Бюджетное учреждение профессионального образования

Ханты-Мансийского автономного округа-Югры

«Нижневартовский социально-гуманитарный колледж»

**Ульянова О.В.**

**Практикум**

**по информационным технологиям**

**для специальности**

**«Программирование в компьютерных системах»**

Нижневартовск

2016

Печатается по решению редакционно-издательского совета

БУ «Нижневартовский социально-гуманитарный колледж»

**Ульянова О.В.**

**Практикум по информационным технологиям для специальности «Программирование в компьютерных системах»**: учебное пособие / О.В. Ульянова – Нижневартовск: изд-во Нижневартовского социально-гуманитарного колледжа, 2016. – 32с.

В пособии представлен теоретический материал, а также практические задания для отработки навыков программирования.

Предназначено для обучающихся СПО специальности «Программирование в компьютерных системах».

|  |  |
| --- | --- |
|  | **©**Ульянова О.В., 2016©Нижневартовский социально-гуманитарный колледж, издание, 2016 |

**Содержание**

|  |  |
| --- | --- |
| Тема 1. Введение……………………………………………………… | 4 |
| Тема 2. Базовые алгоритмические структуры……………………… | 6 |
| Тема 3. Язык программирования……………………………………. | 8 |
| Тема 4. Язык программирования Паскаль………………………….. | 9 |
|  | Тема 4.1 Структура программы на языке Паскаль……………… | 10 |
|  | Тема 4.2 Операторы присваивания, ввода и вывода……………. | 11 |
|  | Тема 4.3 Алгоритмы линейной структуры………………………. | 12 |
|  | Тема 4.4 Алгоритм ветвление…………………………………….. | 13 |
|  | Тема 4.5 Циклы с параметром……………………………………. | 16 |
|  | Тема 4.6 Цикл с постусловием. Оператор повтора Repeat……… | 17 |
|  | Тема 4.7 Цикл с предусловием. Оператор с предусловием While | 19 |
|  | Тема 4.8 Рисование в графическом режиме TurboPascal. Модуль Graph……………………………………………………… | 21 |
| Задания для самостоятельного решения……………………………. | 23 |
| Библиографические ссылки………………………………………….. | 26 |
| Приложение ………………………………………………………….. | 27 |

## Тема 1. Введение

**Алгоритм** - точное предписание исполнителю совершить определенную последовательность действий для достижения поставленной цели за конечное число шагов.

**Свойства алгоритмов**

• **Дискретность** (прерывность, раздельность) – алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов. Каждое действие, предусмотренное алгоритмом, исполняется только после того, как закончилось исполнение предыдущего.

• **Определенность** – каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

• **Результативность (конечность)** – алгоритм должен приводить к решению задачи за конечное число шагов.

• ***Массовость*** – алгоритм решения задачи разрабатывается в общем виде, то есть он должен быть применим для некоторого класса задач, различающихся только исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

На основании этих свойств иногда дается определение алгоритма, например:«Алгоритм – это последовательность математических, логических или вместе взятых операций, отличающихся детерминированностью, массовостью, направленностью и приводящая к решению всех задач данного класса за конечное число шагов».

**Виды алгоритмов**

* **Линейный алгоритм** – набор команд (указаний), выполняемых последовательно во времени друг за другом.

• **Разветвляющийся** алгоритм – алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов.

• ***Циклический алгоритм*** – алгоритм, предусматривающиймногократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными. К циклическим алгоритмам сводится большинство методов вычислений, перебора вариантов.

**Цикл программы** – последовательность команд (серия, тело цикла), которая может выполняться многократно (для новых исходных данных) до удовлетворения некоторого условия.

**Исполнитель алгоритма** – это тот объект или субъект, для управления которым составлен алгоритм.

Система команд исполнителя (СКИ) – это вся совокупность команд, которые исполнитель умеет выполнять.

Формы представления алгоритмов:

* словесная;
* графическая;
* программная.

1. *Словесная форма – это форма описания алгоритма на естественном языке.*

Данная форма очень удобна, если нужно приближенно описать суть алгоритма. Однако при словесном описании не всегда удается ясно и точно выразить идею.

2. Для более наглядного представления алгоритма используется графическая форма, которая представляет собой *изображение алгоритма в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.*

3. При записи алгоритма в словесной и в графической форме допускается определенный произвол при изображении команд. Вместе с тем, такая запись точна на столько, что позволяет человеку понять суть дела и исполнить алгоритм. Однако на практике в качестве исполнителей алгоритмов используются специальные автоматы – компьютеры. Поэтому алгоритм, предназначенный для исполнения на компьютере, должен быть записан на понятном ему языке. Такой язык принято называть языком программирования, а форму представления алгоритма - программной.

## Тема 2. Базовые алгоритмические структуры

**Базовая структура«следование».**

Образуетсяпоследовательностью действий, следующих одно за другим (рис. 1).



Рисунок 1 – Базовая структура «следование»

**Базовая структура «ветвление»**. Обеспечивает в зависимости от результата проверки условия (**да** или **нет**) выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к **общему выходу**, так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран. Структура «**ветвление»** существует в четырех основных вариантах (табл. 1).

Таблица 1 –Основные варианты структуры «**ветвление»**

|  |  |
| --- | --- |
| если—то | C:\Users\yeremeyeva_os\Desktop\Снимок.PNG |
| если—то—иначе | C:\Users\yeremeyeva_os\Desktop\Снимок1.PNG |
| выбор | C:\Users\yeremeyeva_os\Desktop\Снимок2.PNG |
| выбор—иначе | C:\Users\yeremeyeva_os\Desktop\Снимок3.PNG |

**Базовая структура «цикл».** Обеспечивает **многократное выполнение некоторой совокупности действий,** которая называется **телом цикла** (табл. 2).

Таблица 2 – Базовая структура **«цикл»**

|  |  |
| --- | --- |
| Цикл типа **пока.**Предписывает выполнять тело цикла до тех пор, пока выполняется условие, записанное после слова пока. | F:\Методическая работа\Ульянова\Ульянова\Снимок5.PNG |
| Цикл типа**для**.Предписывает выполнять тело цикла для всех значений некоторой переменной (параметра цикла) в заданном диапазоне. | F:\Методическая работа\Ульянова\Ульянова\Снимок6.PNG |

## Тема 3. Язык программирования

*Язык программирования*– формальная знаковая система, предназначенная для описания алгоритмов в форме, которая удобна для исполнителя (например, ЭВМ, т.е. компьютера).

Уровни языков программирования:

* машинные;
* машинно-ориентированные (ассемблеры);
* машинно-независимые (языки высокого уровня).

**Машинные языки и машинно-ориентированные языки**— это языки **низкого уровня**, требующие указания мелких деталей процесса обработки данных. Языки же **высокого уровня**имитируют естественные языки, используя некоторые слова разговорного языка и общепринятые математические символы. Эти языки более удобны для человека.

Языки высокого уровня делятся на:

* **процедурные (алгоритмические)** (Basic, Pascal, C и др.), которые предназначены для однозначного описания алгоритмов; для решения задачи процедурные языки требуют в той или иной форме явно записать процедуру ее решения;
* **логические** (Prolog, Lisp и др.), которые ориентированы не на разработку алгоритма решения задачи, а на систематическое и формализованное описание задачи с тем, чтобы решение следовало из составленного описания;
* **объектно-ориентированные** (ObjectPascal, C++, Java и др.), в основе которых лежит **понятие объекта, сочетающего в себе данные и действия над ними.** Программа на объектно-ориентированном языке, решая некоторую задачу, по сути описывает часть мира, относящуюся к этой задаче. Описание действительности в форме системы взаимодействующих объектов естественнее, чем в форме взаимодействующих процедур.

## Тема 4. Язык программирования Паскаль

Данный язык был разработан в 1968-1971гг. профессором Института информатики Швейцарской высшей политехнической школы **Никлаусом Виртом**.

*Синтаксис языка*

Набор слов, используемых в Паскале, можно разделить на три группы:

* зарезервированные слова;
* стандартные идентификаторы;
* идентификаторы пользователя.

***Идентификаторы*** – это имена величин, процедур, функций, объектов, используемых в программе.

Правила написания идентификаторов.

1. Идентификатор начинается только с буквы латинского алфавита или знака подчеркивания.

Например, **Summa, \_A4**

1. Спецсимволы (**пробел , . + <**и т.п.) в идентификаторах недопустимы.
2. При написании идентификатора можно использовать как прописные, так и строчные буквы. Например, **REZ** и **Rez** – один и тот же идентификатор.

*Состав среды программирования*

Среда программирования любого языка программирования должна включать в себя следующие модули:

* ***текстовый редактор*** (для написания текстов программ);
* ***транслятор*** (для перевода текстов программ);
* ***отладчик*** (для исправления текстов программ);
* ***библиотеки готовых подпрограмм***(система помощи).

### Тема 4.1 Структура программы на языке Паскаль

**program** *название программы*;

**var**

*переменные: тип переменных***;**

**begin**

*тело программы***;**

**end.**

Правила написания программы на языке Паскаль

1. Логически законченные структуры программы разделяются точкой с запятой (**;**). После служебного слова **begin**точка с запятой не ставится.
2. Точка (**.**) ставится в конце программы после служебного слова **end**. Операторы, расположенные после точки, к тексту программы не относятся и при запуске программы не исполняются.
3. Для пояснений программисту в программе используются комментарии. ***Комментарий*** – это текст, заключенный в фигурные**{ }**или составные **(\* \*)** скобки. Комментарий текстом программы не является и при запуске программы пропускается.

### Тема 4.2 Операторы присваивания, ввода и вывода

**1) Оператор присваивания** придает переменной конкретное значение.

Синтаксис:

**Имя переменной:=выражение**

*Например*: x:=2; y:=5

**2) Оператор ввода на экран.**

**READ|READLN (список переменных);**

Список переменных – последовательность имен переменных, разделенных запятыми.

*Например*: READ (а,b);

При выполнении оператора программа останавливается и ждет, пока нужное количество чисел не будет введено. После ввода числового значения нужно нажать клавишу Enter или пробел.

В операторе READ числовые значения вводятся в одну строку, а в операторе READLN каждое числовое значение вводится с новой строки.

**3) Оператор вывода на экран.**

**WRITE|WRITELN (список выражений);**

Выражения в списке разделяются запятыми. В операторе WRITE выражения выводятся в одну строку, а в операторе WRITELN – в отдельные строки.

Например: WRITE (S);

Выражения вначале вычисляются, а потом высвечиваются на экране.

Чтобы прокомментировать выводимые значения, в список вывода можно помещать строки любых символов, заключенные в одинарные кавычки.

*Например*: WRITELN(‘Ответ:’ , Х , ‘км/сек’);

На экране высветится при х=3.5

Ответ: 3.5 км/сек

### Тема 4.3Алгоритмы линейной структуры

*Пример 1. Простейший алгоритм, запрашивающий имя и затем приветствующий его обладателя.*

ProgramHello;

VarName: String;  {Описание переменной Name строкового типа}

BEGIN

  Write('Как тебя зовут ? '); {Вывод на экран текста вопроса}

  ReadLn(Name); {Ввод c клавиатуры имени}

WriteLn('Привет, ', Name, '!'); {Вывод на экран приветствия}

  ReadLn

END.

***Задание. С***оздать чат общения с компьютером (5 вопросов)

*Пример 2.Найти произведение цифр заданного целого четырехзначного числа.*

ProgramDigitsProduct;

UsesCrt; {подключение модуля для очистки экрана}

VarNumber,   {заданное число}

 i, j, k, l,  {цифрычисла}

 P :Integer; {произведениецифр}

BEGINClrScr; {очисткаэкрана}

 Write( 'Введите четырехзначное число : ' ); ReadLn(Number);

  Number:=Abs(Number);

  Write( 'Цифрычисла ' , Number , ' : ' );

  i := Number div 1000; Write(i:3); {перваяцифра}

  j := Number div 100 mod 10; Write(j:3); {втораяцифра}

  k := Number div 10 mod 10; Write(k:3);  {третьяцифра}

  l := Number mod 10; WriteLn(l:3); {четвертаяцифра}

  P := i \* j \* k \* l ;

  WriteLn( 'О т в е т : произведение цифр равно ' , P );

  ReadLn

**END.**

***Задание.***Вычислить периметр и площадь прямоугольного треугольника по двум катетам.

### Тема 4.4Алгоритм ветвление

**if**<условие выбора>**then**<команды, выполняемые при соблюдении условия>**else**<команды, выполняемые при несоблюдении условия>

**ЕСЛИ**<условие выбора>**ТО**<команды, выполняемые при соблюдении условия>**ИНАЧЕ**<команды, выполняемые при несоблюдении условия>

**Особенности:**

* + перед ***else*НЕ** ставится точка с запятой
	+ вторая часть (***else*** …) может отсутствовать (неполная форма)
	+ если в блоке один оператор, можно убрать слова ***begin*** и ***end***

*Пример 1. Ввести два целых числа и вывести на экран наибольшее из них.*

programqq;

var a, b, max: integer;

begin

writeln('Введите два целых числа');

read ( a, b );

max := a;

if b > a then

max := b;

writeln ('Наибольшее число ', max);

end.

***Задание.***Определить, имеется ли среди заданных целых чисел (а, b, c) хотя бы одно четное.

Инструкция выбора**CASE – OF**

**Синтаксис:**

**Case**<выражение>**Of**

<значение выражения 1>: <оператор 1>;

< значение выражения 2>: < оператор 2>;

< значение выражения 3>: < оператор 3>;

…………

< значение выражения m>: < оператор m>

**else**

 оператор n

**end**;

Тип выражения должен быть **Integer** или **Char**, <значение выражения 1>, …<значение выражения m> - возможные значения выражения.

Данная инструкция работает следующим образом:

Проверяемые значения выражения задаются в виде отдельных чисел и интервалов. Интервал задается через указанные минимальные и максимальные его значения, указанные через**..**

Если нужно выполнить оператор для разных значений проверяемого значения, то эти значения могут быть записаны через запятую. После перечня значений пишется**:**, а затем идет оператор, который выполняется, если значение проверяемого выражения попадает в перечень. Если же значение выражения не попадает ни в один из перечней, то выполняется оператор после **Else**, если он имеется.

Раздел **Else** может быть пропущен. Инструкция заканчивается служебным словом **End**.

Если одно и тоже значение указано в нескольких перечнях, то выполняется оператор, который соответствует перечню, где это значение встречается впервые. Затем управление передается на первую команду после окончания инструкции выбора.

*Пример 2.Написать программу, которая запрашивает у пользователя номер месяца и выводит соответствующее название времени года. В случае если пользователь укажет недопустимое число, программа должна вывести сообщение «Ошибка ввода данных».*

Var

M: integer; {номермесяца}

Begin

Writeln (‘введитеномермесяца’);

Readln(m);

Case m of

1,2,12: writeln (‘Зима’);

3..5: writeln (‘Весна’);

6..8: writeln (‘Лето’);

9..11: writeln (‘Осень’)

else

writeln (‘Число должно быть от 1 до 12’);

end;

readln

end.

***Задание.***Написать программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: «Рабочий день», «Выходной».

***Задание.***Написать программу, вычисляющую стоимость междугороднего телефонного разговора (цена одной минуты определяется расстоянием до города, в котором находится абонент). Исходными данными для программы являются код города и продолжительность разговора.

|  |  |  |
| --- | --- | --- |
| Город | Код | Цена минуты, руб. |
| Владивосток | 423 | 2,2 |
| Москва | 095 | 1,0 |
| Мурманск | 815 | 1,2 |
| Самара | 846 | 1,4 |

###

### Тема 4.5Циклы с параметром

Инструкцию **For** на языке Паскаль можно реализовать двумя способами.

1 способ (с увеличением счетчика)

**for параметр:=**начальное значение**to**конечное значение**do**

**оператор1…n;**

Ключевые слова **For, do** обозначают «**для**», «**выполняй**» соответственно. Строка, содержащая **for…do**, называется **заголовком** цикла, оператор, стоящий после **do** образует его **тело**. Очень часто тело цикла – составной оператор. Параметр цикла не может иметь тип **Real.** Применяется переменная типа**Integer**или**Char.** В этих операторах нет шага изменения параметра. Он в данной форме постоянен и равен 1.

*Например:*ForI:=10 to 14 dowrite(I:3);

Выведется на экран последовательность цифр в виде:

10 11 12 13 14

2 способ (с уменьшением счетчика)

**for параметр:=**начальное значение**downto**конечное значение**do**

**оператор1…n;**

*Например*: for i:=14 downto 10 do write(I:3);

Выведется на экран последовательность цифр в виде:

14 13 12 11 10

*Например*: for ch:=’e’ to ‘a’ do write(ch:2);

Выведется на экран последовательность букв в виде:

Edcba

Если в цикле необходимо выполнить несколько инструкций, то они объединяются после **DO** операторными скобками **BEGIN – END.**

**for параметр:=**начальное значение**to**конечное значение**do**

**BEGIN**

**Оператор1;**

**Оператор2;**

**Оператор n;**

**End;**

*Пример 1.Найти сумму чисел на промежутке от 1 до 20.*

Program fordo1;

Uses crt;

Vars,I:integer;

Begin

For i:=1 to 20 do s:=s+I;

Writeln (‘Сумма чисел равна ’, s);

Repeatuntilkeypressed

End.

***Задание.***Найти сумму чисел на промежутке от М до N.

***Задание.***Найти сумму всех четных чисел в промежутке от M до Т**.**

**Тема 4.6Цикл с постусловием. Оператор повтора Repeat**

**Синтаксис оператора:**

**Repeat**

Оператор

**Until**условие выхода из цикла**;**

Таким образом, цикл повтора состоит из:

* Заголовка repeat
* Тела цикла (операторов – повторяющихся действий)
* Условия выхода из цикла после ключевого слова until.

**Порядок выполнения действий:**

Вначале выполняется действие (оператор) после слова repeat, затем проверяется значение условия выхода из цикла. Если условие принимает значение False, то действие повторяется еще раз. И так до тех пор, пока условие не примет значение True.

Следует знать, что:

* Число повторений операторов цикла определяется в ходе выполнения программы и во многих случаях заранее неизвестно;
* Операторы цикла будут выполняться до тех пор, пока условие, стоящее после until, будет оставаться ложным;
* После слова until записывается условие завершения цикла;

Условие – это выражение логического типа: простое выражение отношения или сложное логическое выражение;

Цикл заканчивается, когда логическое выражение принимает значение True. Тело цикла с постусловием выполняется хотя бы один раз.

*Пример 1.Вычислить сумму натурального ряда чисел от 1 до N.*

Program Work\_1;

Uses Crt;

Vara,Sum,n: integer;

Begin

Writeln(‘Введите натуральное N :’);

Readln(n);

a:=1;

Sum:=0;

repeat

 Sum:=Sum+a;

a:=a+1;

until a>n;

Writeln(‘Результатсуммирования : ’,Sum);

Readkey;

End.

***Задание.***Написать программу нахождения скорости V, по известному пути S и времени T (вводятся с клавиатуры). Предусмотреть ситуацию ввода значения T равной нулю, с выводом предупреждения о недопустимости деления на 0.

***Задание.*** Игрок должен угадать число, «задуманное» компьютером – случайное число в диапазоне от 0 до 1000*.*

**Тема 4.7Цикл с предусловием. Оператор с предусловием While**

**Синтаксис оператора:**

**While**Условие выполнения цикла **Do**

**Begin**

Оператор

**End;**

Таким образом, оператор повтора While состоит из:

* Заголовка While (до тех пор, пока) и Do (выполняй)
* Тела цикла (операторов)

Оператор цикла **While** аналогичен оператору **Repeat**, но проверка **Условия выполнения цикла** производится в самом начале оператора – если значение условия равно **True**, то выполняются операторы цикла, находящиеся между **Begin** и **End** и снова вычисляется выражение **Условие выполнения цикла**. Так продолжается до тех пор, пока значение **Условие выполнения цикла** не станет равно **False**.

**Следует знать:**

* Число повторений операторов цикла While определяется в ходе работы программы и, как правило, заранее неизвестно;
* После слова While записывается условие продолжения выполнения операторов цикла, в этом отличие цикла while от цикла repeat;

Условие – это выражение логического типа: простое выражение отношения или сложное выражение отношения (логическое выражение), которое может принимать одно из двух значений: ложь или истина.

*Например*, задача суммирования чисел от 1 до 10.

…

s:=0; i:=1;

 **while**i<=10 **do** ‘находим сумму чисел от 1 до 10

 **begin**

 s:=s+i ;

 i:=i+1; ‘изменение переменной управления циклом

 **end**;

…

*Пример 1. Написать программу, которая должна вывести на экран две строки чисел от 1 до 10 в прямом и обратном порядке.*

Program Work\_2;

Uses Crt;

Var I: integer;

Begin

I:0;

While i<10 do

Begin

 I:=succ(i); ‘можноиспользоватьinc(i); или i:=i+1

Write(I, ‘ ‘);

End;

Writeln;

I:=10;

While i>0 do

Begin

Write(I, ‘ ‘);

 I:=pred(i); можноиспользоватьdec(i); или i:=i-1

End;

End.

***Задание.*** Написать программу определения всех совершенных чисел в интервале от 2 до N. Натуральное число, равное сумме всех своих делителей, включая единицу, называется совершенным. Например: 6 = 1 + 2 + 3.

**Тема 4.8Рисование в графическом режиме TurboPascal.Модуль Graph**

Монитор персонального компьютера может работать в двух режимах: текстовом и графическом. Все, что вы делали до сих пор, вы делали в текстовом режиме. Текстовый экран содержит **2000***знакомест* - **25** строк по **80** позиций, в каждом знакоместе может быть выведен один символ. Графический экран состоит из маленьких точек - *пикселов*, каждый из которых закрашен в какой-либо цвет.

Графический режим отличается:

* Размером графической сетки (Х х У, где Х – число точек по горизонтали, У – число точек по вертикали);
* Цветностью (число воспроизводимых на экране цветов);
* Любое изображение в графическом режиме строится с использованием системы координат, в которой каждый пиксель имеет две координаты (Х,У). Программист может раскрасить каждый пиксель любым цветом. Начало координат находится в верхнем левом углу (0,0). В зависимости от разрешения экрана максимальные координаты Х и У будут равны предпоследнему значению разрешения. Например, при разрешении экрана 640 х 480 Х=639, а У=479.

**Инициализация графического режима**

**Program pr;**

**Usescrt, graph; {использование графического режима}**

**Vard, m:integer; {графический драйвер}**

**Begin**

**ClrScr; {очистка экрана}**

**Initgraph(d, m,’ ’);**

**End.**

*Пример1.Нарисовать в центре экрана на желтом фоне точку, окрашенную в синий цвет.*

Uses graph;

Var driver, mode:integer;

Var path: String;

Var x, y: integer;

Begin

 Driver:=detect;

InitGraph(driver, mode, ‘d:\program Files\tp7’); ‘инициализацияграфрежима

X:=GetMaxXdiv 2; ‘определение координат центра экрана

 Y:=GetMaxYdiv 2;

 SetBkColor (Yellow); ‘ заливка экрана желтым цветом и очистка экрана

ClearDevice;

PutPixel(X,Y, Blue); ‘ рисует точку синим цветом

Readln;

CloseGraph; ‘ закрытиеграфич режима

End.

***Задание.***Нарисовать следующую фигуру:

***Задания для самостоятельного решения***

1. Составить программу вычисления выражения:

****

1. Составить программу вычисления выражения:

, где ***r= p/100***

1. Составить программу вычисления корней квадратного уравнения по формулам

, .

Все аргументы ***a,b*** и значение дискриминанта ***D***вводит пользователь с клавиатуры.

1. Составить программу вычисления площади ромба по стороне и острому углу по формуле

***S = a2sinf***,

 где ***а*** – сторона ромба, ***f***– острый угол.

1. Составить программу вычисления среднего значения четырех чисел. Все числа вводит пользователь. Среднее значение должно быть выведено на экран в формате ***\*\*,\*\*.***
2. Составить программу вычисления среднего возраста семьи, состоящей из пяти человек (мама, папа, дедушка, бабушка, я). Возраст каждого члена семьи вводит пользователь с клавиатуры.
3. Составить программу вычисления стоимости обоев для одной стены. Известно, что высота стены ***k***метров, а длина ее ***– l***метров. Рулон обоев имеет длину 10 метров и ширину 1 метр, а стоимость одного рулона составляет ***х*** рублей.
4. Написать программу, которая проверяет введенную пользователем категорию товара и в зависимости от введенной категории сообщает размер скидки в процентах: для 1 категории – 5%, для 2 – 10%, для 3 – 15%, для 4 – 20%.
5. Написать программу, которая проверяет введенные пользователем категорию покупателя, сумму к оплате и в зависимости от введенных значений выводит сообщение о размерах скидки и сумму со скидкой, рассчитывая ее по правилу: 1 категории – скидка 5%, для 2 – 10%, для 3 – 15%, для 4 – 20%.
6. Написать программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 999), обозначающего денежную единицу, дописывает слово «рубль» в правильной форме. Например, 21 рубль, 22 рубля, 12 рублей и т.д.
7. Написать программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 99), обозначающего денежную единицу, дописывает слово «копейка» в правильной форме. Например, 5 копеек, 22 копейки, 41 копейка и т.д.
8. Дано целое число А. Найти сумму целых чисел из промежутка от 1 до А.
9. Найти произведение целых чисел, больших 10 и меньших 20.
10. Составить программу, которая запрашивает последовательно 5 целых чисел и вычисляет их сумму.
11. Среди всех двузначных чисел выбрать и вывести те, которые делятся на 3.
12. Среди всех трехзначных чисел выбрать и вывести те, которые заканчиваются на цифру К.
13. Среди всех двузначных чисел выбрать и вывести только кратные 8.
14. Составить программу, позволяющую вводить с клавиатуры целые числа до тех пор, пока не будет введено число –38, и вычисляющую произведение введенных чисел.
15. Вычислить сумму всех натуральных нечетных чисел, меньших заданного числа n.
16. Составить программу, позволяющую вводить целые числа до тех пор, пока не будет введено число –7.

## Библиографические ссылки

1. TurboPascal/С.А. Немнюгин. –СПб.: Издательство "Питер", 2001.
2. TurboPascal: практикум/С.А. Немнюгин. – СПб.: Питер, 2002.
3. Информатика. Задачник-практикум в 2т./Под ред. И.Г. Семакина, Е.К. Хеннера: Том 1. – М.: Лаборатория базовых знаний, 2001.
4. Программно-методический комплекс по курсу «основы информатики и вычислительной техники» часть1-4. Руководство для ученика. – КНПП БИТ, 1994
5. Рапаков Г.Г., Ржеуцкая С.Ю. TurboPascal для студентов и школьников. – СПб.: БХВ – Петербург, 2002
6. Семакин И.Г., Шестаков А.П. Основы программирования: Учебник. – М.: Мастерство; НМЦ СПО; Высшая школа, 2001. – 432 с.

## Приложение

**Таблица стандартных функций школьного алгоритмического языка**

|  |  |
| --- | --- |
| **Название и математическое обозначение функции** | **Указатель функции** |
| Абсолютная величина (модуль)  | | х | | abs(x) |
| Корень квадратный | $$\sqrt{х}$$ | sqrt(x) |
| Натуральный логарифм | ln x | ln(x) |
| Десятичный логарифм | lg x | lg(x) |
| Экспонента (степень числа е ~ 2.72) | ex | exp(x) |
| Знак числа x (- 1,  если  х<0;   0,  если x = 0;  1,  если x > 0) | sign | sign(x) |
| Целая часть х (т.е. максимальное целое число,не превосходящее х) |  | int(x) |
| Минимум из чисел х и y |  | min(x,y) |
| Максимум из чисел х и y |  | max(x,y) |
| Частное от деления целого х на целое y |  | div(x,y) |
| Остаток от деления целого х на целое y |  | mod(x,y) |
| Случайное число в диапазоне от 0 до х - 1 |  | rnd(x) |
| Синус (угол в радианах) | sin x | sin(x) |
| Косинус (угол в радианах) | cos x | cos(x) |
| Тангенс (угол в радианах) | tg x | tg(x) |
| Котангенс (угол в радианах) | ctg x | ctg(x) |
| Арксинус (главное значение в радианах) | arcsin x | arcsin(x) |
| Арккосинус (главное значение в радианах) | arccos x | arccos(x) |
| Арктангенс (главное значение в радианах) | arctg x | arctg(x) |

**Таблица «Типы данных»**

|  |  |  |  |
| --- | --- | --- | --- |
| **Идентификатор** | **Длина (байт)** | **Диапазон значений** | **Операции** |
| **Целые типы** |
| integer | 2 | -32768..32767 | +, -, /, \*, Div, Mod,>=, <=, =, <>, <, > |
| byte | 1 | 0..255 | +, -, /, \*, Div, Mod,>=, <=, =, <>, <, > |
| word | 2 | 0..65535 | +, -, /, \*, Div, Mod,>=, <=, =, <>, <, > |
| shortint | 1 | -128..127 | +, -, /, \*, Div, Mod,>=, <=, =, <>, <, > |
| longint | 4 | -2147483648..2147483647 | +, -, /, \*, Div, Mod,>=, <=, =, <>, <, > |
| **Вещественные типы** |
| real | 6 | 2,9x10-39 - 1,7x1038 | +, -, /, \*,>=, <=, =, <>, <, > |
| single | 4 | 1,5x10-45 - 3,4x1038 | +, -, /, \*,>=, <=, =, <>, <, > |
| double | 8 | 5x10-324 - 1,7x10308 | +, -, /, \*,>=, <=, =, <>, <, > |
| extended | 10 | 3,4x10-4932 - 1,1x104932 | +, -, /, \*,>=, <=, =, <>, <, > |
| **Логический тип** |
| boolean | 1 | true, false | Not, And, Or, Xor,>=, <=, =, <>, <, > |
| **Символьный тип** |
| char | 1 | все символы кода ASCII | +,>=, <=, =, <>, <, > |

**Таблица процедуры для изображения графических примитивов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название процедуры** | **Назначение** | **Примечание** |
|  | **PROCEDURE SetBkColor**(*Color*:**Word**); | Устанавливает фо-новый цвет, после ее выполнения весь экран будет закра-шен в цвет *Color* | Цветовая палитра также зависит от типа монитора и вы-бранногографичес-кого режима, но стандартная палитра для цветного монито-ра включает 16 цветов |
|  | **PROCEDURE ClearDevice** | Очистка экрана |  |
|  | **PROCEDURE PutPixel**(*X,Y*: **Integer**;*Color*: **Word**); | Закрашивает пиксел с координатами *X*,*Y* цветом*Color* |  |
|  | **PROCEDURE SetColor**(*Color*:**Word**) | Устанавливает цвет линий | Выводимые на экран линии будут иметь цвет*Color* до выполнения следую-щей процедуры **SetColor** |
|  | **PROCEDURE SetLineStyle**(*LineStyle*, *Pattern*, *Thickness*: **Word**) | Устанавливает*стильлиний* | Действует для всех выводимых линий до выполнения **SetLineStyle** с други-ми аргументами. Параметр *LineStyle* может принимать следующие значения:0 - **SolidLn** - сплошная линия;1 - **DottedLn** - пунктирная линия;2 - **CenterLn** - штрих-пунктирнаялиния;3 - **DashedLn**- штриховая линия;4 - **UserBitLn** - линия, задаваемая программистом.Если стиль линии - **4**, то форма линии определяется вторым параметром проце-дуры - *Pattern*. Толщина линии мо-жет принимать всего два значения:1 -**NormWidth** - тонкая линия;3 - **ThickWidth**- жирная линия |
|  | **PROCEDURE Line**(*x 1*,*y1*,*x2*,*y2*: **Integer**) | Рисует отрезок пря-мой от точки с координатами *x1*,*y1* до точки *x2*,*y2* |  |
|  | **PROCEDURE LineTo**(*x*,*y*: **Integer**) | Рисует отрезок от текущей точки (те-кущего положения графического курсо-ра) до точки *x*,*y* |  |
|  | **PROCEDURE LineRel**(D*x*,D*y*: **Integer**) | Рисует отрезок от текущей точки до точки со смещением *Dx*,*Dy* |  |
|  | **PROCEDURE MoveTo**(*x*,*y*: **Integer**) | Перемещает графи-ческий курсор в точку*x*,*y* | Графический курсор не виден на экране, но ряд процедур использует текущее положение графичес-кого курсора |
|  | **PROCEDURE Rectangle**(*x1*,*y1*,*x2*,*y2*: **Integer**) | Рисует прямоуголь-ник, *x1*,*y1*–коорди-наты левого верхне-го угла,*x2*,*y2*– координаты правого нижнего угла |  |
|  | **PROCEDURE Bar (X1, Y1, X2, Y2)** | Рисует заштрихо-ванныйпрямоу-гольник с координа-тами (X1, Y1) –верхний левый угол и (X2, Y2) – нижний правый угол | Используется стан-дартный цвет и стиль заливки |
|  | **PROCEDURE FloodFilll(X,Y, Border\_Color)** | Заливка произволь-ной замкнутой об-ласти с цветом границ **Border\_Color. X,Y –** координаты лю-бой внутренней точки |  |
|  | **PROCEDURE Circle**(*X*,*Y*: **Integer**; *R*: **Word**) | Рисует окружность радиуса *R* с центром в точке *X*,*Y* |  |
|  | **PROCEDURE FillEllipse**(*X*,*Y*: **Integer**;*Rx*,*Ry*: **Word**) | Рисуетзакрашен-ныйэллипс |  |
|  | **PROCEDURE PieSlice**(*X*,*Y*: **Integer**; *f1*,*f2*,*R*: **Word**); | Рисует закрашен-ный круговой сек-тор |  |

**Учебное пособие**

**Ульянова Ольга Викторовна**

**Практикум**

**по информационным технологиям**

**для специальности**

**«Программирование в компьютерных системах»**

Подписано в печать 06.06.2016 года.

Формат А 5. Бумага для множительных аппаратов.

Гарнитура Таймс. Усл. печ. листов 2

**Тираж 40 экз. Заказ 52**

Вёрстка и дизайн: Мечев О.А.

Отпечатано в БУ «Нижневартовский социально-гуманитарный колледж»

628602, Тюменская область,

г. Нижневартовск, ул. Дружбы Народов 13А