

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тульский государственный университет»

Интернет-институт

ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

«Программирование»

Семестр 2

Вариант 3

Выполнил: студент гр. ИБ262521-ф

Артемов Александр Евгеньевич

Проверил: канд. техн. наук, доц.

Сафронова Марина Алексеевна

Тула 2023

Лабораторная работа № 1

Название: Освоение интегрированной среды программирования Free Pascal.

Цель работы: Целью работы является изучение среды Free Pascal.

Выполнение лабораторной работы.

1. Изучены теоретические положения работы. Установочный файл fpc-3.2.2.win32.and.win64.exe скачан со страницы <https://www.freepascal.org/download.html> и установлен по пути C:\FPC\3.2.2.

2. Описание команд главного меню и сочетаний горячих клавиш среды Free Pascal:

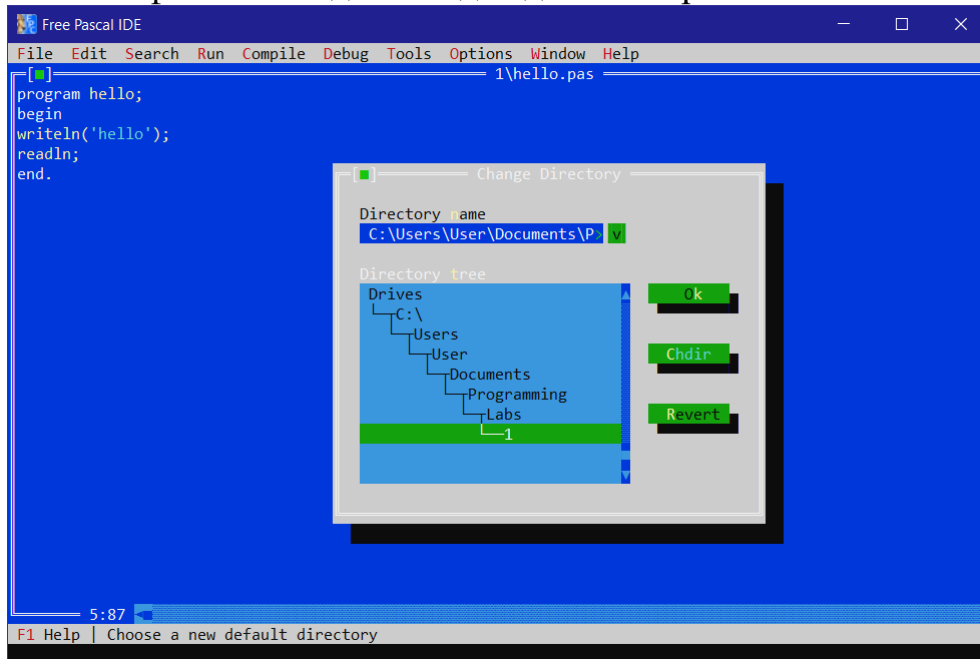
Меню	Пункт меню	Сочетание клавиш	Назначение
File			Работа с файлами
	New	F3	Создание нового файла
	Open	F2	Открыть файл
	Save		Сохранение файла, находящегося в редакторе
	Save as		Сохранение файла, находящегося в редакторе, под новым именем или в новом месте
	Change dir		Отображает текущий каталог и позволяет назначить текущим другой дисковод и каталог
	Exit	Alt+X	Выход
Edit			Редактор текста (выделение, копирование, вставка и пр.)
Search			Поиск и замена текста
Run			Выполнение
	Run	Ctrl+F9	Запуск программы на выполнение
	Step over	F8	Пошаговое выполнение. При обращении к процедуре она будет выполняться как одна команда
	Trace into	F7	Трассировать внутрь. При обращении к процедуре будет выполняться каждый оператор процедуры.
	Goto cursor	F4	Выполнять программу до курсора
Compile			Компиляция
	Compile	Alt+F9	Компилировать в объектный файл
	Make	F9	Создать выполняемый файл
	Target		Устанавливает платформу

			(выходную операционную систему), для которой будет компилироваться
	Compiler messages	F12	Сообщения компилятора
Debug			Отладка
	User screen	Alt+F5	Просмотр результата работы программы
	Breakpoint	Ctrl+F8	Установить точку прерывания в строке в месте расположения курсора.
	Add watch	Ctrl+F7	Ввод контролируемой переменной в окно Watches
	Watches		Отобразить список выражений окна Watches
Tools			Инструменты
Options			Опции
	Directories		Пункты этого меню позволяют указать Free Pascal, где ему следует искать файлы для компиляции, компоновки, где размещать выходные файлы и где искать файлы конфигурации, помощи (help)
	Save		Сохранить опции по умолчанию
	Save as		Сохранить текущие опции
Window			Работа с окнами
	Zoom	F5	Расширяет активное окно на весь экран. Повторное нажатие клавиши F5 переводит экран в режим наложения окон
	Next	F6	Активизировать следующее окно в списке окон
	Previous	Shift+F6	Активизировать предыдущее окно в списке окон
Help			Помощь (справка)
	Contents		Оглавление справки
	Index	Shift+F1	Переход к справке индексов
	Topic search	Ctrl+F1	Переход к теме, связанной с высвеченным текстом

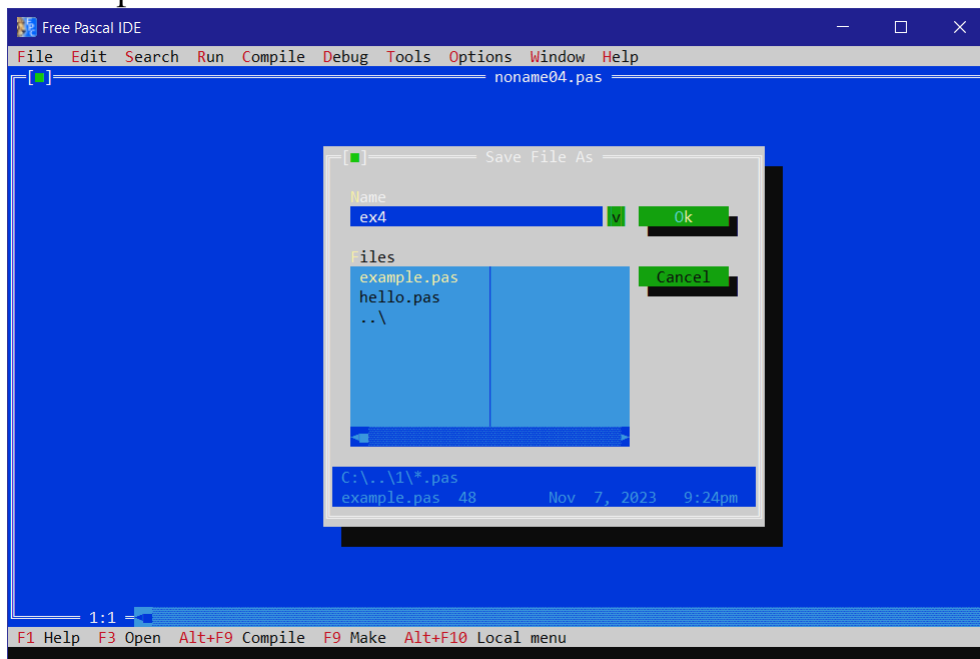
Кроме того, сами меню вызываются сочетанием Alt+буква, с которой начинается название меню, например, для вызова меню File сочетание будет Alt+F, а для Options – Alt+O. При редактировании текста полезно использовать сочетания Ctrl+Y для удаления строки и Ctrl+T для удаления слова.

3. Описание технологии сохранения программ в среде Free Pascal.

Перед началом работы с новым проектом рекомендую указывать рабочую папку командой File -> Change dir., т.е. папку, в которой будут храниться все файлы исходных кодов данного проекта.



Создаем новый файл командой File -> New и вводим текст программы. Для сохранения программы выполняем команду File -> Save или нажимаем F2, вводим имя файла и нажимаем ОК.

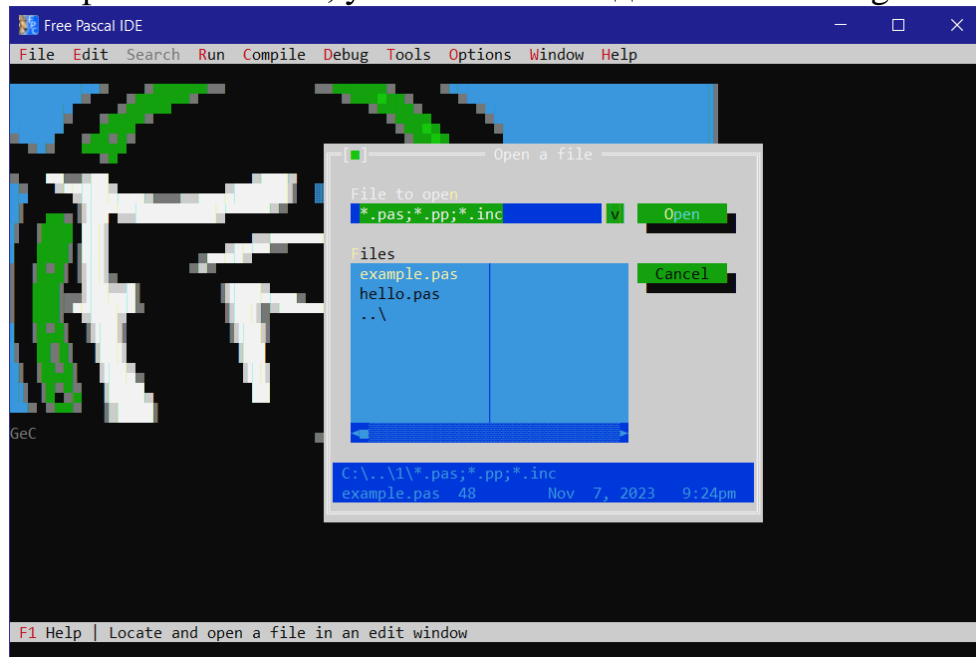


В процессе редактирования программы периодически сохраняем изменения, нажимая F2. При необходимости сохранить файл программы под другим именем или в другом месте выполняем команду File -> Save as...

4. Описание технологии загрузки файлов в окно редактирования среды Free Pascal.

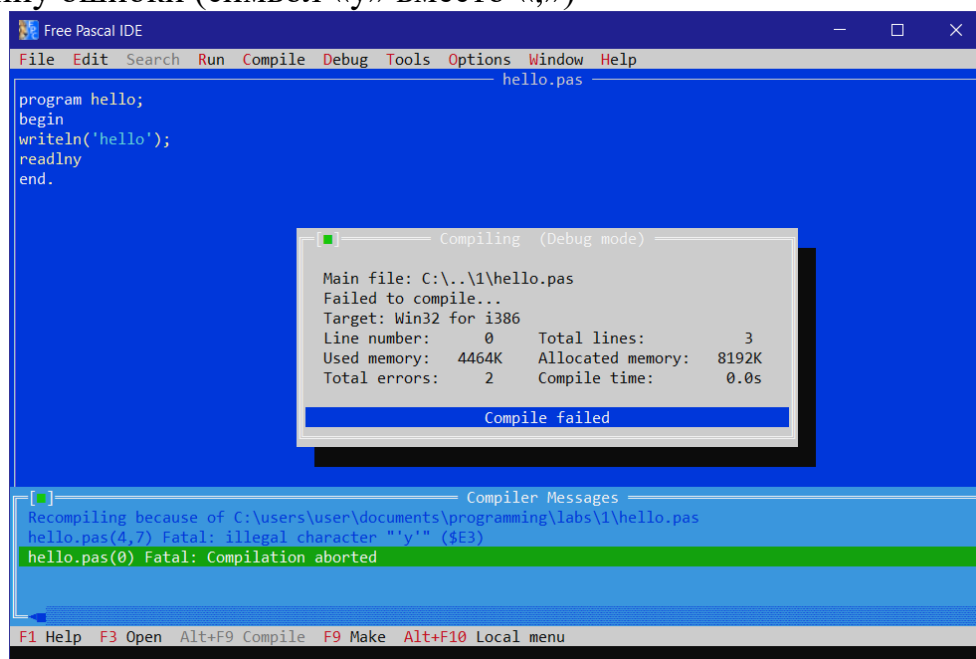
Для открытия файла программы выполняем команду File -> Open.. или нажимаем F3, выбираем файл и жмем кнопку Open. По умолчанию будут

показаны файлы из папки, указанной командой File -> Change dir..

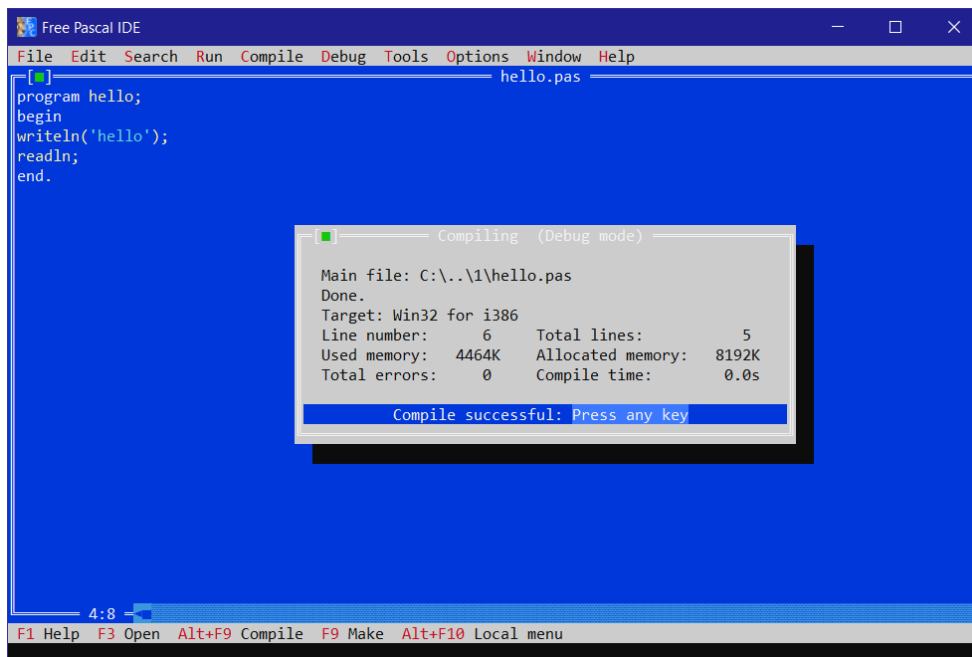


5. Описание окна получения результатов после запуска программы на выполнение.

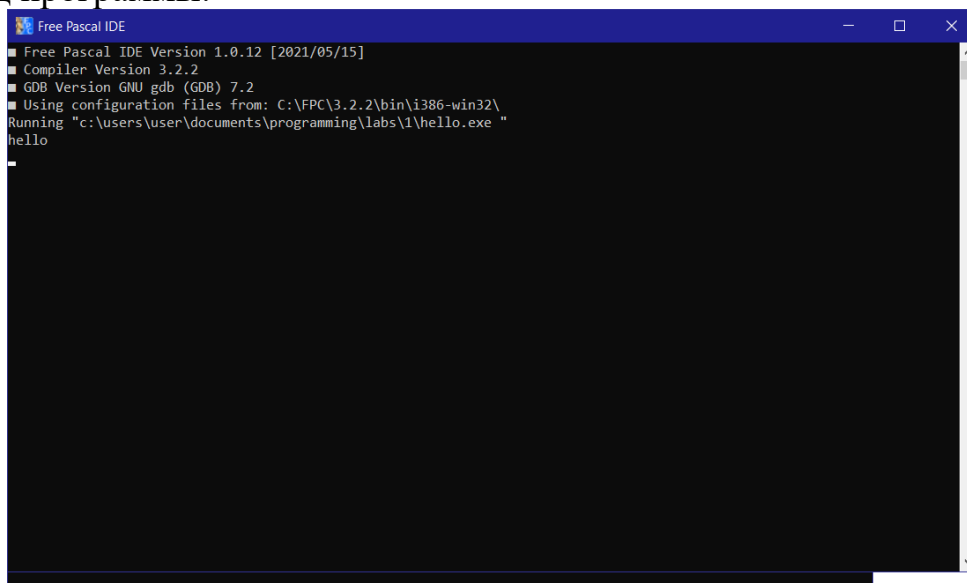
Запуск программы выполняем командой Run -> Run или Ctrl+F9. В тексте программы специально допущена ошибка, о чем говорит сообщение «Compile failed». Так же компилятор указывает в нижней части окна место и причину ошибки (символ «у» вместо «;»)



При устранении ошибки и сохранении файла выполняем команду Compile -> Compile или Alt+F9. Компилятор сообщает об успешной компиляции:



Чтобы выполнить компиляцию с последующим запуском программы на выполнения применяется команда Run -> Run или Ctrl+F9, после которой идет вывод программы:



Как видно на снимке экрана выше, компилятор выводит информацию об используемой IDE и ее версии, версии компилятора, версии отладчика, используемой конфигурации IDE и полный путь к скомпилированному исполняемому файлу.

В данной работе я ознакомился с IDE FreePascal, меню и сочетаниями клавиш, способами создания файлов исходного кода программ, их сохранения и загрузки в редактор, способами компиляции и выполнения программ. Данная IDE полностью функциональна, но, ввиду выполнения в DOS-режиме, сводит олдскулы))

Ответы на контрольные вопросы.

1. Что такое процесс компиляции?

Процесс компиляции – это выполнение трансляции исходного кода

программы в объектный файл и компоновки (из сгенерированных на фазе трансляции) объектных файлов в исполняемый файл.

2. Как пользователь узнает о том, что компиляция прошла успешно?

Обычно компилятор сообщает пользователю о результатах после выполнения своей работы. Так же, при выполнении блока команд типа Компиляция -> Запуск, пользователь видит, что успешно скомпилированная программа запущена, либо сообщение об ошибке компиляции. В FreePascal IDE сообщения об ошибках компиляции можно просмотреть, выполнив команду Compile -> Compiler messages, либо нажав клавишу F12.

3. Какие операторы языка Паскаль Вам известны?

Оператор	Пояснение
begin	Начало блока
end	Конец блока
:=	Оператор присваивания
if ... then ... else ...	Условный оператор ветвления (2 ветки)
case ... of	Оператор ветвления (несколько веток)
read, readln	Ввод данных
write, writeln	Вывод данных
for ... to ... do ...	Цикл с возрастающим параметром
for ... downto ... do ...	Цикл с убывающим параметром
while ... do ...	Цикл с предусловием
repeat ... until ...	Цикл с постусловием

4. Какие сочетания клавиш можно использовать в среде Free Pascal?

Все сочетания клавиш, соответствующие командам меню, отображаются непосредственно в меню, можно подсмотреть и запомнить для дальнейшего использования. Чаще всего, на мой взгляд, используются команды сохранения текста F2, запуска на выполнения Ctrl+F9, и, возможно, пошагового выполнения при отладке F8.

5. Как сохранить набранный текст программы в файл?

Выполнить команду File -> Save, либо нажав F2. Для сохранения под другим именем выполняется команда File -> Save as...

6. Как загрузить набранный текст программы из файла в окно редактирования?

Выполнить команду File -> Open, либо нажав F3, в появившемся окне выбрать папку и файл программы, нажать кнопку Open.

Лабораторная работа № 2

Название: Составление простейших схем и программ в среде программирования Free Pascal (изучение структуры «следование»).

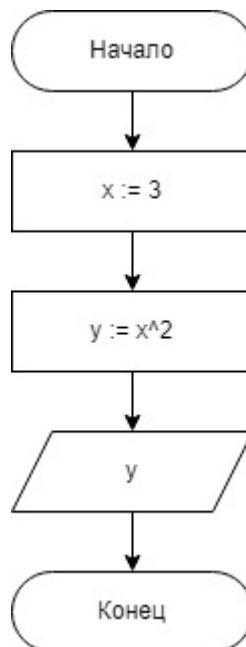
Цель работы: Получение навыков по составлению и отладке простейших схем и программ на языке Паскаль (изучение последовательности или следования).

Выполнение лабораторной работы.

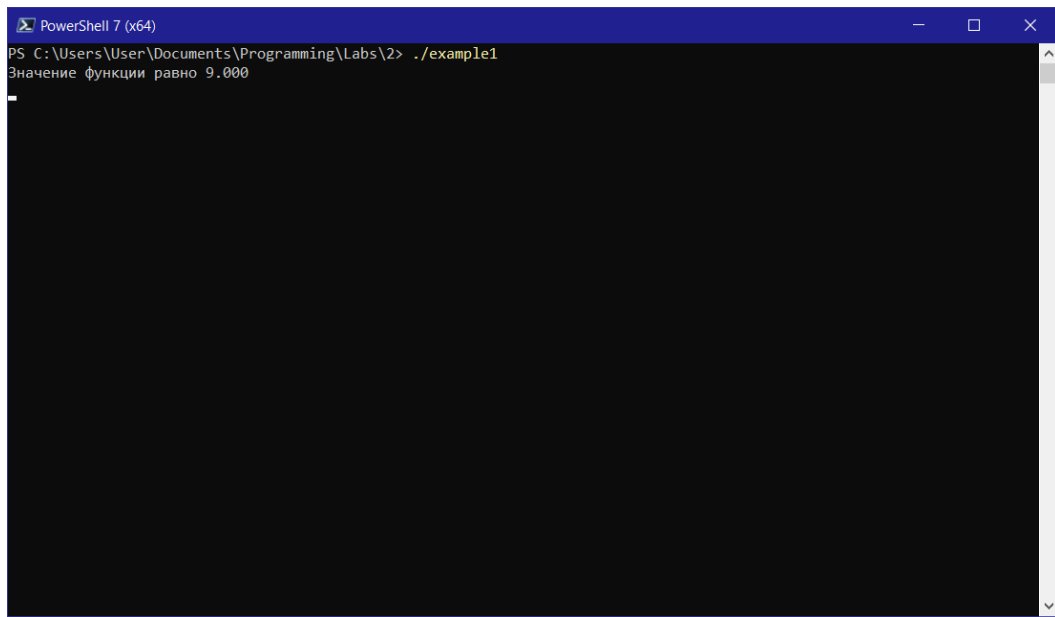
1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 1 (файл 2/example1.pas):

```
program example1;  
var x, y: real;  
begin  
  x := 3;  
  y := sqr(x);  
  writeln('Значение функции равно ', y:0:3);  
  readln;  
end.
```

Блок-схема алгоритма программы:



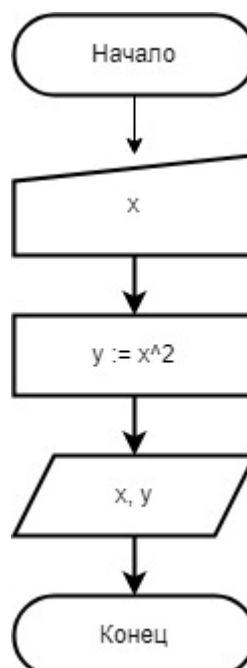
Результат выполнения программы (запуск произведен через PowerShell):



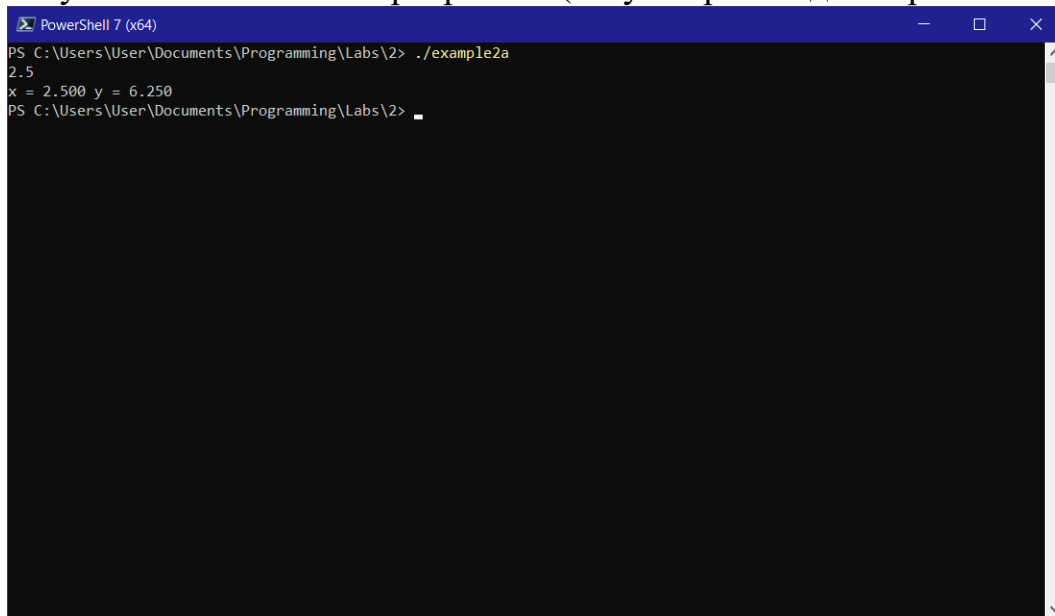
3. Исходный код программы в соответствии с примером 2а (файл 2/example2a.pas):

```
program example2a;  
var x, y: real;  
begin  
  read(x);  
  y := sqr(x);  
  writeln('x = ', x:0:3, ' y = ', y:10:5);  
  readln;  
end.
```

Блок-схема алгоритма программы:



Результат выполнения программы (запуск произведен через PowerShell):

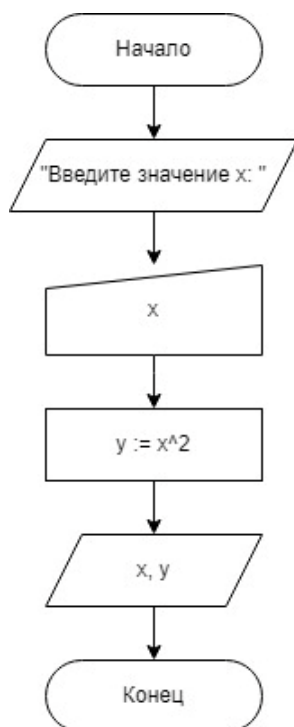


```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\2> ./example2a
2.5
x = 2.500 y = 6.250
PS C:\Users\User\Documents\Programming\Labs\2> _
```

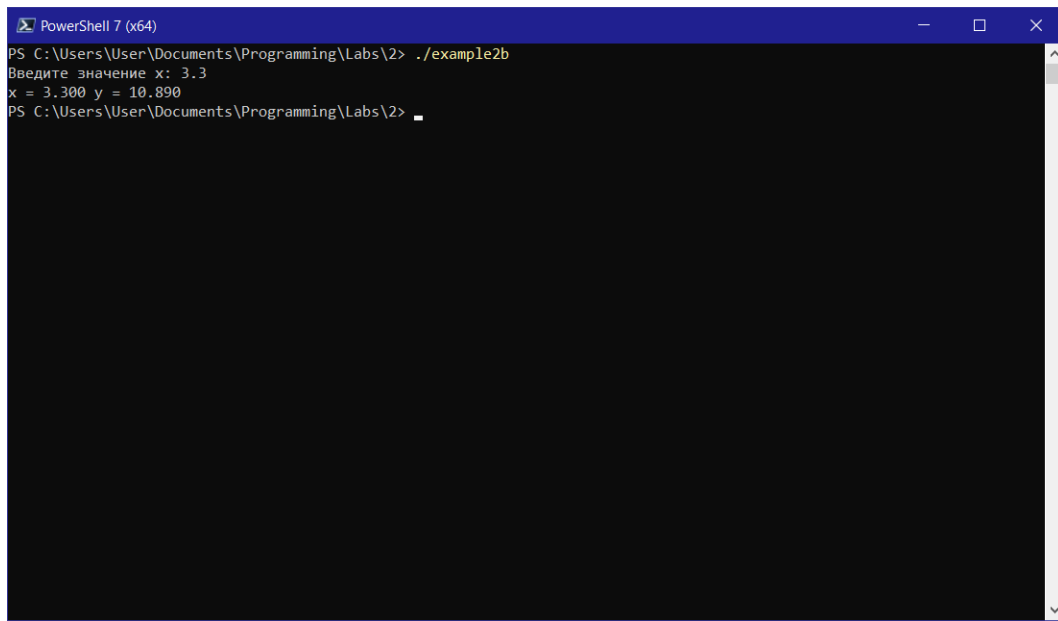
Исходный код программы в соответствии с примером 2b (файл 2/example2b.pas):

```
program example2b;
var x, y: real;
begin
write('Введите значение x: ');
read(x);
y := sqr(x);
write('x = ', x:0:3, ' y = ', y:0:3);
readln;
end.
```

Блок-схема алгоритма программы:



Результат выполнения программы (запуск произведен через PowerShell):



```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\2> ./example2b
Введите значение x: 3.3
x = 3.300 y = 10.890
PS C:\Users\User\Documents\Programming\Labs\2> _
```

Ответы на контрольные вопросы.

1. Как в среде Free Pascal при написании программ указываются операторные скобки?

Операторные скобки для объединения нескольких команд в один блок начинаются с оператора `begin`, далее – команды, по окончании блока – оператор `end`. В основном блоке программы после оператора `end` ставится точка, во внутренних блоках – точка с запятой.

2. Какие функции выполняет оператор ввода?

Оператор ввода `read(x)` помещает значение с ввода командной строки в переменную `x` без перехода курсора на следующую строку. Оператор ввода `readln(x)` помещает значение с ввода командной строки в переменную `x` с переходом курсора на следующую строку.

3. Какие функции выполняет оператор вывода?

Оператор вывода `write(x)` выводит значение переменной `x` в командную строку без перехода курсора на следующую строку. Оператор вывода `writeln(x)` выводит значение переменной `x` в командную строку с переходом курсора на следующую строку. Оба оператора могут принять несколько аргументов разного типа для вывода.

4. Как в среде Free Pascal возвести число в квадрат?

Число возводится в квадрат умножением самого на себя «`y := x*x;`», либо с использованием функции `sqr()` – «`y := sqr(x);`».

5. Как ограничить количество цифр после запятой вещественного числа в среде Free Pascal?

Число знаков вещественного числа ограничивается при указании аргумента оператора вывода, например, `write(x:3:5)` выводит число `x` с тремя цифрами в целой части и с пятью цифрами в дробной части.

Лабораторная работа № 3

Название: Вычисление значений функций в интегрированной среде программирования Free Pascal (изучение структуры «ветвление»).

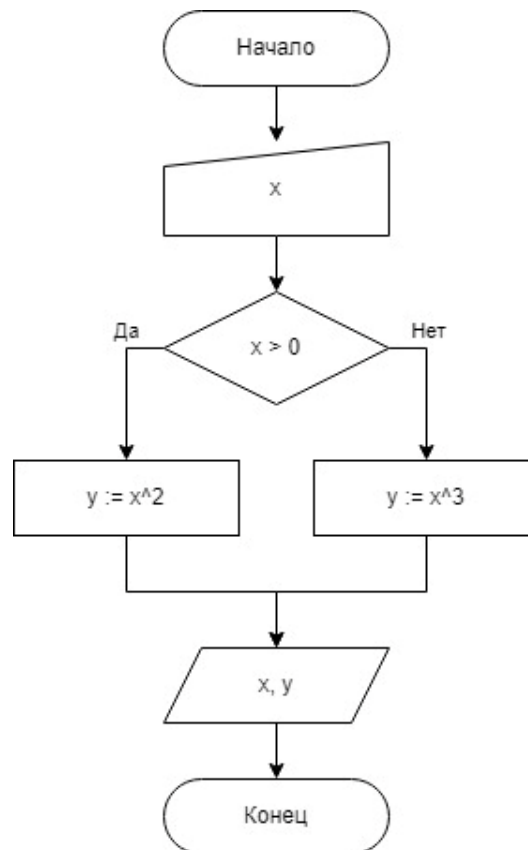
Цель работы: Получение навыков по вычислению значений функций при помощи написания программ на языке Паскаль (изучение ветвления или выбора).

Выполнение лабораторной работы.

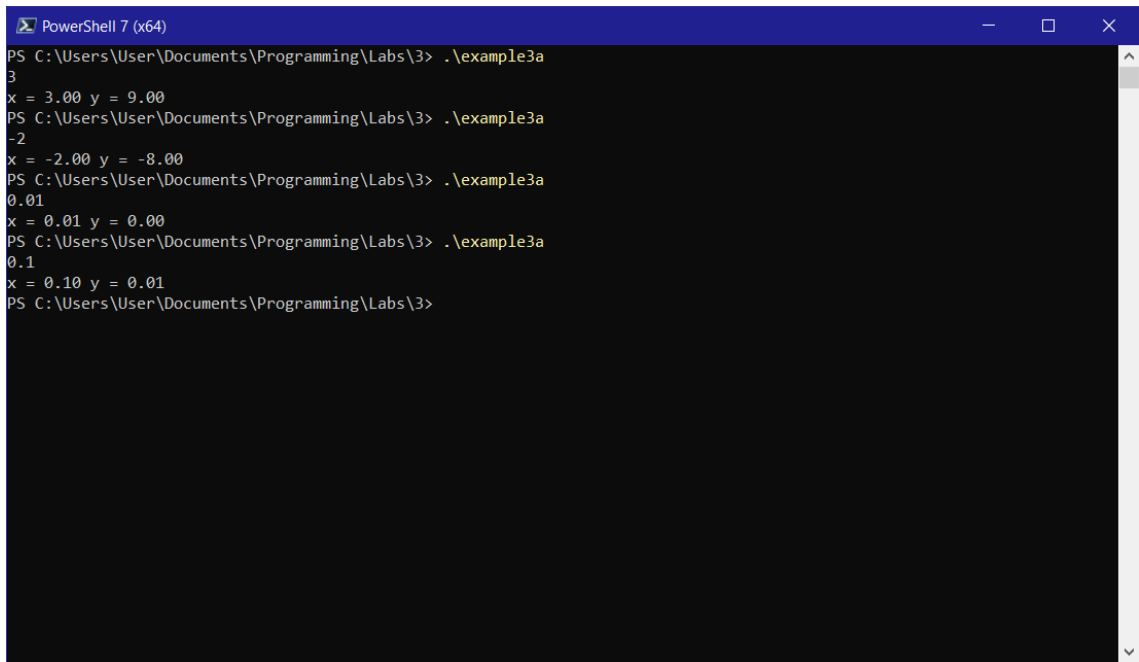
1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 3а (файл 3/example3a.pas):

```
program example3a;  
var x, y: real;  
begin  
  read(x);  
  if (x>0) then y := sqr(x)  
  else y := sqr(x) * x;  
  writeln('x = ', x:3:2, ' y = ', y:3:2);  
end.
```

Блок-схема алгоритма программы:



Результат выполнения программы (запуск произведен через PowerShell):



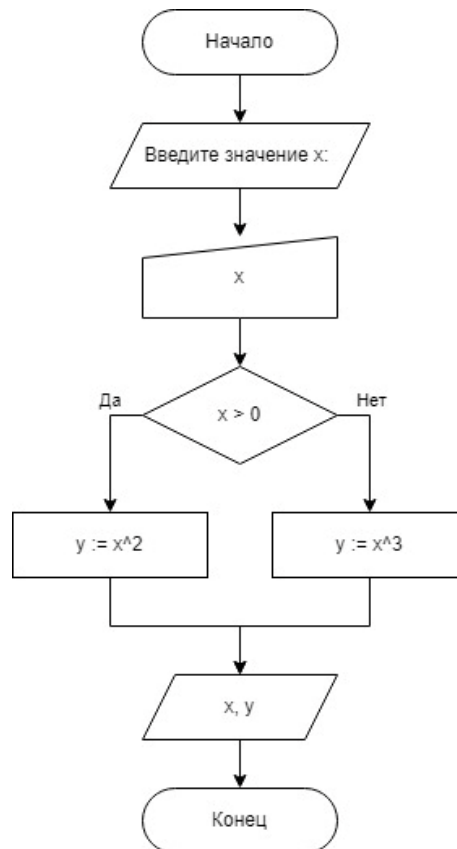
```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\3> .\example3a
3
x = 3.00 y = 9.00
PS C:\Users\User\Documents\Programming\Labs\3> .\example3a
-2
x = -2.00 y = -8.00
PS C:\Users\User\Documents\Programming\Labs\3> .\example3a
0.01
x = 0.01 y = 0.00
PS C:\Users\User\Documents\Programming\Labs\3> .\example3a
0.1
x = 0.10 y = 0.01
PS C:\Users\User\Documents\Programming\Labs\3>
```

Показано выполнение программы несколько раз с различными значениями x : положительным, отрицательным и десятичным. Все вычисления выполняются программой верно, с заданной точностью (два знака после запятой).

3. Исходный код программы в соответствии с примером 3b (файл 3/example3b.pas):

```
program example3b;
var x, y: real;
begin
write('Введите значение x: ');
read(x);
if (x>0) then y := sqr(x)
else y := sqr(x) * x;
write('x = ', x:3:2, ' y = ', y:3:2);
end.
```

Блок-схема алгоритма программы:



Результат выполнения программы (запуск произведен через PowerShell):

```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\3> .\example3b
Введите значение x: 3
x = 3.00 y = 9.00
PS C:\Users\User\Documents\Programming\Labs\3> .\example3b
Введите значение x: -5
x = -5.00 y = -125.00
PS C:\Users\User\Documents\Programming\Labs\3> .\example3b
Введите значение x: 1.3
x = 1.30 y = 1.69
PS C:\Users\User\Documents\Programming\Labs\3> .\example3b
Введите значение x: -0.4
x = -0.40 y = -0.06
PS C:\Users\User\Documents\Programming\Labs\3>
```

The screenshot shows the PowerShell console running the program multiple times with different inputs. The output for each run shows the calculated values of x and y, matching the logic of the flowchart.

Показано выполнение программы несколько раз с различными значениями x: положительным, отрицательным, дробным и отрицательным дробным. Все вычисления выполняются программой верно, с заданной точностью (два знака после запятой).

Ответы на контрольные вопросы.

1. Что такое ветвление или выбор?

Это условный оператор, который выполняет заданные команды в зависимости от истинности некоторого условия.

2. Какие виды ветвления или выбора существуют?

При использовании ветвления вида «if (условие) then команда_1» выполняется команда 1, но только при истинности условия.

При использовании ветвления вида «if (условие) then команда_1 else команда_2» выполняется команда 1, условие истинно, или команда_2, если условие ложно.

3. Для чего нужен оператор if?

Ключевое слово if говорит о начале использования условного оператора. После ключевого слова if в скобках указывается условие, например, if (x > 0)

4. Какие функции выполняет then?

После ключевого слова then задается команда или набор команд, которые выполняются при истинности условия.

5. Какие функции выполняет else?

После ключевого слова else задается команда или набор команд, которые выполняются если условие ложно.

6. Какими особенностями обладает структура if then else?

Структура if then else поддерживает многократную вложенность в саму себя, что в принципе позволяет создать дерево ветвления не ограниченной сложности, что с другой стороны усложняет отладку программы и общую семантику кода, так называемая «спаггетификация кода».

Лабораторная работа № 4

Название: Вычисление значений функций при различных переменных в интегрированной среде программирования Free Pascal (изучение структуры «цикл»).

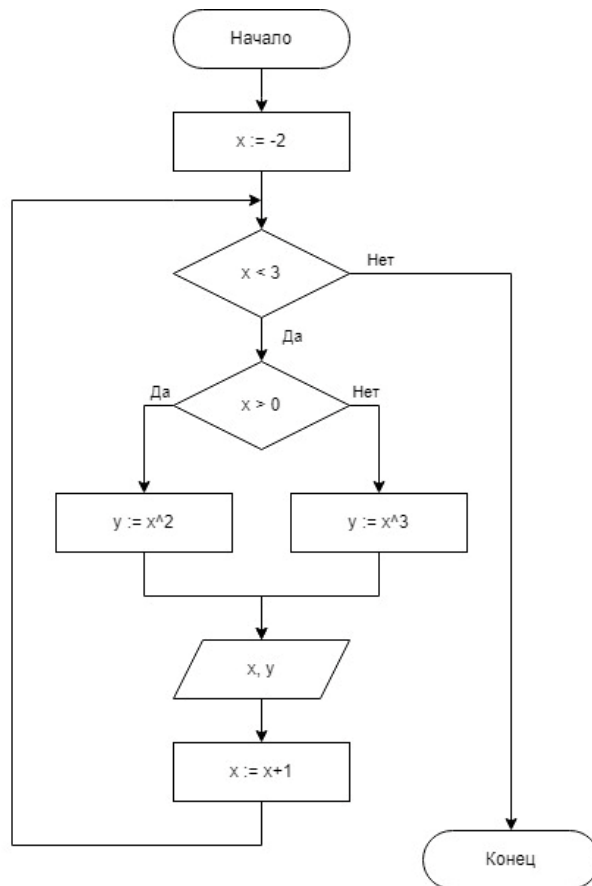
Цель работы: Получение навыков вычисления значений функций при различных переменных на языке Паскаль (изучение циклов).

Выполнение лабораторной работы.

1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 4 (файл 4/example4a.pas):

```
program example4a;
var x, y: real;
begin
  writeln(' x      ', ' y      ');
  x := -2;
  while (x <= 3) do begin
    if (x > 0) then y := sqr(x)
    else y := sqr(x) * x;
    if (x >= 0) then write(' ');
    write(x:1:3);
    if (y >= 0) then write(' ')
    else write(' ');
    writeln(y:1:3);
    x := x + 1;
  end;
end.
```


Блок-схема алгоритма программы:



3. Результат выполнения программы (запуск произведен через PowerShell):

```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\4> ./example4
x      y
-2.000 -8.000
-1.000 -1.000
0.000  0.000
1.000  1.000
2.000  4.000
3.000  9.000
PS C:\Users\User\Documents\Programming\Labs\4>
```

The screenshot shows a PowerShell window with the command `./example4` executed. The output is a table with two columns, `x` and `y`, showing values from -2.000 to 3.000 and their corresponding squares.

Ответы на контрольные вопросы.

1. Что такое цикл?

Действия, неоднократно повторяющиеся в программе при различных переменных, называются циклом.

2. Какие виды циклов существуют?

Цикл с предусловием, цикл с постусловием, цикл с параметром.

3. Что такое табуляция?

Табуляция – это табличная группировка вычисляемых данных.

4. Как в среде Free Pascal реализовать цикл «пока»?

Цикл с предусловием «пока» реализован при помощи ключевых слов `while` (условие) `do` команда_1. Несколько команд обертываются в скобки `begin ... end;`. Необходимо самостоятельно контролировать условие окончания цикла.

5. Как в среде Free Pascal реализовать цикл «до тех пор»?

Цикл с постусловием «до тех пор» реализован при помощи ключевых слов `repeat` команда_1; команда_2; `until` (условие). Необходимо самостоятельно контролировать условие окончания цикла.

6. Как в среде Free Pascal реализовать цикл «для»?

Цикл «для» реализован при помощи ключевых слов `for ... to ... do ...` с увеличивающимся индексом, например, `for i := 1 to 10 do` команда_1(i); или с уменьшающимся индексом `for i := 10 downto 1 do` команда_1(i);. Цикл в обоих случаях выполняется 10 раз, то есть выполняется команда_1 с параметром i, контролировать выход из цикла необязательно, если в теле цикла вручную не изменяется значение индекса i. Несколько команд обертываются в скобки `begin ... end;`.

Лабораторная работа № 5

Название: Вычисление значений функций при различных переменных, введенных с клавиатуры, в интегрированной среде программирования Free Pascal (изучение структуры «цикл»).

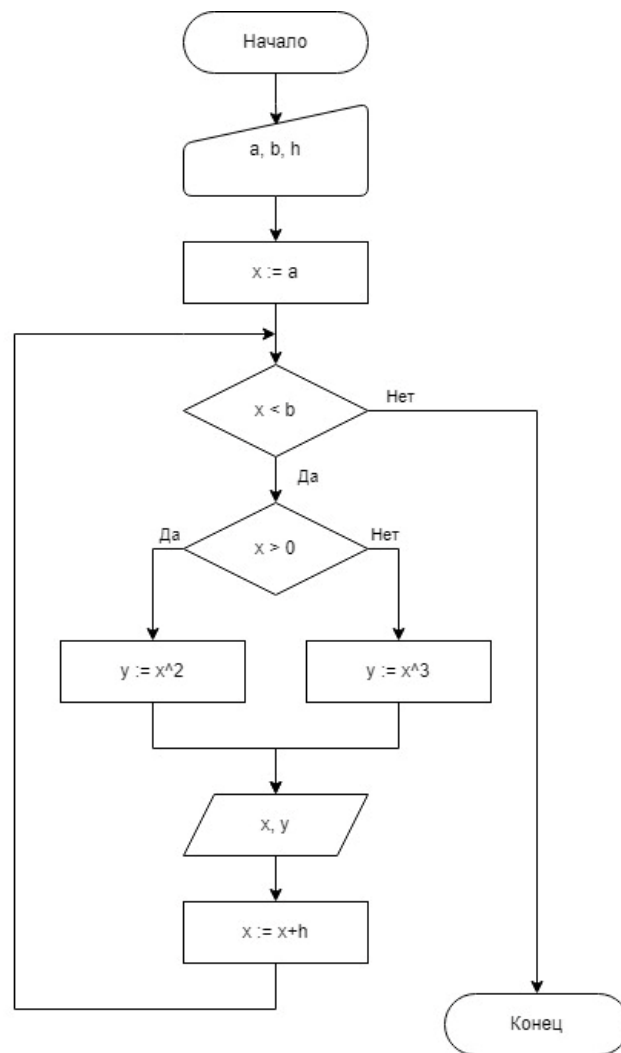
Цель работы: Получение навыков вычисления значений функций при различных переменных на языке Паскаль, введенных с клавиатуры (изучение циклов).

Выполнение лабораторной работы.

1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 5 (файл 5/example5.pas):

```
program example5;
var x, y: real;
    a, b, h: real;
begin
write('Введите a, b, h: ');
read(a, b, h);
writeln(' x,      y');
x := a;
while (x <= b) do begin
    if (x > 0) then y := sqr(x)
    else y := sqr(x) * x;
    if (x >= 0) then write(' ');
    write(x:1:3);
    if (y >= 0) then write('  ')
    else write(' ');
    writeln(y:1:3);
    x := x + h;
end;
end.
```

Блок-схема алгоритма программы:



3. Результат выполнения программы (запуск произведен через PowerShell):

```
PS C:\Users\User\Documents\Programming\Labs\5> ./example5
Введите a, b, h: -5 7 0.8
x,      y
-5.000  -125.000
-4.200  -74.088
-3.400  -39.304
-2.600  -17.576
-1.800  -5.832
-1.000  -1.000
-0.200  -0.008
0.600    0.360
1.400    1.960
2.200    4.840
3.000    9.000
3.800   14.440
4.600   21.160
5.400   29.160
6.200   38.440
7.000   49.000
PS C:\Users\User\Documents\Programming\Labs\5>
```

The screenshot shows a PowerShell window where the program was executed. The user entered the values $a = -5$, $b = 7$, and $h = 0.8$. The program then outputs a table of values for x and y at intervals of h , ranging from $x = -5.000$ to $x = 7.000$.

Ответы на контрольные вопросы.

1. Как организовать ввод переменных с клавиатуры?

Ввод переменных с клавиатуры организуется при помощи операторов `read()` и `readln()`. `read()` записывает значения, введенные через пробел одной строкой, в переменные, указанные как параметры оператора. `readln()` записывает значения в переменные с переводом строки. Желательно перед вводом значений выполнить приглашение для пользователя программы с пояснением, что требуется ввести.

2. Чем отличаются программы, работающие с переменными, значения которых прописаны в программе от программ, где значения переменных вводятся с клавиатуры?

Программы, работающие с переменными, значения которых прописаны в исходном коде отличаются от программ, где значения переменных вводятся с клавиатуры, отсутствием взаимодействия с пользователем, поэтому результат из выполнения всегда один и тот же. В программах с вводом значений программисту всегда необходимо предусматривать все критические значения, которые может ввести пользователь, так как есть большая вероятность аварийного завершения программы. Например, пользователь может ввести текст вместо числа, или 0 в переменную, которая является знаменателем дроби.

Лабораторная работа № 6

Название: Суммирование значений в интегрированной среде программирования Free Pascal (изучение структуры «цикл»).

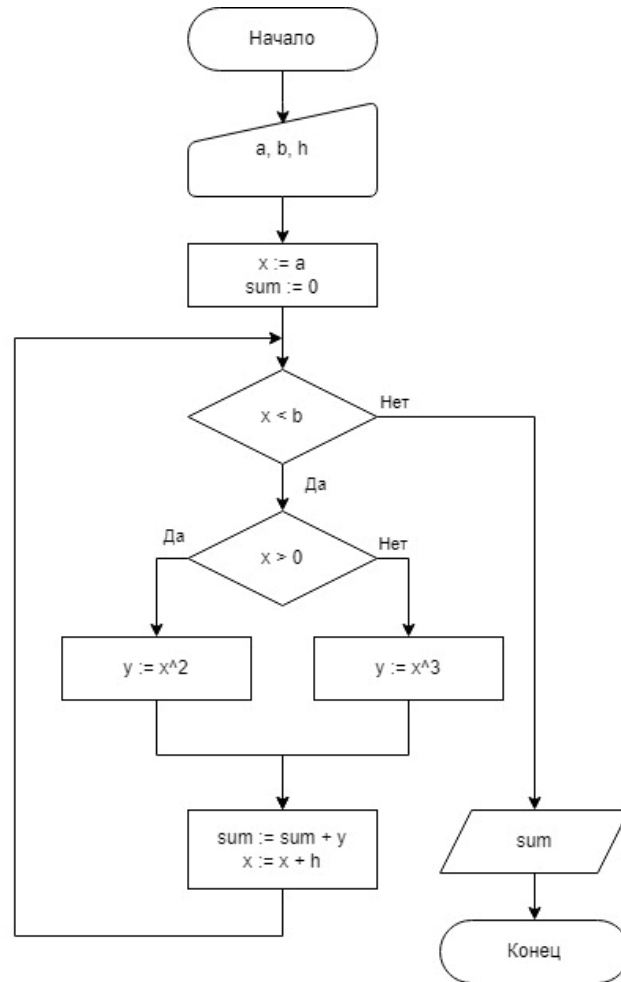
Цель работы: Получение навыков по суммированию значений на языке Паскаль (составление сложных программ).

Выполнение лабораторной работы.

1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 6 (файл 6/example6.pas):

```
program example6;
var x, y: real;
    a, b, h: real;
begin
write('Введите a, b, h: ');
read(a, b, h);
writeln(' x,      y');
x := a;
sum := 0;
while (x <= b) do begin
    if (x > 0) then y := sqr(x)
    else y := sqr(x) * x;
    sum := sum + y;
    x := x + h;
end;
write('Сумма равна ');
writeln(sum:1:2);
end.
```

Блок-схема алгоритма программы:



3. Результат выполнения программы (запуск произведен через PowerShell):

```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\6> ./example6
Введите a, b, h: -5 7 0.8
Сумма равна -94.45
PS C:\Users\User\Documents\Programming\Labs\6> .
```

The screenshot shows a PowerShell 7 (x64) window. The user runs the command './example6'. The program prompts 'Введите a, b, h:' and the user enters '-5 7 0.8'. The program outputs 'Сумма равна -94.45'. The prompt 'PS C:\Users\User\Documents\Programming\Labs\6>' is shown again with a trailing dot.

Ответы на контрольные вопросы.

1. Как в программе получить сумму некоторых значений?

Необходимо объявить переменную, например, `sum`, в которую будет сохраняться значение суммы, и инициализировать ее нулем (`sum := 0`). Далее, при вычислении каждого следующего слагаемого в теле цикла, на каждой итерации увеличиваем значение суммы на значение слагаемого путем их сложения, например, `sum := sum + value`. По окончании цикла переменная `sum` будет иметь значение равное сумме всех значений.

2. Как в программе получить произведение некоторых значений?

Необходимо объявить переменную, например, `mul`, в которую будет сохраняться значение произведения, и инициализировать ее единицей (`mul := 1`). Далее, при вычислении каждого следующего значения в теле цикла, на каждой итерации умножаем значение произведения на новое полученное значение и это значение присваиваем переменной произведения, например, `mul := mul * value`. По окончании цикла переменная `mul` будет иметь значение равное произведению всех значений.

Лабораторная работа № 7

Название: Вычисление значений факториалов и степеней в интегрированной среде программирования Free Pascal.

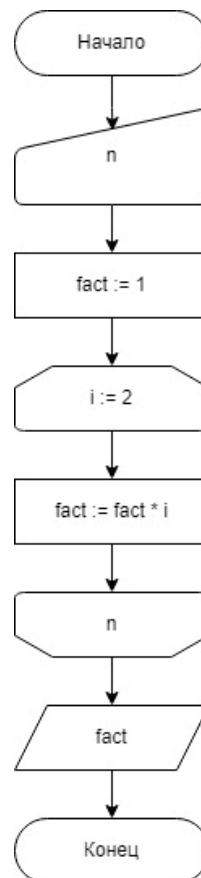
Цель работы: Получение навыков по вычислению значений факториалов и степеней в программе, составленной на языке Паскаль.

Выполнение лабораторной работы.

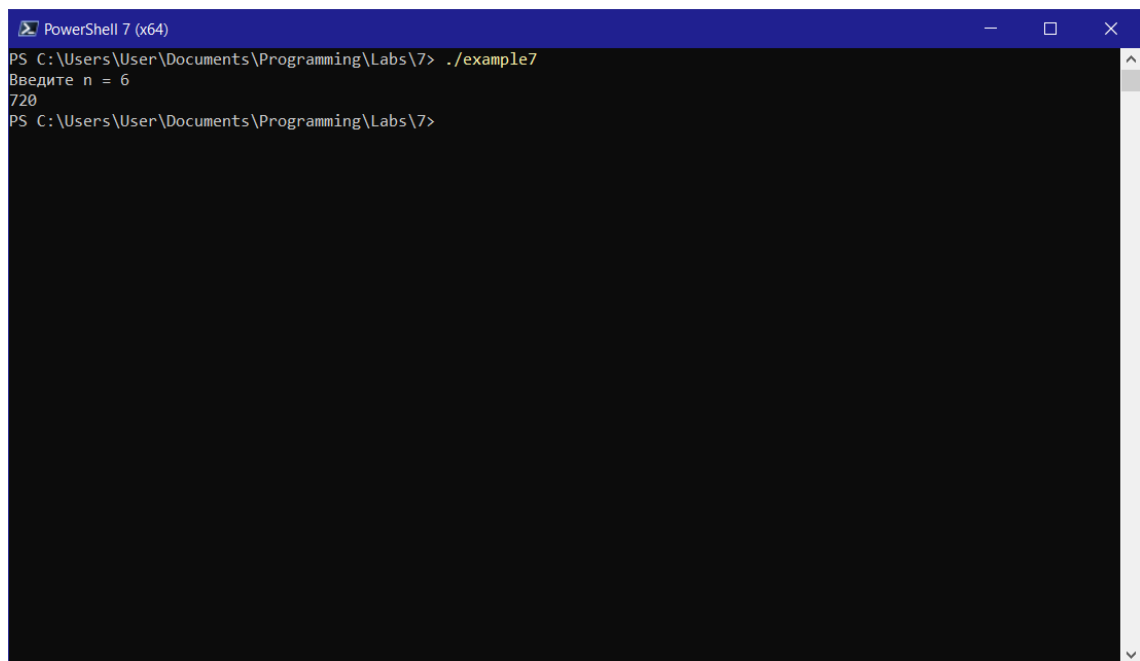
1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 7 (файл 7/example7.pas):

```
program example7;  
var fact, n, i: integer;  
begin  
  write('Введите n = ');  
  readln(n);  
  fact := 1;  
  for i := 2 to n do fact := fact * i;  
  writeln(fact);  
end.
```

Блок-схема алгоритма программы:



3. Результат выполнения программы (запуск произведен через PowerShell):

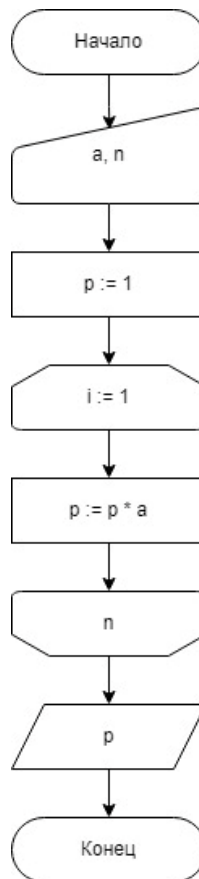


```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\7> ./example7
Введите n = 6
720
PS C:\Users\User\Documents\Programming\Labs\7>
```

4. Исходный код программы в соответствии с примером 8 (файл 7/example8.pas):

```
program example8;
var a, p: real;
    i, n: word;
begin
write('Введите основание степени a = ');
readln(a);
write(' Введите показатель степени n = ');
readln(n);
p := 1;
for i := 1 to n do p := p * a;
writeln('p = ', p:2:3);
end.
```

Блок-схема алгоритма программы:



5. Результат выполнения программы (запуск произведен через PowerShell):

```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\7> ./example8
Введите основание степени a = 4.5
Введите показатель степени n = 4
p = 410.063
PS C:\Users\User\Documents\Programming\Labs\7> █
```

The screenshot shows a PowerShell 7 (x64) terminal window. The user runs the command `./example8`. The program prompts for the base 'a' (4.5) and the exponent 'n' (4). It then outputs the result 'p = 410.063'.

Ответы на контрольные вопросы.

1. Что такое «строгий цикл»?

Строгим называется цикл который должен быть выполнен заданное число раз.

2. Как в программе организовать «строгий цикл»?

Строгий цикл задается командой типа

for индекс := конечное_знач downto начальное_знач do <оператор>
или

for индекс := начальное_знач to конечное_знач do <оператор>.

Оператор, представляющий собой тело цикла может быть простым или составным.

3. Что такое параметр цикла и тело цикла?

Параметр цикла – это переменная любого из перечисляемых типов (целого, булевого, символьного, диапазонного, перечисления). Начальные и конечные значения могут быть представлены не только значениями, но и выражениями, возвращающими совместимые с типом параметра типы данных. Значение параметра можно использовать в теле цикла, но крайне не рекомендуется изменять, так как это может привести к непредвиденным ситуациям.

Тело цикла – это простой или составной оператор, выполняющийся на каждой итерации цикла.

4. Какие значение может принимать параметр цикла?

Параметр цикла может принимать любое целое значение в качестве начального значения и изменяется либо в большую, либо в меньшую сторону на единицу на каждой итерации цикла.

5. Как устанавливается шаг цикла при использовании структуры for ... do ...?

Шаг цикла всегда постоянный и равен интервалу между двумя ближайшими значениями типа, которым задается параметр цикла. Если тип integer, то шаг равен 1, так как указанный тип описывает натуральные числа. При необходимости задать шаг больше 1, например, 3, необходимо увеличить диапазон так же в 3 раза, и в теле цикла выполнять действия только при значении параметра, кратном 3.

Лабораторная работа № 8

Название: Обработка массивов (часть 1) на языке высокого уровня Паскаль в среде Free Pascal.

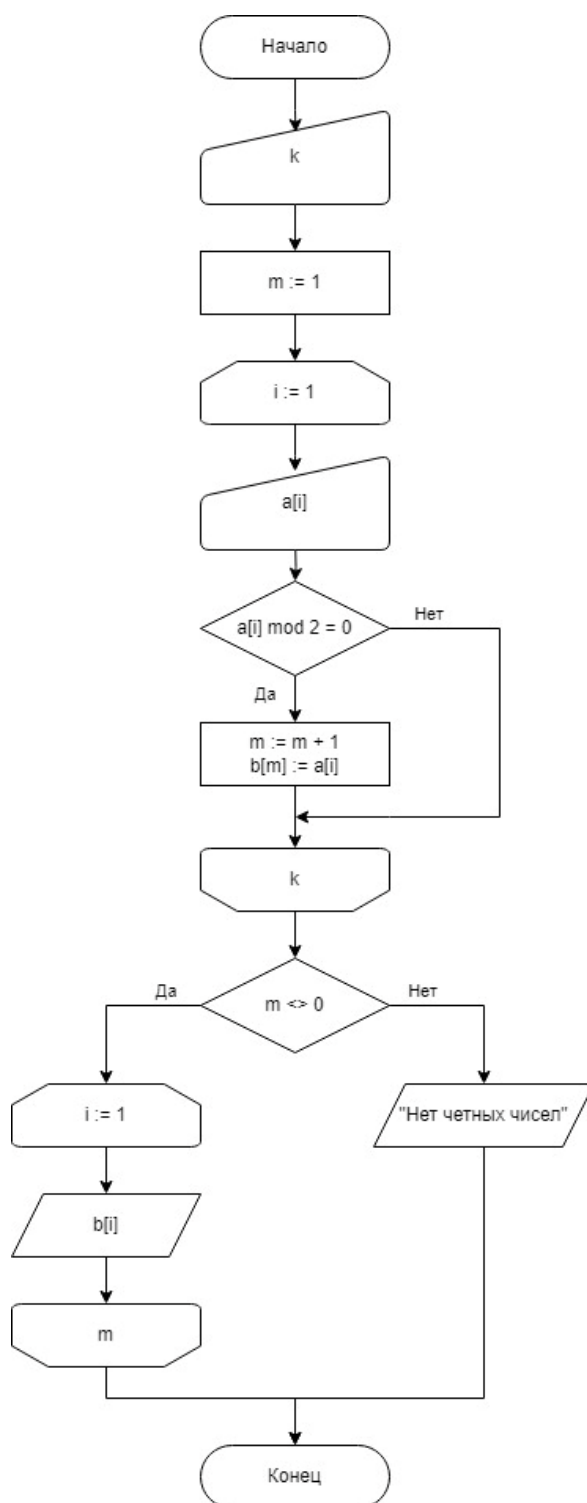
Цель работы: Освоить основные приемы обработки массивов на языке высокого уровня Паскаль.

Выполнение лабораторной работы.

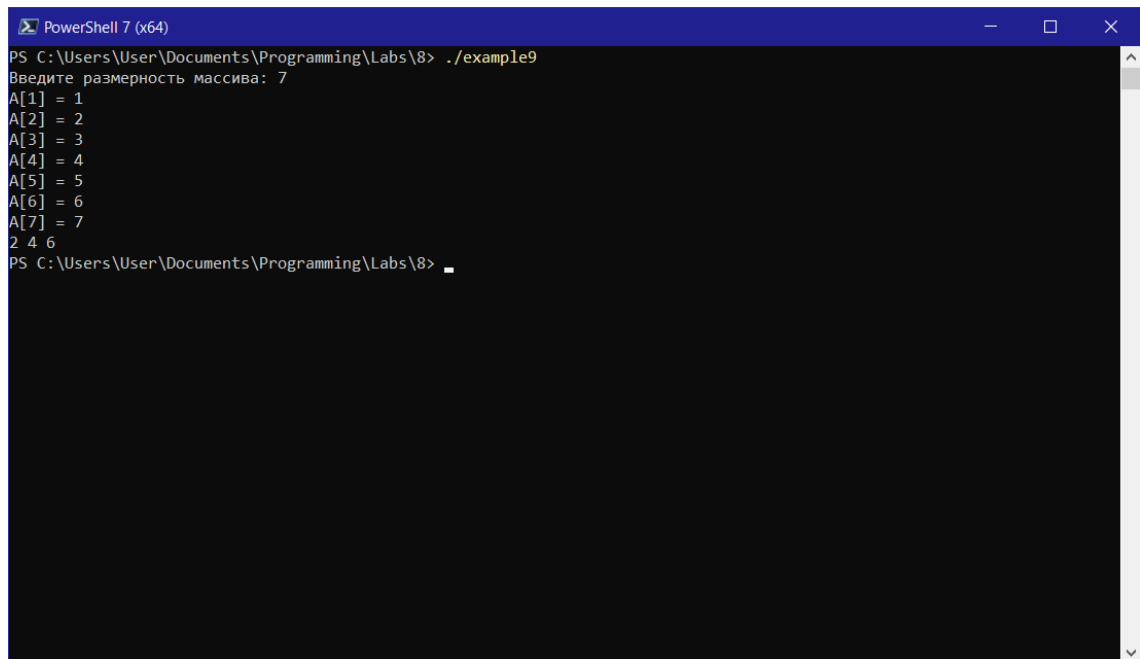
1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 9 (файл 8/example9.pas):

```
program example9;
var a, b: array[1..20] of word;
    k, m, i: byte;
begin
write('Введите размерность массива: ');
readln(k);
m := 0;
for i := 1 to k do begin
    write('A[' , i, '] = ');
    readln(a[i]);
    if (a[i] mod 2 = 0) then begin
        m := m + 1;
        b[m] := a[i];
    end;
end;
if (m <> 0) then
    for i := 1 to m do
        write(b[i], ' ')
else write('В массиве нет четных чисел');
end.
```

Блок-схема алгоритма программы:



3. Результат выполнения программы (запуск произведен через PowerShell):



```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\8> ./example9
Введите размерность массива: 7
A[1] = 1
A[2] = 2
A[3] = 3
A[4] = 4
A[5] = 5
A[6] = 6
A[7] = 7
2 4 6
PS C:\Users\User\Documents\Programming\Labs\8> _
```

Ответы на контрольные вопросы.

1. Что такое массив?

Массив – это структурированный тип данных, состоящий из фиксированного числа элементов одного типа.

2. Какие массивы существуют?

Массивы бывают одномерные, двумерные и многомерные.

3. Как задается одномерный массив?

var

a: array [1..n] of real;

где a – имя массива, [1..n] – размерность от 1 до n, real – тип данных массива.

4. Как задается двумерный массив?

var

a: array [1..n, 1..m] of real;

где a – имя массива, [1..n, 1..m] – n строк, m столбцов, real – тип данных массива.

5. Какие операции можно осуществлять над массивами?

ввод/вывод элементов массива;

вычисление суммы или произведения элементов массива;

поиск минимального или максимального элемента в массиве;

сортировка элементов в массиве (различными методами).

Добавлять или удалять элементы массива невозможно, так как это структура фиксированной размерности.

6. Что такое однородность данных?

Однородностью данных называется состояние структуры, использующейся для хранения данных, при одинаковом типе хранимых данных.

7. Какими способами задается одномерный массив на языке Паскаль?

В разделе var

```
a: array [1..5] of real;
```

Через задание типа:

Type

```
massiv= array [0..12] of real;
```

var

```
a: massiv;
```

С использованием констант:

```
const
```

```
n = 10;
```

var

```
a: array [1..n] of real;
```

8. Какими способами задается двумерный массив на языке Паскаль?

В разделе var

```
a: array [1..5, 1..10] of real;
```

Через задание типа:

Type

```
massiv = array [0..10] of real;
```

```
matrix = array [0..10] of massiv;
```

или

```
matrix = array [0..10, 1..10] of real;
```

var

```
a: matrix;
```

С использованием констант:

```
const
```

```
n = 10;
```

```
m = 15;
```

var

```
a: array [1..n, 1..m] of real;
```


Лабораторная работа № 9

Название: Обработка массивов (часть 2) на языке высокого уровня Паскаль в среде Free Pascal.

Цель работы: Освоить основные приемы обработки массивов на языке высокого уровня Паскаль.

Выполнение лабораторной работы.

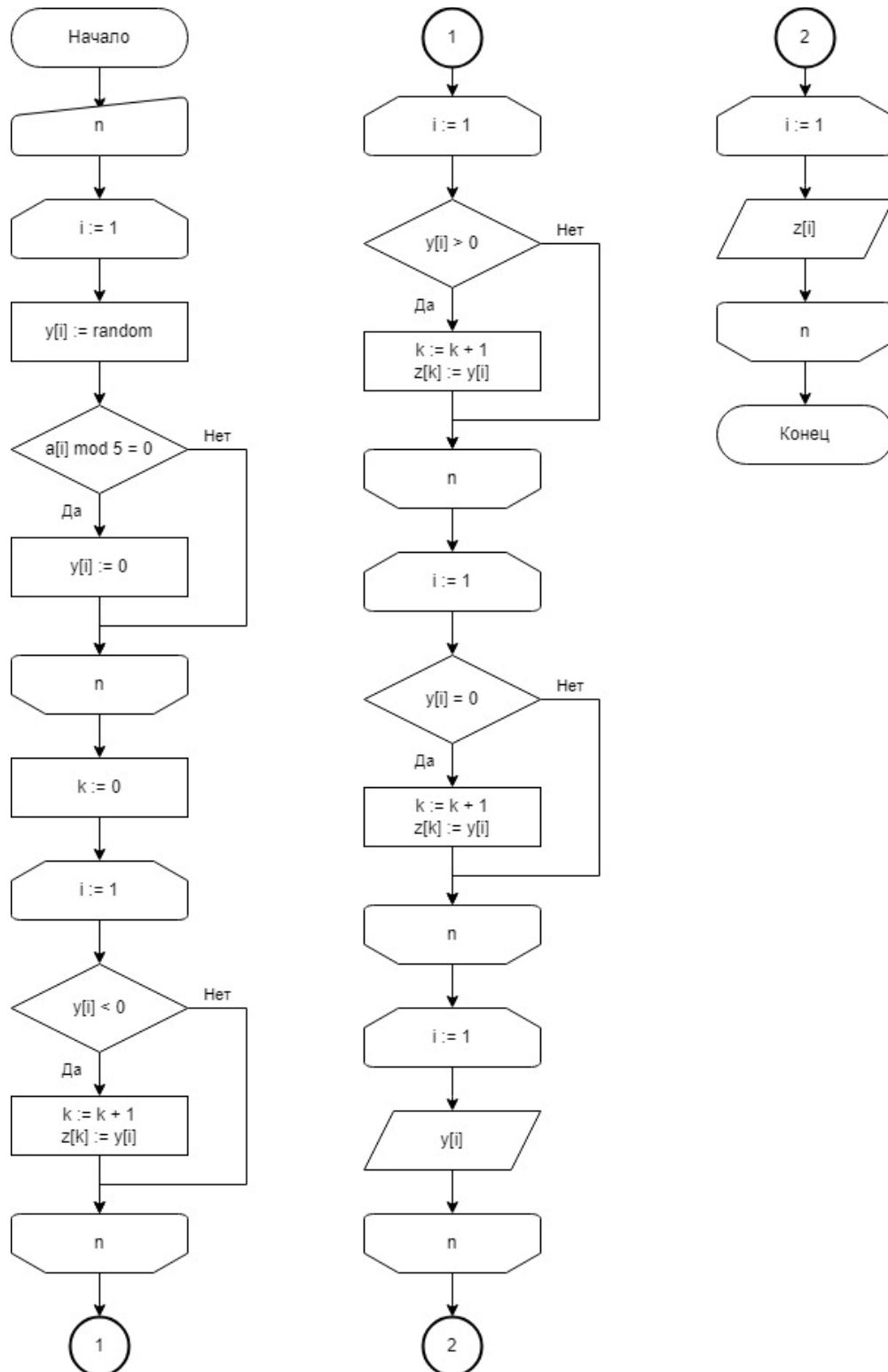
1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 10 (файл 9/example10.pas):

```
program example10;
var y, z: array[1..50] of integer;
    i, k, n: integer;
begin
  randomize;
  write(' Введите размерность массива (меньше 50): ');
  readln(n);
  for i := 1 to n do begin
    y[i] := random(200) - 100;
    if ((y[i] mod 5) = 0) then y[i] := 0;
  end;
  k := 0;
  for i := 1 to n do
    if (y[i] < 0) then begin
      k := k + 1;
      z[k] := y[i];
    end;
  for i := 1 to n do
    if (y[i] > 0) then begin
      k := k + 1;
      z[k] := y[i];
    end;
  for i := 1 to n do
    if (y[i] = 0) then begin
      k := k + 1;
      z[k] := y[i];
    end;
  writeln('Массив y:');
  for i := 1 to n do write(y[i], ' ');
  writeln;
  writeln('Массив z:');
  for i := 1 to n do write(z[i], ' ');
  writeln;
end.
```

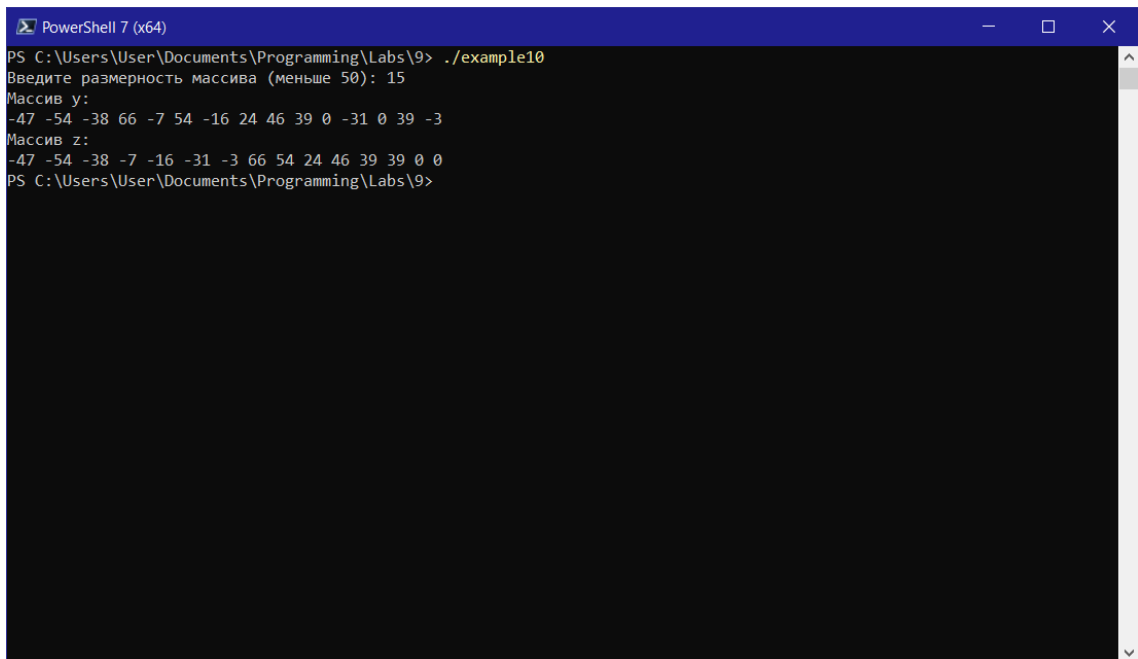
Пояснение к коду программы:

Для ускорения тестирования и отладки программы вместо ручного наполнения с клавиатуры тестовых данных применено использование генератора случайных чисел в диапазоне от -100 до 100, причем числа кратные 5 заменены на 0.

Блок-схема алгоритма программы:



3. Результат выполнения программы (запуск произведен через PowerShell):



```
PS C:\Users\User\Documents\Programming\Labs\9> ./example10
Введите размерность массива (меньше 50): 15
Массив у:
-47 -54 -38 66 -7 54 -16 24 46 39 0 -31 0 39 -3
Массив z:
-47 -54 -38 -7 -16 -31 -3 66 54 24 46 39 39 0 0
PS C:\Users\User\Documents\Programming\Labs\9>
```

Ответы на контрольные вопросы.

1. Как реализовать запись из одного массива в другой отрицательных элементов массива?

Необходимо в цикле пройти по всем элементам одного массива проверяя их меньше ли они 0. Если да, то копировать данный элемент массива в другой массив.

2. Как реализовать запись из одного массива в другой положительных элементов массива?

Необходимо в цикле пройти по всем элементам одного массива проверяя их больше ли они 0. Если да, то копировать данный элемент массива в другой массив.

3. Как реализовать запись из одного массива в другой нулевых элементов массива?

Необходимо в цикле пройти по всем элементам одного массива проверяя их на равенство 0. Если да, то копировать данный элемент массива в другой массив.

4. Как присвоить элементу, записываемому в новый массив, нового индекса?

Зная размерность нового массива через индекс можно получить доступ к любому элементу этого нового массива. Если элементы необходимо записывать последовательно, то в данной ситуации можно использовать отдельный счетчик, который увеличивается при каждом получении нового значения, и использовать его как индекс или количество элементов нового массива.

Лабораторная работа № 10

Название: Обработка массивов (часть 3) на языке высокого уровня Паскаль в среде Free Pascal.

Цель работы: Освоить основные приемы обработки массивов на языке высокого уровня Паскаль.

Выполнение лабораторной работы.

1. Изучены теоретические положения работы.
2. Исходный код программы в соответствии с примером 11 (файл 10/example11.pas):

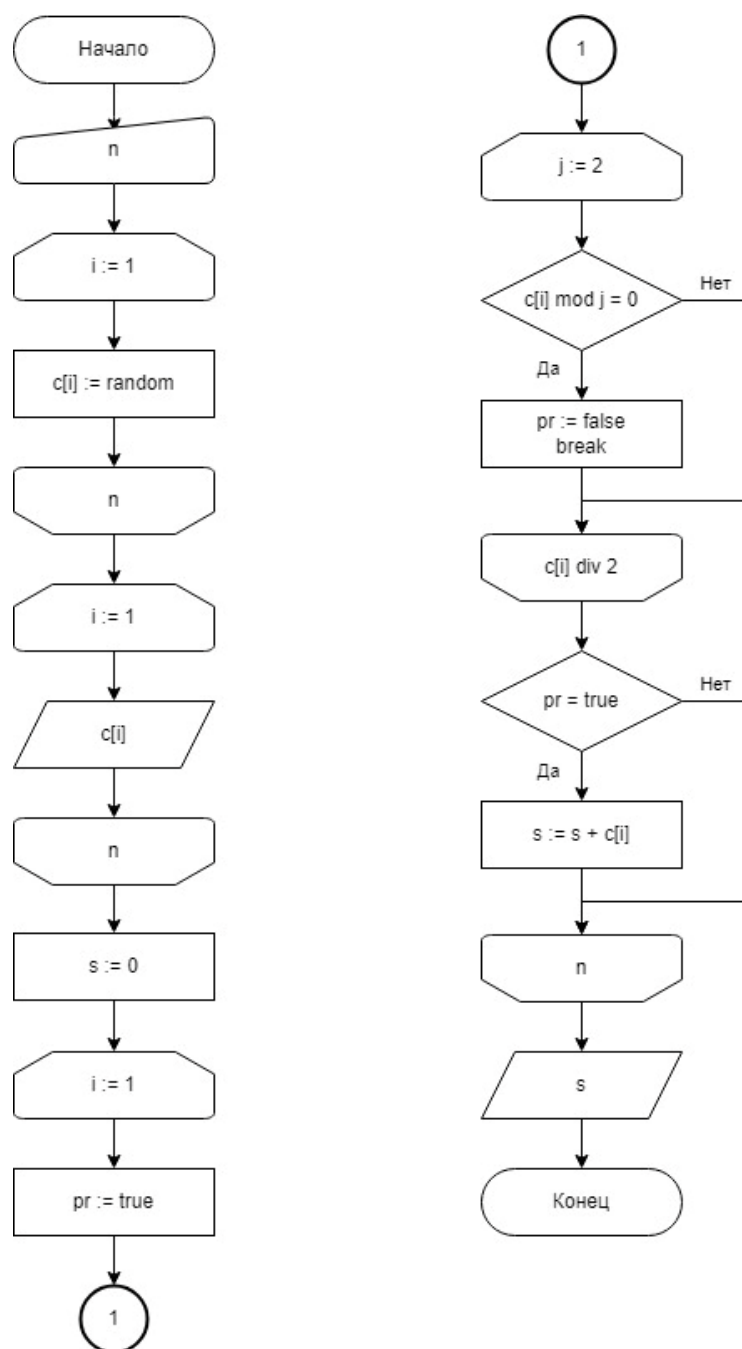
```
program example11;
var c: array[1..50] of integer;
    i, j, n: byte;
    s: word;
    pr: boolean;
begin
  randomize;
  write(' Введите размерность массива (меньше 50):');
  readln(n);
  for i := 1 to n do begin
    c[i] := random(100);
  end;
  for i := 1 to n do write(c[i], ' ');
  writeln;
  s := 0;
  for i := 1 to n do begin
    pr := true;
    for j := 2 to c[i] div 2 do
      if (c[i] mod j = 0) then begin
        pr := false;
        break;
      end;
    if (pr = true) then s := s + c[i];
  end;
  writeln(' Сумма простых чисел массива: ', s);
end.
```

Пояснение к коду программы:

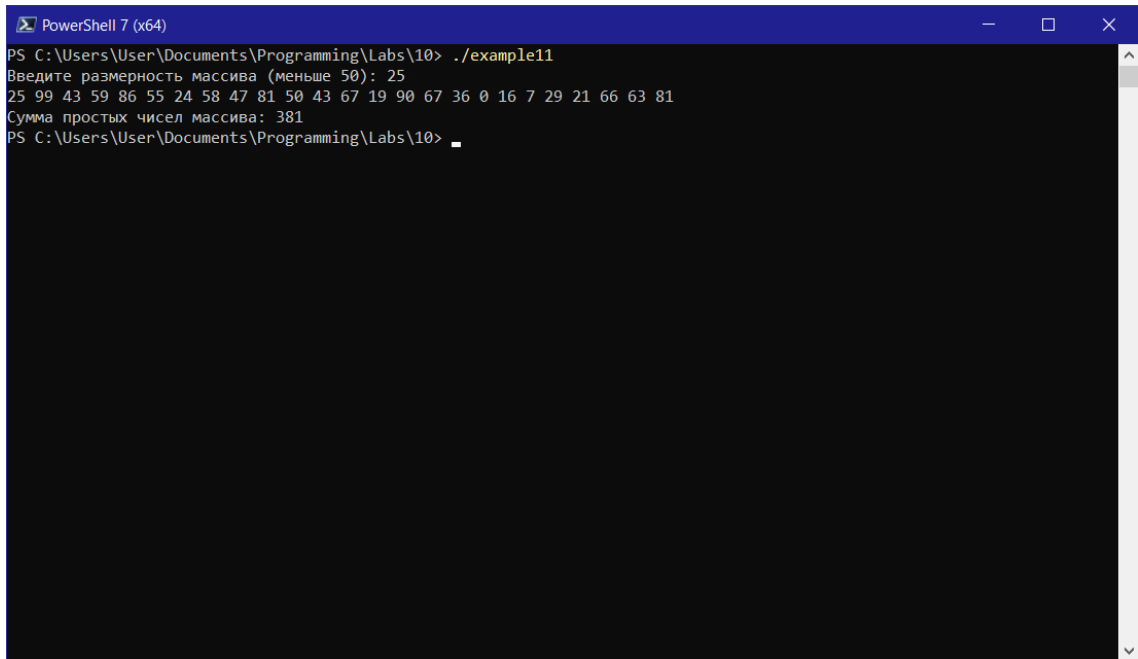
Для ускорения тестирования и отладки программы вместо ручного наполнения с клавиатуры тестовых данных применено использование генератора случайных чисел в диапазоне от 0 до 100.

Для определения является ли очередное число простым, данное число делится на числа от 2 до половины этого числа (например, если исследуем 15, то делим его на пополам и берем целую часть, то есть 7). Если в каком-либо случае число делится без остатка, значит, число не является простым. Если же нет чисел, делящих исследуемое число нацело, то число является простым.

Блок-схема алгоритма программы:



3. Результат выполнения программы (через PowerShell):



```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\10> ./example11
Введите размерность массива (меньше 50): 25
25 99 43 59 86 55 24 58 47 81 50 43 67 19 90 67 36 0 16 7 29 21 66 63 81
Сумма простых чисел массива: 381
PS C:\Users\User\Documents\Programming\Labs\10> _
```

Ответы на контрольные вопросы.

1. Как найти сумму элементов (любого свойства) массива?

Необходимо в цикле пройтись по всем элементам данного массива, чтобы получить к ним доступ по индексу. На каждой итерации выполняем соответствующие действия, например, сравнение для поиска, сложение для получения суммы всех элементов, умножение для получения произведения всех элементов и так далее.

2. Зачем вводится логическая переменная типа boolean?

Переменная типа boolean является своего рода флагом, определяющим какое-либо свойство для некоторого диапазона значений или элемента массива, например, в рассмотренном выше примере булева переменная является флагом для выявления является ли число простым, так как для каждого проверяемого числа на простоту проверяется диапазон чисел от 2 до целой половины это числа.

3. Как работает операция дивергенция (div)?

Операция div является операцией целочисленного деления, то есть ее результатом является целая часть от деления. Например, при выполнении $14 \div 3$ результат будет равен 4.

4. Как работает операция модуляция (mod)?

Операция mod является операцией получения остатка от целочисленного деления. Например, при выполнении $14 \bmod 3$ результат будет равен 2.

5. Как осуществить досрочный выход из цикла?

Досрочный выход из цикла выполняется оператором break. Например, в данном примере происходит досрочный выход из цикла для проверки является ли число простым при обнаружении первого же делителя, так дальнейшая проверка не имеет смысла.

Лабораторная работа № 11

Название: Разработка программы на языке высокого уровня Паскаль (в среде Free Pascal).

Цель работы: Освоить основные приемы алгоритмизации и самостоятельного составления программ на языке высокого уровня Паскаль.

Выполнение лабораторной работы.

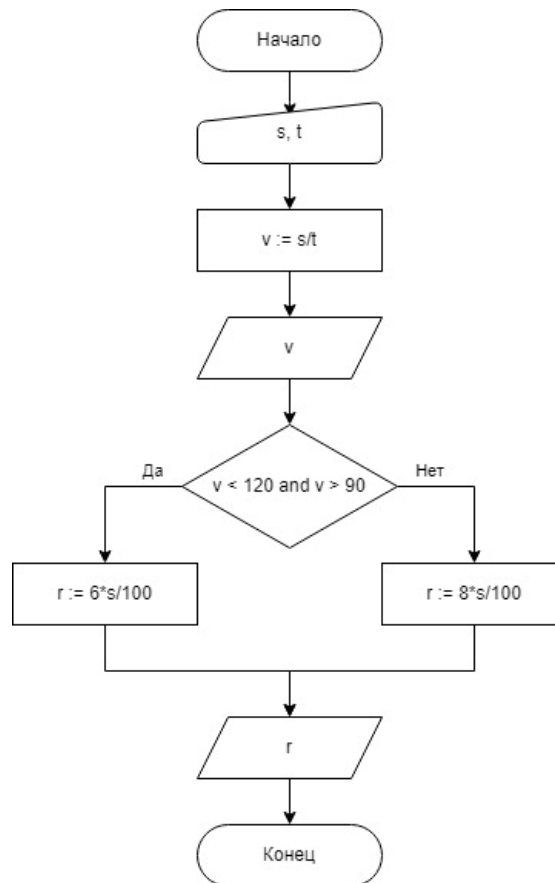
1. Изучены теоретические положения работы.
2. Выполнено решение задачи с вариантом № 3. Исходный код программы (файл 11/example12.pas):

```
program example12;
var s, t, v, r: real;
begin
write(' Введите расстояние (км): ');
readln(s);
write(' Введите время движения (час): ');
readln(t);
v := s/t;
writeln(' Скорость движения: ', v:1:2, ' км/ч');
if (v < 120) and (v > 90) then r := 6*s/100
else r := 8*s/100;
writeln(' Расход топлива: ', r:1:2, ' л');
end.
```

Пояснение к коду программы:

Поучаем от пользователя значения расстояния и времени движения, на основании них вычисляем скорость. В зависимости от значения скорости считаем расход топлива либо с коэффициентом 6 (при скорости от 90 км/ч до 120 км/ч), либо с коэффициентом 8 при других значениях.

Блок-схема алгоритма программы:



3. Результат выполнения программы (через PowerShell):

```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\11> ./example12
Введите расстояние (км): 100
Введите время движения (час): 1
Скорость движения: 100.00 км/ч
Расход топлива: 6.00 л
PS C:\Users\User\Documents\Programming\Labs\11> ./example12
Введите расстояние (км): 100
Введите время движения (час): 1.5
Скорость движения: 66.67 км/ч
Расход топлива: 8.00 л
PS C:\Users\User\Documents\Programming\Labs\11> ./example12
Введите расстояние (км): 100
Введите время движения (час): 0.5
Скорость движения: 200.00 км/ч
Расход топлива: 8.00 л
PS C:\Users\User\Documents\Programming\Labs\11> _
```

The screenshot shows a PowerShell 7 (x64) terminal window. It displays three runs of the program './example12'. Each run prompts for distance (s) and time (t), calculates speed (v), and then outputs the fuel consumption (r). The first run (s=100, t=1) results in v=100.00 km/h and r=6.00 l. The second run (s=100, t=1.5) results in v=66.67 km/h and r=8.00 l. The third run (s=100, t=0.5) results in v=200.00 km/h and r=8.00 l.

Программа была запущена 3 раза с расстоянием 100 км и различным временем движения: час, полтора часа и полчаса. В первом случае скорость 100 км/ч, то есть в диапазоне от 90 до 120 км/ч – расход 6 литров. В двух остальных скорость либо меньше, либо больше – расход 8 литров.

Ответы на контрольные вопросы.

1. Какие структуры вы использовали для составления схемы работы программы в соответствии с полученным вариантом?

Использована структура условного ветвления на основании значения полученной скорости движения.

2. Какие операторы Вы использовали для составления программы на языке паскаль в среде FreePascal?

Использованы операторы ввода `readln`, операторы вывода `writeln`, оператор условного ветвления `if ... then ... else ...`, оператор присваивания `:=`.

Лабораторная работа № 12

Название: Разработка программы на языке высокого уровня Паскаль (в среде Free Pascal) с использованием строгого цикла.

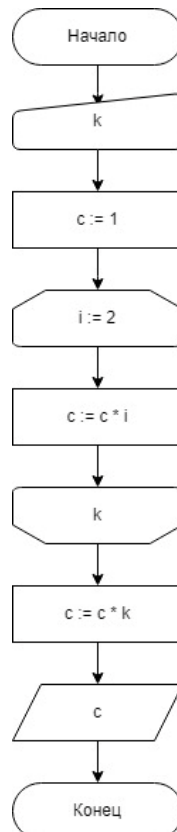
Цель работы: Освоить основные приемы алгоритмизации и самостоятельного составления программ на языке высокого уровня Паскаль с использованием строгого цикла.

Выполнение лабораторной работы.

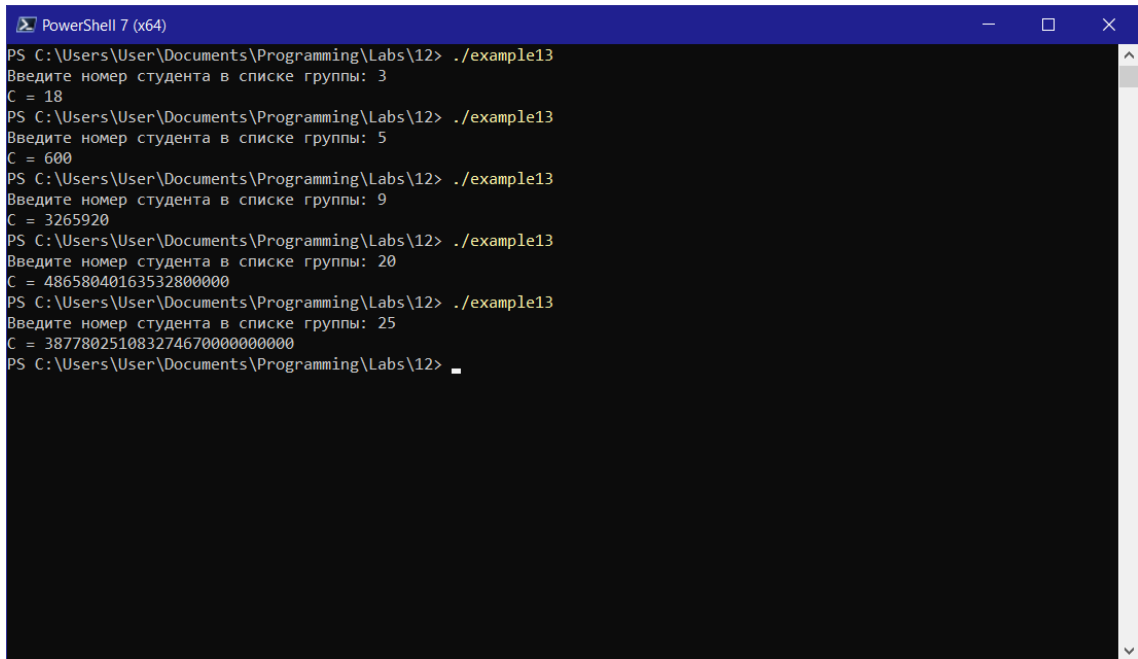
1. Изучены теоретические положения работы.
2. Выполнено решение задачи с вариантом № 3. Исходный код программы (файл 12/example13.pas):

```
program example13;  
var k, i: integer;  
    c: real;  
begin  
  write(' Введите номер студента в списке группы: ');  
  readln(k);  
  c := 1;  
  for i := 2 to k do c := c * i;  
  c := c * k;  
  writeln('C = ', c:1:0);  
end.
```

Блок-схема алгоритма программы:



3. Результат выполнения программы (через PowerShell):



```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\12> ./example13
Введите номер студента в списке группы: 3
C = 18
PS C:\Users\User\Documents\Programming\Labs\12> ./example13
Введите номер студента в списке группы: 5
C = 600
PS C:\Users\User\Documents\Programming\Labs\12> ./example13
Введите номер студента в списке группы: 9
C = 3265920
PS C:\Users\User\Documents\Programming\Labs\12> ./example13
Введите номер студента в списке группы: 20
C = 48658040163532800000
PS C:\Users\User\Documents\Programming\Labs\12> ./example13
Введите номер студента в списке группы: 25
C = 3877802510832746700000000000
PS C:\Users\User\Documents\Programming\Labs\12> .
```

Ввиду быстрого выхода из диапазона значений типа `integer` для типа конечного результата (переменной `c`) был выбран тип `real` с форматированием вывода без дробной части.

Ответы на контрольные вопросы.

1. Какие типы данных Вы использовали для составления программы на языке Паскаль в среде FreePascal?

Для значения номера студента в группе и счетчика цикла был использован тип `integer`, для значения факториала – тип `real`.

2. Какие структуры вы использовали для составления схемы работы программы в соответствии с полученным вариантом?

Был использован строгий цикл типа `for ... to ... do`

3. Какие операторы Вы использовали для составления программы на языке Паскаль в среде FreePascal?

Использованы операторы ввода `readln`, операторы вывода `writeln`, операторы строго цикла `for ... to ... do`, оператор присваивания `:=`.

4. Что составило тело цикла программы?

Тело цикла составили операции вывода приглашения, ввода номера студента в группе, присваивание начального значения результату, строгий цикл с подсчетом факториала, расчет конечного результата, вывод результата.

5. Какие ошибки были выявлены при компиляции?

Так как номер студента задается интерактивно, ошибки компиляции отсутствовали. Но так как изначально для результата был выбран тип `integer`, происходила ошибка времени выполнения при вводе номера студента больше

8. Ошибка устранена при выборе типа результата как `real`.

Лабораторная работа № 13

Название: Разработка программы на языке высокого уровня Паскаль (в среде Free Pascal) с использованием средств обработки массивов.

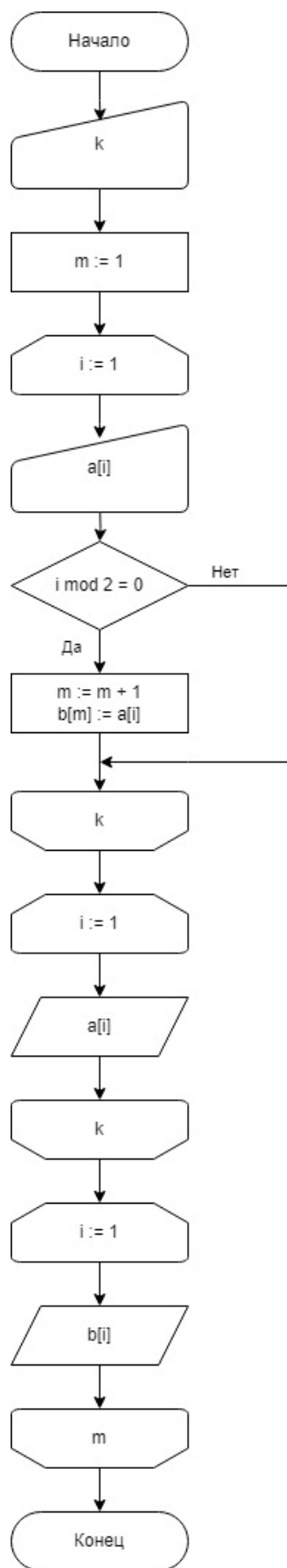
Цель работы: Освоить основные приемы алгоритмизации и самостоятельного составления программ на языке высокого уровня Паскаль с использованием средств обработки массивов.

Выполнение лабораторной работы.

1. Изучены теоретические положения работы.
2. Выполнено решение 1-ой части задачи с вариантом № 3. Исходный код программы (файл 13/example14_1.pas):

```
program example14_1;
var a, b: array[1..20] of word;
    k, m, i: byte;
begin
write(' Введите размерность массива: ');
readln(k);
m := 0;
for i := 1 to k do begin
    write('A[', i, '] = ');
    readln(a[i]);
    if (i mod 2 = 0) then begin
        m := m + 1;
        b[m] := a[i];
    end;
end;
writeln(' Массив A:');
for i := 1 to k do write(a[i], ' ');
writeln;
writeln(' Массив B:');
for i := 1 to m do write(b[i], ' ');
end.
```

Блок-схема алгоритма программы:



Результат выполнения программы (через PowerShell):

```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\13> ./example14_1
Введите размерность массива: 10
A[1] = 1
A[2] = 2
A[3] = 3
A[4] = 2
A[5] = 4
A[6] = 2
A[7] = 5
A[8] = 2
A[9] = 6
A[10] = 2
Массив A:
1 2 3 2 4 2 5 2 6 2
Массив B:
2 2 2 2
PS C:\Users\User\Documents\Programming\Labs\13>
```

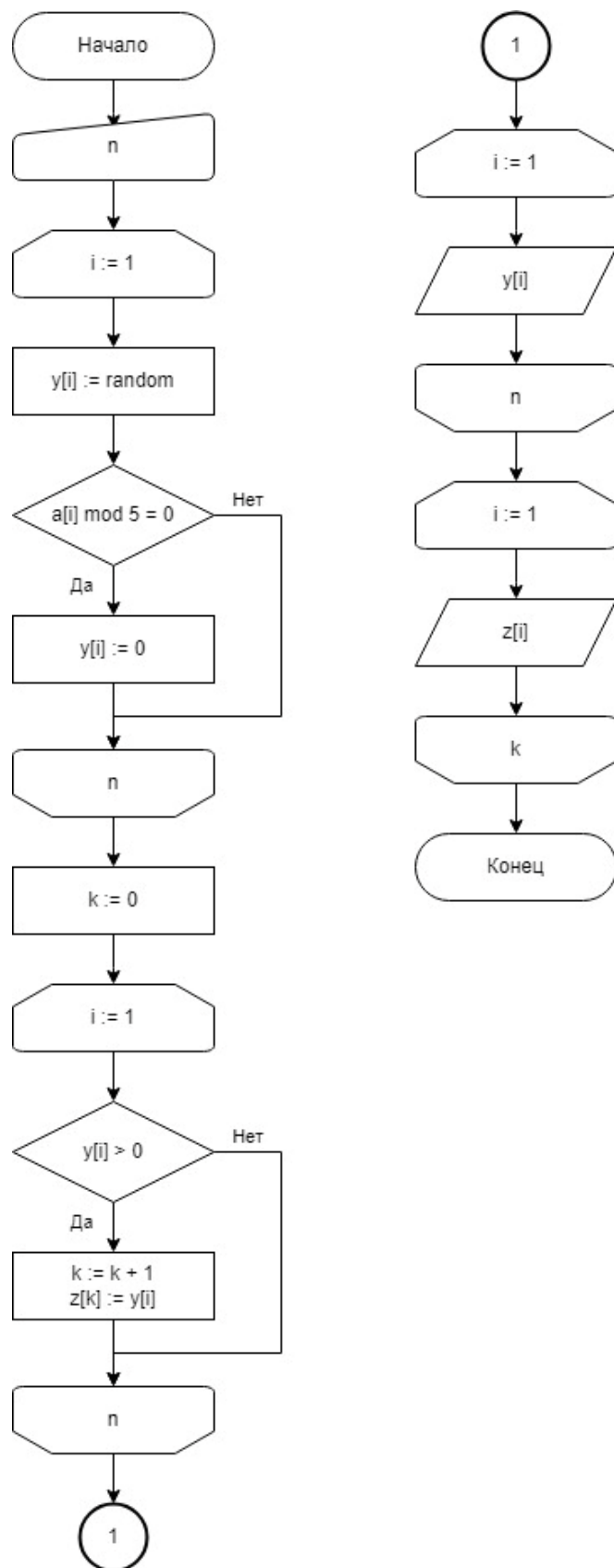
3. Выполнено решение 2-ой части задачи с вариантом № 3. Исходный код программы (файл 13/example14_2.pas):

```
program example14_2;
var y, z: array[1..50] of integer;
    i, k, n: integer;

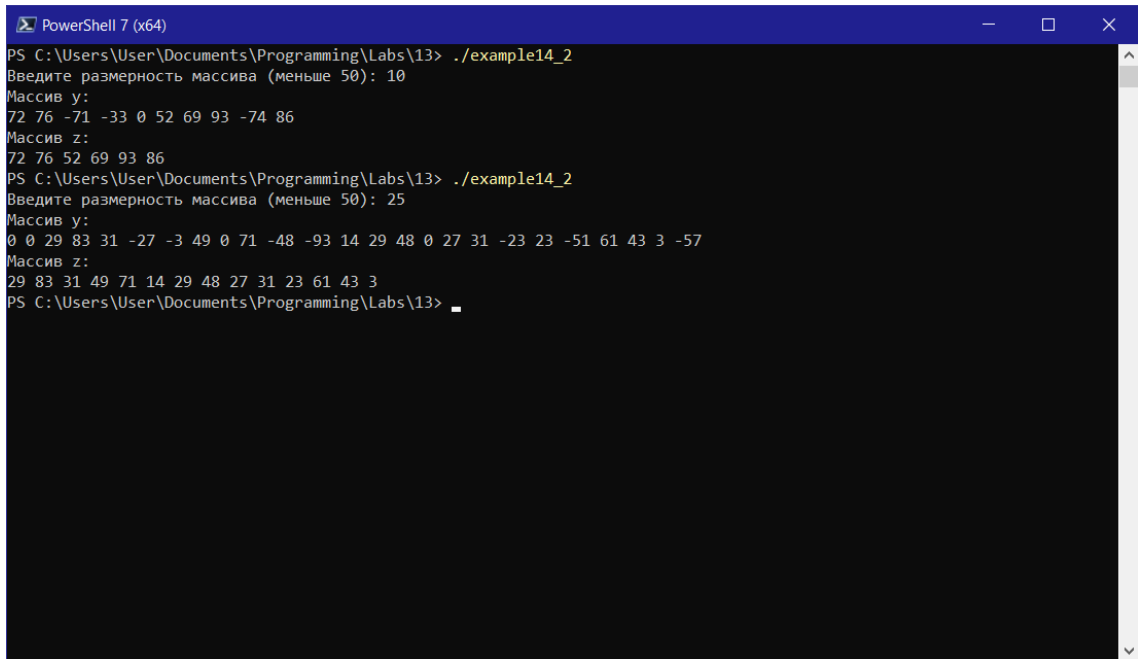
begin
    randomize;
    write(' Введите размерность массива (меньше 50):');
    readln(n);
    for i := 1 to n do begin
        y[i] := random(200) - 100;
        if ((y[i] mod 5) = 0) then y[i] := 0;
    end;
    k := 0;
    for i := 1 to n do
        if (y[i] > 0) then begin
            k := k + 1;
            z[k] := y[i];
        end;

    writeln(' Массив y:');
    for i := 1 to n do write(y[i], ' ');
    writeln;
    writeln(' Массив z:');
    for i := 1 to k do write(z[i], ' ');
    writeln;
end.
```

Блок-схема алгоритма программы:



Результат выполнения программы (через PowerShell):

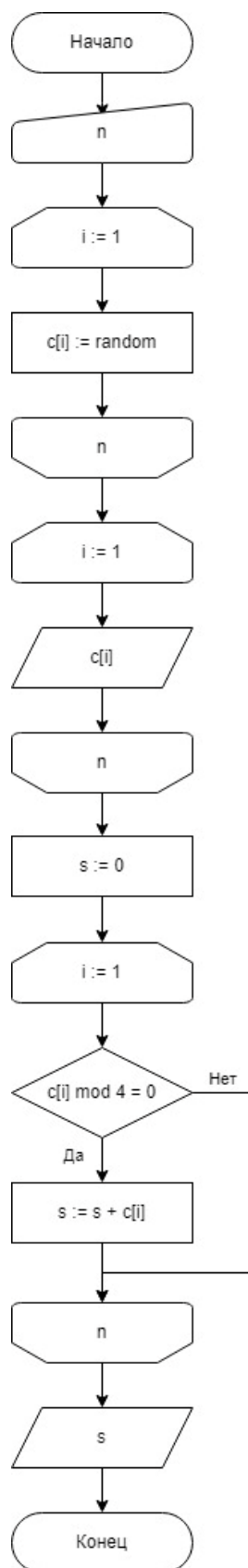


```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\13> ./example14_2
Введите размерность массива (меньше 50): 10
Массив y:
72 76 -71 -33 0 52 69 93 -74 86
Массив z:
72 76 52 69 93 86
PS C:\Users\User\Documents\Programming\Labs\13> ./example14_2
Введите размерность массива (меньше 50): 25
Массив y:
0 0 29 83 31 -27 -3 49 0 71 -48 -93 14 29 48 0 27 31 -23 23 -51 61 43 3 -57
Массив z:
29 83 31 49 71 14 29 48 27 31 23 61 43 3
PS C:\Users\User\Documents\Programming\Labs\13> _
```

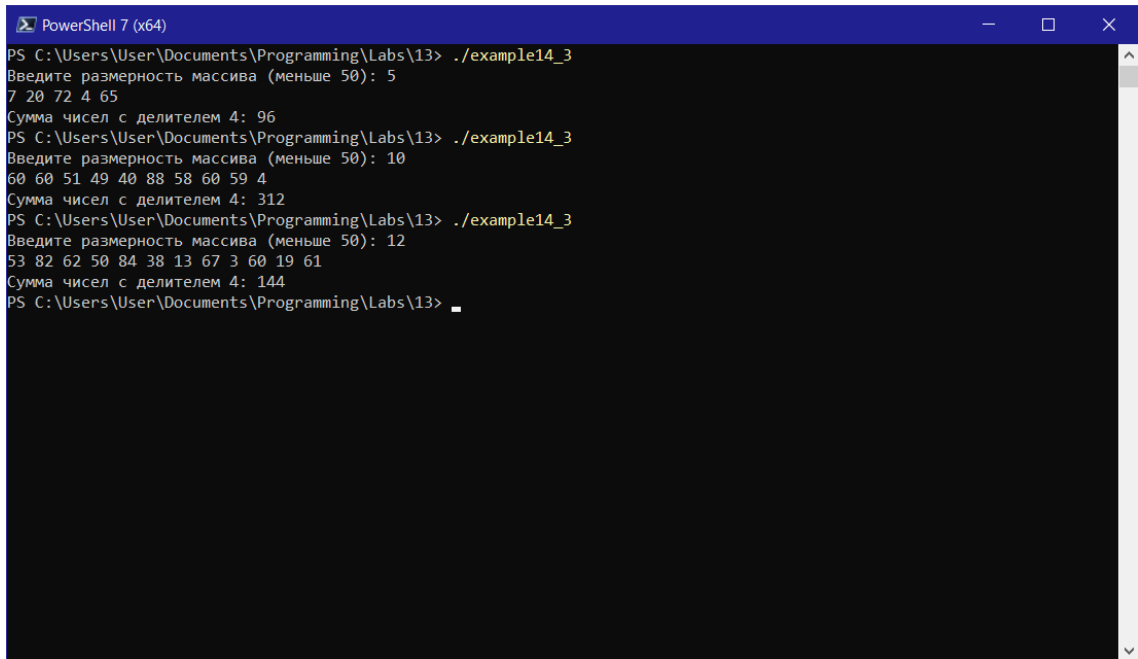
4. Выполнено решение 3-ой части задачи с вариантом № 3. Исходный код программы (файл 13/example14_3.pas):

```
program example14_3;
var c: array[1..50] of integer;
    i, j, n: byte;
    s: word;
begin
    randomize;
    write(' Введите размерность массива (меньше 50): ');
    readln(n);
    for i := 1 to n do begin
        c[i] := random(100);
    end;
    for i := 1 to n do write(c[i], ' ');
    writeln;
    s := 0;
    for i := 1 to n do
        if (c[i] mod 4 = 0) then s := s + c[i];
    writeln(' Сумма чисел с делителем 4: ', s);
end.
```


Блок-схема алгоритма программы:



Результат выполнения программы (через PowerShell):



```
PowerShell 7 (x64)
PS C:\Users\User\Documents\Programming\Labs\13> ./example14_3
Введите размерность массива (меньше 50): 5
7 20 72 4 65
Сумма чисел с делителем 4: 96
PS C:\Users\User\Documents\Programming\Labs\13> ./example14_3
Введите размерность массива (меньше 50): 10
60 60 51 49 40 88 58 60 59 4
Сумма чисел с делителем 4: 312
PS C:\Users\User\Documents\Programming\Labs\13> ./example14_3
Введите размерность массива (меньше 50): 12
53 82 62 50 84 38 13 67 3 60 19 61
Сумма чисел с делителем 4: 144
PS C:\Users\User\Documents\Programming\Labs\13> _
```

Ответы на контрольные вопросы.

1. Что необходимо изменить в программе примеров 9, 10, 11 для решения Вашего варианта текущей лабораторной работы?

Пример 9: изменилось условие в операторе ветвления, теперь на четность проверяется индекс, а не сам элемент массива. Так же, для сравнения выводятся оба массива.

Пример 10: после ввода данных (рандомно))) проход цикла по массиву происходит только 1 раз вместо 3, так как ищем только положительные числа, без нулевых и отрицательных. При выводе массива положительных чисел используется счетчик k, росший только при нахождении положительного элемента.

Пример 11: процедура нахождения чисел с делителем равным 4 проще, так как достаточно проверить, что остаток от деления элемента на 4 равен 0, флаг простого числа тем более не нужен, поэтому достаточно одного пробега цикла по массиву для нахождения суммы.

2. Как изменится схема алгоритма?

Новые алгоритмы представлены соответственно в пунктах 2, 3, 4 данной работы для сравнения.