна = 1000 кг).

Задача 6. Дано трехзначное число. Найти сумму и произведение его цифр.

Задание 2. Описать исходные, выходные и промежуточные данные и составить блок-схему решения для задачи по варианту в программе MS Visio.

варианта	1-ая буква фамилии	задача
1	А, Б, В	Дан диаметр окружности d. Найти ее длину $L = \pi \cdot d$. В качестве значения π использовать 3.14.

2	Г, Д, Е,	Дана длина ребра куба a. Найти объем куба $V = a3$ и площадь его поверхности $S = 6 \cdot a^2$.
3	Ж, З, И	Даны длины ребер a, b, c прямоугольного параллелепипеда. Найти егообъем $V = a \cdot b \cdot c$ и площадь поверхности $S = 2 \cdot (a \cdot b + b \cdot c + a \cdot c)$.
4	К, Л	Найти длину окружности L и площадь круга S заданного радиуса R:$L = 2 \cdot \pi \cdot R$, $S = \pi \cdot R^2$.
5	Н, О, П	Даны два числа a и b. Найти их *среднее арифметическое*: $(a + b)/2$.
6	Р, С, Т	Даны два неотрицательных числа a и b. Найти их *среднее геометрическое*, то есть квадратный корень из их произведения: $a \cdot b$.
7	У, Ф, Х	Даны два ненулевых числа. Найти сумму, разность, произведение и частное их квадратов.
8	Ч, Ш, Щ	Даны катеты прямоугольного треугольника a и b. Найти его гипотенузу c и периметр P: $c = a^2 + b^2$, $P = a + b + c$.
9	Э, Ю, Я	Дана длина L окружности. Найти ее радиус R и площадь S круга, ограниченного этой окружностью, учитывая, что $L = 2 \cdot \pi \cdot R$, $S = \pi \cdot R^2$. В качестве значения π использовать 3.14.
10	Ё, М	Найти расстояние между двумя точками с заданными координатами $x1$ и $x2$ на числовой оси: $\|x2 - x1\|$.

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №2

Тема: Операторы языка программирования

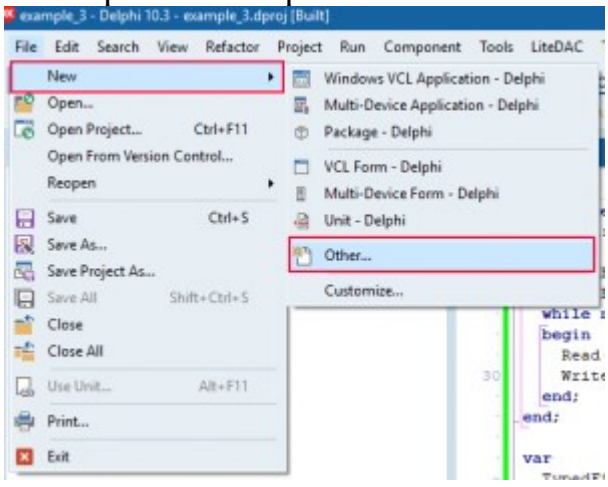
Цель: создавать проект с использованием текстовых файлов.

Методические указания: Работа с файлами. Создание проектов с использованием текстовых файлов.

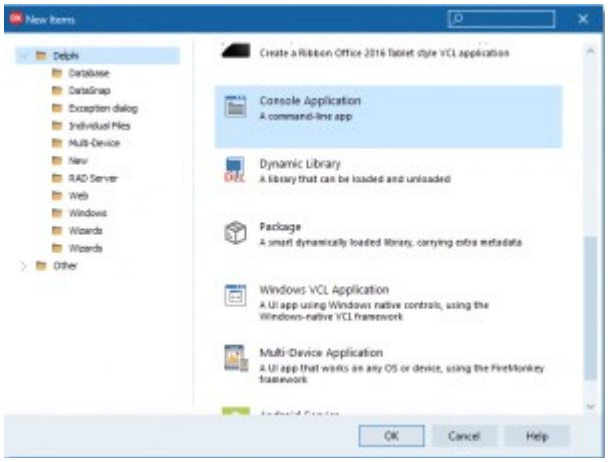
Решить задачу. Ввести текст с клавиатуры, сохраняя каждую введенную строку в файле text.txt. Работа продолжается, пока не будет введена пустая строка.

1. **Создаем новое консольное приложение в Delphi**

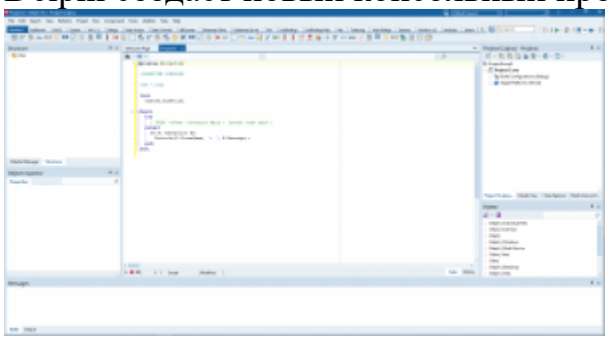
Открываем Delphi и в меню File выбираем «*File — New — Other*«:



В открывшемся окне выбираем в списке слева пункт «Delphi», а справа — «Console Application»:



Delphi создаст новый консольный проект:



2. Пишем код программы для работы с текстовым файлов в соответствии с заданием

```
program TextFiles;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  System.SysUtils;

var F: TextFile;
    s: string;
begin
  AssignFile(F, 'text.txt');
  Rewrite(F);
  repeat
    writeln('Введите новую строку:');
    Readln(s);
    if Length(S)&gt;0 then
      Writeln(F, s);
  until Length(s)=0;
  CloseFile(F);
end.
```

Рассмотрим алгоритм решения этой задачи:

1. Определяем переменные в разделе *var*

F: TextFile;

— это файловая переменная для работы с текстовым файлом

```
s: string;
```
— это переменная для хранения строки, введенной пользователем с клавиатуры
2. Ассоциируем файловую переменную с внешним файлом на диске:
```
AssignFile(F, 'text.txt');
```
3. После того, как файловая переменная ассоциирована с внешним файлом, открываем текстовый файл, используя метод *Rewrite*.
```
Rewrite(F);
```
Для открытия текстового файла в Pascal/Delphi могут использоваться три метода — *Rewrite*, *Reset* и *Append*. Однако, так как в тексте задачи не ставится вопрос о том, должен ли файл создаваться заново или файл должен сохраняться и так далее, то я использовал кратчайший путь решения задачи, используя метод *Rewrite*, который создает новый файл (если файл отсутствует на диске) и открывает его.
4. Создаем цикл с постусловием *repeat…until* в котором запрашиваем у пользователя ввод очередной строки
```
 repeat
   writeln('Введите новую строку:');
   Readln(s);
   if Length(S)&gt;0 then
     Writeln(F, s);
 until Length(s)=0;
```
Условие выхода из цикла — длина строки (*Length(s)*) равна нулю. Внутри цикла проверяется длина строки и, если длина строки больше нуля, то строка *s* записывается в файл — это делается для того, чтобы пустая строка не записалась в файл перед выходом из программы.
Цикл ***repeat..until*** можно заменить на цикл с предусловием ***while..do***, например, так:
```
 writeln('Введите новую строку:');
 Readln(s);
 while Length(s)&gt;0 do
   begin
     Writeln(F, s);
     writeln('Введите новую строку:');
     Readln(s);
   end;
```
Улучшение программы
Для улучшения программы и, возможно, поощрения со стороны преподавателя, можно предложить следующее улучшение нашей программы — вынести строки приглашение ввода от пользователя и имя файла в раздел констант, что позволит упростить внесение изменений в программу, с одной стороны, а, с другой стороны — ваш Delphi-код станет более чистым :) Улучшенная версия нашей программы может выглядеть, например, вот так:
```
program TextFiles;

{$APPTYPE CONSOLE}

{$R *.res}
```

```
uses
  System.SysUtils;

const
  cFileName = 'text.txt'; //имя файла для записи
  cInputStr = 'Введите новую строку:';//строка-приглашение для пользователя

var F: TextFile;
    s: string;
begin
  AssignFile(F, cFileName);
  Rewrite(F);
  writeln(cInputStr);
  Readln(s);
  while Length(s)&gt;0 do
    begin
      Writeln(F, s);
      writeln(cInputStr);
      Readln(s);
    end;
  CloseFile(F);
end.
```

Самостоятельно реализуйте данную задачу для формы

Форма отчета: файл (задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №3

Тема: Операторы языка программирования

Цель: создавать проект с использованием текстовых файлов.

Методические указания: Работа с файлами. Создание проектов с использованием текстовых файлов. Файлы последовательного доступа.

Задание. Реализовать задачи приложения 1 и приложение 2.

В Pascal традиционно используются файлы трех типов — текстовые, типизированные и двоичные. В отличие от Basic и C, где с именами открываемых файлов связываются программные числовые номера, в Pscal их заменяют переменные файлового типа. Эти переменные объявляются в разделе описания переменных:

```
type
PhoneRec = record
   Name:string[20];
   Phone:string [15];
   Address:string [40];
   BirthDay:TDateTime;
end;

var
  ft:TextFile;          {файловая переменная для текстового файла}
  fb:File;             {файловая переменная для двоичного файла}
  fr:File of PhoneRec; {файловая переменная для типизированного файла}
```

После этого в разделе исполняемых операторов имена файловых переменных с помощью процедуры *AssignFile* связываются с именами дисковых файлов соответствующего типа:

```
AssignFile(ft,'abc1.txt');
AssignFile(fb,'abc2.bin');
AssignFile(fr,'abc3.rec');
```

И только теперь можно приступить к процедуре открывания файлов. Не в обиду будет сказано, но авторы Паскаля здесь умудрились выбиться из общепринятой колеи и вместо четкого *Open* взяли на вооружение гораздо менее привычные служебные слова *Reset* (для файлов ввода), *Rewrite* (для файлов вывода) и *Append* (для пополнения текстовых файлов):

```
 Reset (ft);         Rewrite(ft);         Append(ft);
Reset(fb[,len]);  Rewrite(fb[,len]);
Reset(fr);         Rewrite(fr);
```

Второй необязательный аргумент *len*, используемый при открытии двоичных файлов, устанавливает в байтах размер порции данных, которые принимают участие в процедурах обмена. По умолчанию используются блоки размером по 128 байт, но эта цифра, скорее, дань традиции Pascal. С точки зрения оптимизации скорости обмена длина блока должна быть равной длине кластера. Однако и это не совсем точно, т. к. на время передачи данных существенно большее влияние оказывает аппаратура управления винчестером, располагающая своим собственным кэшем.

Открытые файлы любого типа закрываются с помощью процедуры *CloseFile*:

```
 CloseFile(ft);    CloseFile(fb);     CloseFile(fr);
```

Для обмена с текстовыми файлами используются обычные операторы *read/readln* и *write/writeln*, первым аргументом которых являются имена переменных файлового типа:

```
 write(ft, x1:10,x2:6);
writeln(ft,' ',x3);
.......................
readln(ft, y1, y2, y3);
```

Чтобы очередная порция данных, записываемая в текстовый файл, была завершена признаком конца строки (символ "Возврат каретки" или пара символов "Возврат каретки" и "Перевод строки"), вывод последней (или единственной) части строки должен быть завершен оператором *writeln*. Кроме этого, пользователь должен позаботиться о том, чтобы числовые данные, направляемые на диск, разделялись, как минимум, одним пробелом. В цикле считывания строк из текстового файла программа может определить момент исчерпания данных по значению логической функции *Eof (ft)* . Она принимает значение *True*, если между текущей позицией указателя и концом файла ничего кроме пробелов нет.

Приложение 1 (Delphi)

Приводимое ниже приложение осуществляет побайтовое копирование содержимого одного текстового файла в другой. Первый текстовый файл должен существовать, т. к. его имя выбирается пользователем в стандартном диалоговом окне *opendialog1*. Второй файл может быть создан - такая возможность предусмотрена в диалоговом окне *saveoDialog1*. Процедура переписи включена в состав обработчика события *onActivate*.

```
 unit Unit1;
interface
uses
  Windows,Messages,SysUtils,Variants,Classes,Graphics,Controls,Forms, Dialogs;
type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormActivate(Sender: TObject);
var
  ft1, ft2: TextFile;
  ch: Char;
begin
  if OpenDialog1.Execute then begin
    AssignFile(ft1,OpenDialog1.Filename);
    Reset(ft1);
    if SaveDialog1.Execute then
    begin
      AssignFile(ft2,SaveDialog1.Filename);
      Rewrite(ft2);
      while not Eof(ft1) do begin
        Read(ft1,ch);
        Write(ft2,ch);
      end;
      CloseFile(ft2);
    end;
    CloseFile(ft1);
  end;
end;
end.
```

В обмене с типизированными файлами используются только операторы *read* и *write*. При этом источником или получателем данных должна быть единственная переменная типа *record*, имеющая такую же структуру, которая была использована при объявлении файла.

Приложение 2 (Delphi)

В приведенном ниже приложении на форме находятся две кнопки -Button1 с надписью Записать в файл и Button2 с надписью Прочитать из файла. Щелчок по первой кнопке приводит к созданию типизированного файла с именем Poly.dat, записи которого представляют координаты вещественных точек. В обработчике этого же события координаты точки P1 записываются в файл. Щелчок по второй кнопке извлекает координаты первой точки из файла в запись P2.

```
 unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
type
  DPoint=record
    x:double;
    y:double
  end;
var
```

```
  Form1: TForm1;
  P1,P2:DPoint;
  fp:file of DPoint;
implementation
{$R  *.dfm}
procedure TForm1.ButtonlClick(Sender: TObject);
begin
  AssignFile(fp,'Poly.dat');
  Rewrite(fp);
  P1.x:=1;
  P1.y:=2;
  Write(fp,P1);
  CloseFile(fp);
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  AssignFile(fp,'Poly.dat1);
  Reset(fp);
  Read(fp,P2);
  CloseFile(fp);
end;
end.
```

При обмене с типизированным файлом допустим переход к любой существующей записи с заданным номером (записи в файле нумеруются от о). Установка указателя файла на запись с номером *nr* осуществляется процедурой *Seek*:

```
Seek(fr,nr);
```

С двоичными файлами обмен производится блоками, размер которых установлен либо по умолчанию (128 байт), либо был объявлен в момент открывания файла. Для этой цели используются специальные операторы *BlockWrite* и *BlockRead*:

```
BlockWrite(fb, buf, n_block,v1);
BlockRead(fb, buf, n_block,v1);
```

Здесь *buf* — массив (обычно типа *byte*), в котором либо находятся данные, подготовленные для записи на диск, либо предназначенный для хранения считываемых с диска данных.
Параметр *n_block* задает количество блоков, участвующих в обмене. В переменную *v1* заносится фактическое количество записанных или считанных блоков. Несовпадение между *n_block* и значением *v1* при чтении обычно связано с тем, что реальная длина файла не всегда кратна размеру блока. При выводе такое событие, как правило, свидетельствует об исчерпании дисковой памяти. Подобно записям в типизированных файлах блоки данных в двоичном файле тоже нумеруются от 0.

И для перехода к блоку с нужным номером необходимо воспользоваться процедурой *Seek*.

Перечень дополнительных процедур и функций по управлению файлами в системе Delphi приведен ниже в табл. 1. *Процедуры и функции для работы с файлами, принятые в среде Delphi*

Процедура/Функция	**Назначение**
ChDir	Назначает новый текущий каталог на текущем диске
Eoln	Возвращает значение *True,* если в строке текстового файла обнаружен признак конца строки
Erase	Уничтожает указанный файл
FilePos	Возвращает номер текущей записи или текущего блока в типизированном или двоичном файле
FileSize	Возвращает количество записей или блоков в типизированном или двоичном файле

Flush	Выталкивает содержимое буфера обмена в выводной текстовый файл
GetDir	Возвращает имя текущего каталога на указанном логическом диске
IOResult	Возвращает числовой код, свидетельствующий о результате завершения предыдущей операции с дисковым файлом. При нормальном завершении *IOResult=0*
MkDir	Создает новый подкаталог
Rename	Переименовывает указанный файл
RmDir	Удаляет пустой подкаталог
SeekEoln	Возвращает значение *True*, если между текущей позицией указателя и концом строки в текстовом файле нет значащих символов кроме пробела
SetTextBuf	Изменяет длину встроенного буфера у текстового файла
Truncate	Удаляет данные от текущей позиции указателя до конца типизированного или двоичного файла

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №4

Тема: Операторы языка программирования

Цель: создавать типизированные и не типизированные файлы.

Методические указания: Типизированные и не типизированные файлы.

Типизированные файлы

Все компоненты такого файла принадлежат к одному типу; тип может быть любым, кроме файлового.

Компоненты размещены непосредственно один за другим.

Файл хранится только на внешнем запоминающем устройстве (в частности, на диске).

Объявление файловой переменной

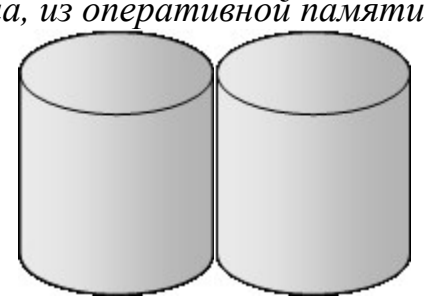
Var <идентификатор> :<имя файлового типа>; //ссылка на файловый тип

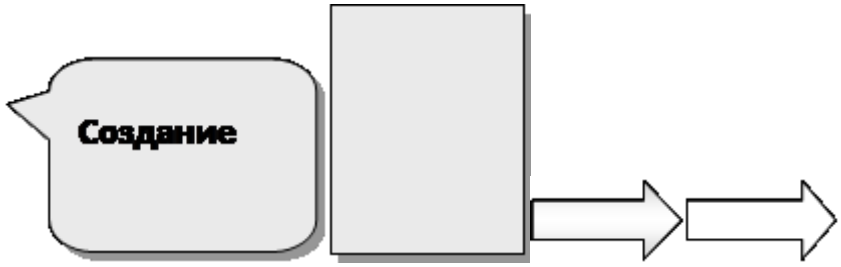
Var <идентификатор>: File Of <тип компонента>;//без ссылки на тип

Обычно <тип компонента> является типом записи.

Информация в типизированных файлах хранится *во внутреннем представлении*.

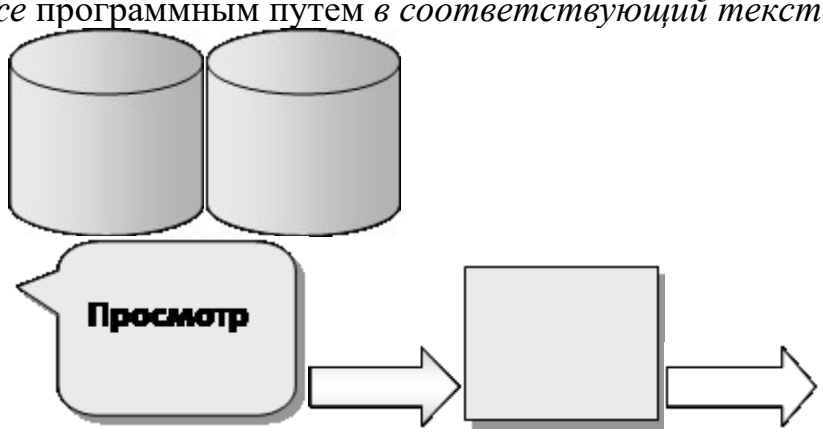
Поэтому такой файл может быть создан только программным путем - *путем записи содержимого некоторой переменной того же типа, что и тип файла, из оперативной памяти в файл:*



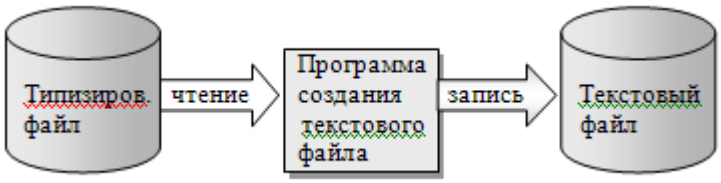


Программа создания типизированного файла — запись в Типизированный файл — чтение Типизированного файла

По этой же причине такой файл нельзя просмотреть непосредственно, как текст (при такой попытке содержимое файла, физически являющее собой последовательность байтов, на экране будет представлено последовательностью символов, закодированных этими байтами согласно таблице кодов). Для просмотра типизированный файл необходимо перевести также программным путем *в соответствующий текстовый файл:*



Программа создания текстового файла — запись в Текстовый файл — чтение Типизированного файла



Компоненты типизированного файла считываются и записываются целиком.

Открытие файла

Файловая переменная связывается с именем файла в результате обращения к стандартной процедуре ***AssignFile(F;S)***

Процедуры Reset***(F), Rewrite(F)*** открывают файл для чтения и для записи, соответственно.

Процедуры обработки файла

Read(F,<список ввода>) **-** читает из открытого файла F в указанные переменные из списка ввода такого же типа, что и компоненты файла.

Write (F,<список вывода>) **-** записывает в открытый файл F значения из указанных переменных из списка такого же типа, что и компоненты файла.

Seek(F;N:Longint) - устанавливает текущую запись файла под номером N (отсчет записей с нуля).

Закрытие файла осуществляется процедурой ***CloseFile(F)***.

Erase(F) - удаляет с диска файл F.

Файлы последовательного доступа

Общая схема создания последовательного файла: открытие файла (AssignFile, Rewrite), цикл формирования и вывода записей (Write), закрытие файла (CloseFile).

Порядок чтения последовательного файла: открытие файла (AssignFile, Reset), цикл чтения и обработки записей (Read), закрытие файла (CloseFile).

Файлы прямого доступа

Delphi не имеет специальных файлов с прямым доступом, а используется последовательный файл, допускающий прямой метод доступа. Общая схема создания файла: создать пустой последовательный файл с максимально допустимым числом пустых записей (форматирование файла), а затем создать последовательный файл и обращаться к записям по их порядковому номеру, используя процедуру ***Seek***(F,<номер записи>), а затем ***Read*** или ***Write*** соответственно.

Можно проверить, существует ли нужный файл, оператором **FileExists:**

```
if FileExists('FileName.ini')
  then Read(SaveF, SaveV);
```

При последовательном чтении из файла рано или поздно будет достигнут конец файла, и при дальнейшем чтении произойдёт ошибка. Проверить, не достигнут ли конец файла, можно оператором **EOF** (аббревиатура **E**nd **O**f **F**ile), который равен **true**, если указатель установлен в конец файла:

```
while (not EOF(SaveF)) do
  Read(SaveF, SaveV);
```

Оператор **Truncate(SaveF)** позволяет отсечь (стереть или, если хотите, удалить!) все записи файла, начиная от текущей позиции указателя, и до конца файла.

FilePos(F):longint - возвращает текущую позицию в нетекстовом файле F. Началу файла соответствует позиция 0.

FileSize(F):Integer - возвращает текущий размер нетекстового файла.

IOResult:Integer - возвращает код ошибки в последней выполненной операции ввода/вывода: 0 - ошибок нет, 2 - нет файла, 3 - ошибка в имени файла, 4 - много открытых файлов, 5 - файл недоступен, 100 - конец файла, 101 - диск переполнен, 106 - ошибка ввода-вывода. Используется при директиве компилятора {$I-} .

Rename(F;S) - изменяет имя внешнего файла F на S.

Преимущества типизированных файлов:

- компактное хранение данных, для внешнего представления которых требуется значительно больше памяти, чем для внутреннего (числа, записи);
- удобство работы с компонентами сложной структуры (за счет чтения-записи компонентов целиком).
- *для типизированных файлов возможна запись в файл, открытый для чтения*.

Нетипизированный файл (бестиповый)

Файл рассматривается как последовательность байтов, заканчивающаяся символом ***eof,*** и обрабатывается (считывается и записывается) блоками ***записей*** - совокупностей байтов заданной программистом длины.

Описание ***var <ф.п.>:file;***

Нетипизированный файл совместим с файлом любого типа. Это означает, что любой уже созданный файл *(текстовый, типизированный, нетипизированный)* может быть открыт и считан как нетипизированный, *по байтам; с другой стороны,* любое данное из оперативной памяти может быть по байтам записано в нетипизированный файл, *который затем можно открыть и обрабатывать как текстовый, типизированный или нетипизированный.*

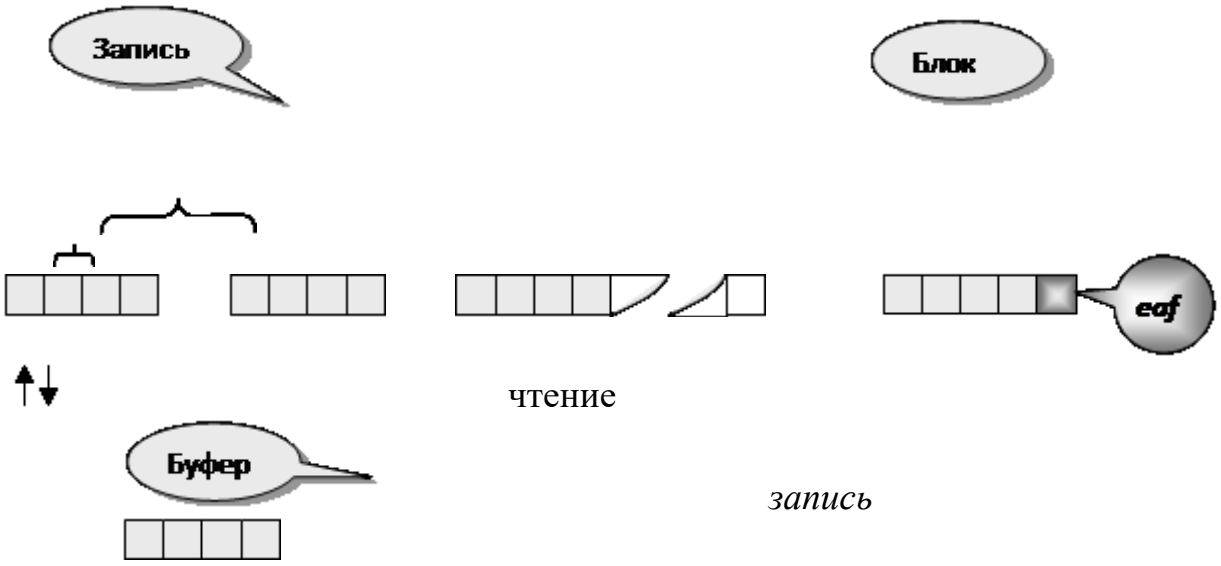
Открытие и закрытие файла производится аналогично типизированному файлу. Но в процедурах Reset и Rewrite вторым параметром типа можно указать длину записей в байтах.

Вместо процедур Read и Write используются процедуры:

BlockRead(F;var Buf;N:Integer[;var R:Integer]) **-** читает N записей из файла F в переменную B. Истинное количество считанных записей в R.

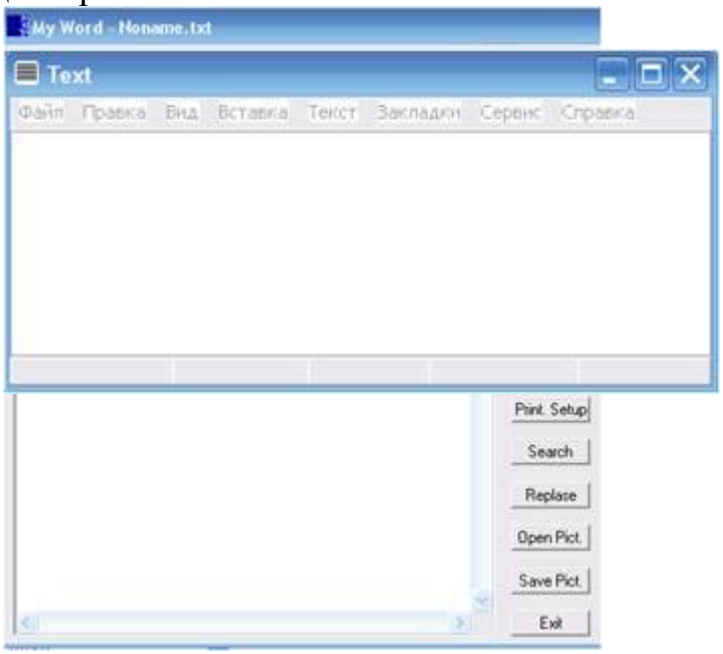
BlockWrite(F;var Buf;N:Integer[;var R:Integer]) **-** выводит N записей из переменной B в файл F. Истинное количество выведенных записей в R.

Общая структура нетипизированного файла и схема обработки



Задание 1.

Какие компоненты необходимы для создания следующих форм текстовых редакторов?



Задание 2. Соедините стрелками процедуры и функции с соответствующим типам файла:
Типизированный
Текстовый
Нетипизированный
Процедуры и функции: ***AssignFile, Read, Write, Seek, CloseFile, Erase, Truncate, FilePos, FileSize, Rename, BlockRead, BlockWrite, Reset, Rewrite, Append***

Задание: Вставь недостающие операторы:
Даны три целых числа.
Определить, имеется ли среди них хотя бы
одна пара равных между собой чисел.
Формат входных данных
Входной файл содержит три целых числа через пробел.
Формат выходных данных
Выведите 'YES' если это так, и 'NO' в противном случае.
РЕШЕНИЕ:

```
Program file;
    Var
    a,b,c:integer;
    f,g: text;
begin
    assign (f,'input.txt');
    reset (f);
    readln (f,a,b,c);
    close (f);
    assign (g,'output.txt');
    rewrite (g);
     if (a=b) or (b=c) or (c=a) then
     writeln (g,'YES')
     else
     writeln (g,'NO');
     close(g);
```

Впиши пропущенные слова в предложениях:

Компоненты для работы с файлами в Delphi – это компоненты **ListBox**, **ComboBox** и **Memo**, которые расположены на вкладке Standard Палитры компонентов.

Они читают и сохраняют своё содержимое, строки типа **String**, в файл текстового формата.

Каждая строка компонентов **ListBox** и **ComboBox** является объектом **Items[i]**, а **Memo** - **Lines[i]**, где **i** - номер строки, который отсчитывается от нуля.

Добавление строк в компоненты выполняется методами **Add** и **Insert**.

У компонента **ComboBox** дополнительно есть свойство **Text**, где находится вводимый текст.

ListBox1.Items.SaveToFile('Имя_файла**');** - **команда позволяет сохранить файл**

Для загрузки файла выполняется команда: **ListBox1.Items.LoadFromFile('**Имя_файла**');**

Диалоги открытия и сохранения файлов – это компоненты, позволяющие в работающей программе осуществлять выбор файлов. Они расположены на вкладке палитры компонентов **Dialogs.**

Компонент **OpenDialog** позволяет открыть в нашей программе стандартное Windows-окно открытия файла, компонент **SaveDialog** - окно сохранения.

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №5

Тема: Процедуры и функции

Цель: создавать проект с использованием процедур и функции.

Методические указания: Применение рекурсивных функций.

Рекурсия — это такой способ организации вычислительного процесса, при котором подпрограмма в ходе выполнения составляющих ее операторов обращается сама к себе.

Рассмотрим классический пример - вычисление факториала. Программа получает от компонента edinput целое число n и выводит в компонент lboutput значение N!, которое вычисляется с помощью рекурсивной функции Factorial.

При выполнении правильно организованной рекурсивной подпрограммы осуществляется многократный переход от некоторого текущего уровня организации алгоритма к нижнему уровню последовательно до тех пор, пока, наконец, не будет получено тривиальное решение поставленной задачи. В нашем случае решение при n = 0 тривиально и используется для остановки рекурсии.

```
procedure TfmExample.bbRunClick(Sender: TObject);
  function Factorial(N: Word): Extended;
  begin
    if N = 0 then
      Result: = 1 else
      Result := N * Factorial(N - 1)
  end;
var
  N: Integer;
begin
  try
    N := StrToInt(Trim(edinput.Text));
  except
    Exit; end;
  IbOutput.Caption := FloatToStr(Factorial(N))
end;
```

Рекурсивная форма организации алгоритма обычно выглядит изящнее итерационной и дает более компактный текст программы, но при выполнении, как правило, медленнее и может вызвать переполнение стека (при каждом входе в подпрограмму ее локальные переменные размещаются в организованной особым образом области памяти, называемой программным стеком).

Рекурсивный вызов может быть косвенным. В этом случае подпрограмма обращается к себе опосредованно, путем вызова другой подпрограммы, в которой содержится обращение к первой, например:

```
procedure A(i: Byte);
begin
  B(i);
end;
procedure B(j: Byte);
begin
  a(j);
end;
```

Если строго следовать правилу, согласно которому каждый идентификатор перед употреблением должен быть описан, то такую программную конструкцию использовать нельзя. Чтобы такого рода вызовы стали возможны, вводится опережающее описание:

```
procedure B(j: Byte); forward;
procedure A(i: Byte);
begin
  B(i);
end;
procedure B; begin
  A(j);
end;
```

Как видим, опережающее описание заключается в том, что объявляется лишь заголовок процедуры в, а ее тело заменяется стандартной директивой Forward. Теперь в процедуре, а можно использовать обращение к процедуре в - ведь она уже описана, точнее, известны ее формальные параметры, и компилятор может правильным образом организовать ее вызов. Обратите внимание: тело процедуры начинается заголовком, в котором уже не указываются описанные ранее формальные параметры

Задание: оформить задачу согласно всем этапам решения задач

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №6

Тема: Структуризация в программировании

Цель: изучить библиотеку подпрограмм.

Методические указания: Программирование модуля. Создание библиотеки подпрограмм.

Реализовать задачу своего варианта.

варианта	1-ая буква фамилии	задача
1	А, Б, В	Z=Sin(x+5y).
2	Г, Д, Е,	Z=Tan(3X+2Y).
3	Ж, З, И	Z=Sin(2x+3y).
4	К, Л , Ё	Z=Tan(2X+Y).
5	Н, О, П	Z=Cos(4X-2Y).
6	Р, С, Т	Z=Log(2X+Y+3).

7	У, Ф, Х	Создайте кнопку для возведения X в целочисленную степеньY
8	Ч, Ш, Щ	Z=X!
9	Э, Ю, Я, М	Z=Cos(6X-3Y+1).

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №7

Тема: Указател**и**

Цель: изучить решение задач с запомнинием промежуточных результат для будущих вычислений.

Методические указания: Использование указателей для организации связанных списков. Задача о стеке.

Задача, которую решает стек — запомнить промежуточный результат для будущих вычислений. тек (stack) — это контейнер, работающий по принципу "последний вошел, первый вышел" (last in, first out — LIFO). На рисунке показано представление стека, где метод Push() добавляет элемент, а метод Pop() — получает элемент, добавленный последним.

Задача. Стек с защитой от ошибок

Реализуйте структуру данных "стек". Напишите программу, содержащую описание стека и моделирующую работу стека, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

push n

Добавить в стек число n (значение n задается после команды). Программа должна вывести ok.

pop

Удалить из стека последний элемент. Программа должна вывести его значение.

back

Программа должна вывести значение последнего элемента, не удаляя его из стека.

size

Программа должна вывести количество элементов в стеке.

clear

Программа должна очистить стек и вывести ok.

exit

Программа должна вывести bye и завершить работу.

Перед исполнением операций back и pop программа должна проверять, содержится ли в стеке хотя бы один элемент. Если во входных данных встречается операция back или pop, и при этом стек пуст, то программа должна вместо числового значения вывести строку error.

Входные данные

Вводятся команды управления стеком, по одной на строке

Выходные данные

Программа должна вывести протокол работы стека, по одному сообщению на строке

Примеры

входные данные

```
size
push 1
size
push 2
size
push 3
size
exit
```

выходные данные

```
0
ok
1
ok
2
ok
3
bye
```

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №8

Тема: Основные принципы объектно-ориентированного программирования (ООП)

Цель: закрепить знание по решению задач с наследственными классами.

Методические указания: Создание наследованных классов.

Форма отчетности: файл **(**задачи).

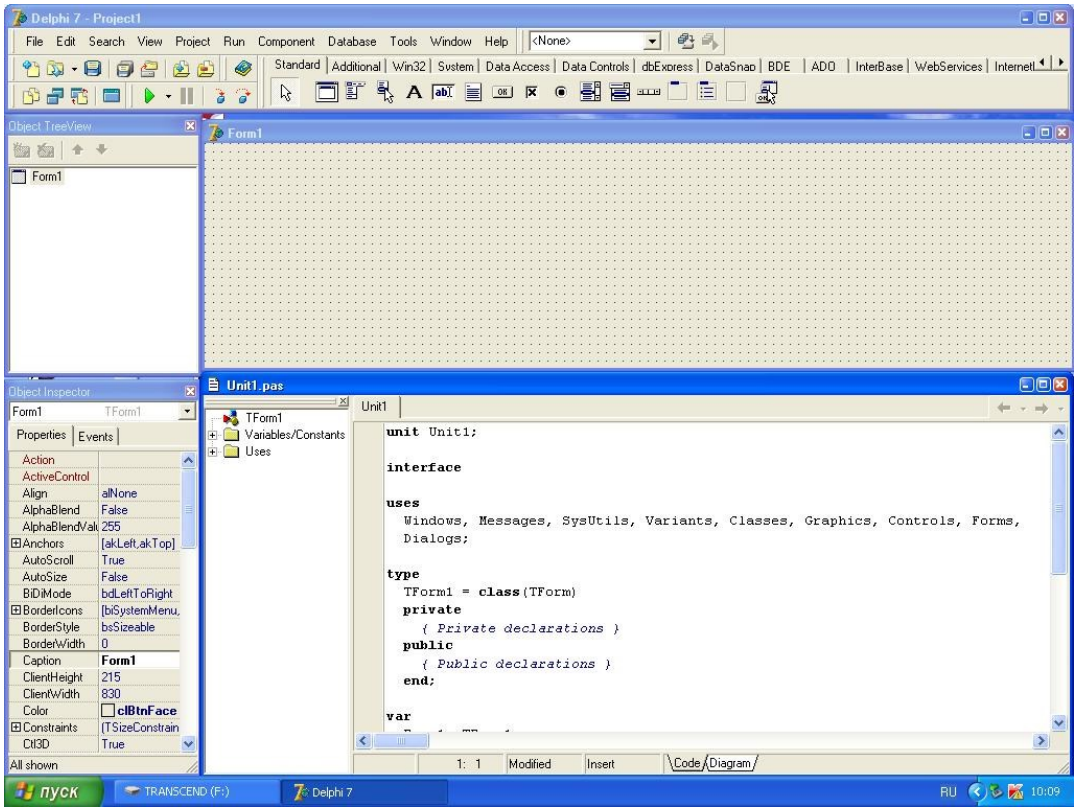
САМОСТОЯТЕЛЬНАЯ РАБОТА №9

Тема: Интегрированная среда разработчика

Цель: изучить интерфейс среды разработчика: характеристика, основные окна, инструменты, объекты.

Методические указания: Изучение интегрированной среды разработчика delphi.

Задание 1. На рисунке нужно отметить следующие объекты: главное окно, окно формы, окно кода программы, окно браузера кода,окно дерева объектов, окно инспектора объектов.



Задание 2. Обозначить инструментальные кнопки которые открывают быстрый длоступ к наиболее важным командам главного меню.

кнопка	описание	Эквивалентно команде
		
		
		
		
		
		
		
		
		
		
		
		

Форма отчетности: файл.

САМОСТОЯТЕЛЬНАЯ РАБОТА №10

Тема: Визуальное событийно-управляемое программирование

Цель: закрепить знание по средствам прорисовки контурного изображения.

Методические указания: Построение графика по вариантам.

Вариант	функция
1	$y=\cos^2 x$
2	$y=\sin^2 x$
3	$y=\cos(2x)$
4	$y=\sin(3x/2)$
5	$y=\cos(5x/7)$
6	$y=\sin x/\cos 5x$
7	$y=5/(7x+1)$.
8	$y=x^2+3x+7$
9	$y= x^4+2x^3+7$
10	$y= x^3+7x+10$

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №11

Тема: Разработка оконного приложения

Цель: закрепить по разработке многооконных приложений.

Методические указания: Разработка оконного приложения с несколькими формами по вариантам.

Вставить таблицу

Форма отчета: файл **(**задачи).

САМОСТОЯТЕЛЬНАЯ РАБОТА №12

Тема: Этапы разработки приложений

Цель: закрепить по тестированию и отладке приложений.

Методические указания: Тестирование и отладка приложений.

Протестировать любую задачу из задачника Д.10 раздел IF согласно этапам решения задач.

Форма отчета: файл **(**задачи).

3. КРИТЕРИИ ОЦЕНКИ ВЫПОЛНЕНИЯ САМОСТОЯТЕЛЬНЫХ РАБОТ

Оценки «5» (отлично) заслуживает студент, обнаруживший при выполнении заданий всестороннее, систематическое и глубокое знание учебно - программного материала, учения свободно выполнять профессиональные задачи с всесторонним творческим подходом, обнаруживший познания с использованием основной и дополнительной литературы, рекомендованной программой, усвоивший взаимосвязь изучаемых и изученных дисциплин в их значении для приобретаемой специальности, проявивший творческие способности в понимании, изложении и использовании учебно- программного материала, проявивший высокий профессионализм, индивидуальность в решении поставленной перед собой задачи, проявивший неординарность при выполнении практических заданий.

Оценки «4» (хорошо) заслуживает студент, обнаруживший при выполнении заданий полное знание учебно- программного материала, успешно выполняющий профессиональную задачу или проблемную ситуацию, усвоивший основную литературу, рекомендованную в программе, показавший систематический характер знаний, умений и навыков при выполнении теоретических и практических заданий по дисциплине «Информатика».

Оценки «3» (удовлетворительно) заслуживает студент, обнаруживший при выполнении практических и теоретических заданий знания основного учебно-программного материала в объеме, необходимом для дальнейшей учебной и профессиональной деятельности, справляющийся с выполнением заданий, предусмотренных программой, допустивший погрешности в ответе при защите и выполнении теоретических и практических заданий, но обладающий необходимыми знаниями для их устранения под руководством преподавателя, проявивший какую-то долю творчества и индивидуальность в решении поставленных задач.

Оценки «2» (неудовлетворительно) заслуживает студент, обнаруживший при выполнении практических и теоретических заданий проблемы в знаниях основного учебного материала, допустивший основные принципиальные ошибки в выполнении задания или ситуативной задачи, которую он желал бы решить или предложить варианты решения, который не проявил творческого подхода, индивидуальности.

4. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

4.1 Основные электронные издания:

О-1. Семакин, И.Г. Основы алгоритмизации и программирования: учебник / И.Г. Семакин, А.П. Шестаков. – 6-е изд., стер. – М.: Образовательно-издательский центр «Академия», 2024. – 304 с. – URL: https://academia-moscow.ru/catalogue/5546/768351/. – Режим доступа: Электронная библиотека «Academia-moscow». – Текст: электронный.

О-2. Трофимов, В. В. Основы алгоритмизации и программирования : учебник для среднего профессионального образования / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — 4-е изд. — Москва: Издательство Юрайт, 2024. — 119 с. — (Профессиональное образование). — ISBN 978-5-534-17498-4. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/539994 (дата обращения: 02.05.2024).

4.2 Дополнительные источники:

Д-1. Голицына, О.Л. Попов И.И. Основы алгоритмизации и программирования: учебник. – М.: ИД "ФОРУМ"-ИНФРА-М, 2006. – 432 с.

Д–2. Голицына, О.Л. Попов И.И. Основы алгоритмизации и программирования: учебник. – М.: ИД "ФОРУМ"-ИНФРА-М, 2004. – 432 с.

Д–3. Колдаев, В.Д. Основы алгоритмизации и программирования: учебник. – М.: ИД "ФОРУМ"-ИНФРА-М, 2009. – 416 с.

Д–4. Канцедал, С.А. Алгоритмизации и программирования: учебник. – М.: ИД "ФОРУМ"-ИНФРА-М, 2008. – 352 с.

Д–5. Голицына, О.Л., Партыка Т.Л., Попов И.И. Программное обеспечение: Учебное пособие – М.: ИД "ФОРУМ"-ИНФРА-М, 2006. – 432 с.

Д–6. Голицына, О.Л., Партыка Т.Л., Попов И.И. Языки программирования: Учебное пособие – М.: ИД "ФОРУМ", 2008. – 400 с.

Д–7. Голицына, О.Л., Попов И.И., Попов И.И. Программирование на языках высокого уровня: Учебное пособие – М.: ИД "ФОРУМ", 2008. – 496 с.

Д–8. Голицына О.Л., Партыка Т.Л., Попов И.И. Программное обеспечение: Учебное пособие – М.: ИД "ФОРУМ"-ИНФРА-М, 2008. – 432 с.

Д–9. Семакин, И.Г. Основы программирования: Учебное пособие. – М.: Академия, 2003. – 432 с.

Д–10. Мишенин А.И. Сборник задач по программированию: учебное пособие – М.: Инфра-М, 2009. – 224 с.

Д–11. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения: учебное пособие – М: ИД "ФОРУМ"-ИНФРА-М, 2009. – 400 с.

Д–12. Шамис, В.А. C++ Builder 4. Техника визуального программирования – М: Нолис, 2000. – 656 с.

Д–13. Вальпа О.Д. C++Builder. Экспресс-курс. - М: БХВ -Петербург, 2006. – 224 с.

Д–14. Культи, Н. C++Builder в задачах и примерах– М: БХВ -Петербург, 2007. – 336 с.

Д–15. Пахомов, Б. C/C++ и Borlanl C++Builder . Для начинающих– М: БХВ -Петербург, 2007. – 640 с.

Д–16. Архангельский, А.Я. Приемы программирования C++Builder 6 и 2006 – М: Бином-Пресс, 2006. – 992 с.

Д–17. Лаптев, В.В. C++.Объектно-ориентированное программирование: учебное пособие – М: Питер, 2008. – 464 с.

Д–18. Павловская, Т.А., Щупак Ю.А. C++.Объектно-ориентированное программирование. Практикум: Практикум – М: Питер, 2008. – 265 с.

Д–19. Карпов, Б., Баранова Т. C++.: Справочник – М: Питер, 2005. – 381 с.

Д–20. Пол Айра Объектно-ориентированное программирование. C++ – М: Бином, 1999. – 462 с.

Д–21. Иванова, Г.С., Ничушкина Т.Н., Пугачев Е.К. Объектно-ориентированное программирование: учебник – М: МГТУ им Баумана, 2003. – 368 с.

Д–22. Павловская, Т.А., Щупак Ю.А. ПАСКАЛЬ. Программирование на языке высокого уровня: учебник – М: Питер, 2004. – 368 с.

Д-23. Начало программирования [Электронный ресурс]. – Режим доступа: www.url: https://pas1.ru/. – 02.05.2024.

ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ, ВНЕСЕННЫХ В МЕТОДИЧЕСКИЕ УКАЗАНИЯ

№ изменения, дата внесения, № страницы с изменением	
Было	**Стало**
Основание: **Подпись лица, внесшего изменения**	