Введение

Язык формирует наш способ
мышления и определяет то,
о чем мы можем мыслить.
Б.Л Ворф

Мы живём во время бурного развития высоких технологий и компьютерной отрасли в целом. Каждый день появляются новые устройства и приборы, призванные облегчить жизнь человека, автоматизировать, столь утомительные для людей, монотонные многократно повторяющиеся действия… Компьютеры появляются в каждом доме и прочно входят в жизнь людей. Они многократно упрощают работу, процесс общения между людьми и просто являются развлекательными центрами всего дома. В тяжёлой промышленности используются специальные роботы - помощники. Они выполняют за людей опасную и трудоёмкую работу. Неоценим вклад компьютерных технологий в облегчение жизни людей!

Но какими бы гениальными небыли инженеры и конструкторы, создавшие робота или инновационный микропроцессор, их устройства не «оживут» без программного обеспечения! Ведь именно от программы зависит безошибочность, продуктивность, да и работа устройства в целом. Над созданием нового, более совершенного программного обеспечения работают программисты.

Но какой бы сложной не была программа, все равно в ней найдут место базовые алгоритмические конструкции такие как: ввод – вывод, разнообразные расчёты и преобразования, запись и чтение информации и т.д.

Цели моей работы:

- Изучить особенности языка программирования TURBO PASCAL
- Провести классификацию основных алгоритмических конструкций
- Написать программы на языке программирования TURBO PASCAL, реализующие основные алгоритмические конструкции

Глава 1. Язык программирования TURBO PASCAL

1.1. Общие данные.

На заре развития электронной вычислительной техники программы составлялись в машинных кодах, что предъявляло особые требования к квалификации программистов и обуславливало трудности в написании и восприятии уже готовых программ. Появление алгоритмических языков программирования постепенно сняло ореол таинственности с процесса программирования и сделало его доступным и понятным для широкого круга пользователей. Большое распространение получили такие языки, как FORTRAN, Algol-60, PL/1, Basic.

По мере развития вычислительной техники необходимость в привлечении к процессу написания программ широкого круга специалистов потребовала разработать язык «для обучения программированию как систематической дисциплине». В качестве такого языка профессором Федерального технологического института в Цюрихе Н.Виртом был предложен язык Pascal, который создавался в 1968-1971 годах.

Развитие и совершенствование языка привело к появлению в 1979 году его стандартной версии, а также большого количества диалектов и прикладных библиотек.

Существенный вклад в распространение языка среди самого широко круга пользователей обеспечила компания Borland International, которая расширила стандарт языка Pascal и снабдила его мощным компилятором, фактически создав новую версию языка, получившего название Turbo Pascal. В 1985г. Эта компания предложила свою версию языка для персональных компьютеров (версия Turbo Pascal 3.0). Разработанная позже версия Turbo Pascal 4.0 характерна тем, что в ней язык программирования был соединен с текстовым редактором. В дальнейшем язык совершенствовался в соответствии с требования текущего момента и тенденциями развития программирования.

В настоящее время язык программирования Turbo Pascal 7.0 остаётся одним из наиболее распространенных языков, объединяя в себе большие возможности и удобство интерфейса пользователя. Его основные конструкции включили в себя такие языки, как Object Pascal и Delphi.

1.2. История TURBO PASCAL.

Turbo Pascal - это среда разработки для языка программирования Паскаль. Используемый в Turbo Pascal диалект базировался на более раннем UCSD Pascal, получившем распространение, в первую очередь, на компьютерах серии Apple II.

Когда в 1983 году появилась первая версия Turbo Pascal, такой тип среды разработки был относительно новым. Во время дебюта на американском рынке, Turbo Pascal продавался по цене в 49,99$. Помимо привлекательной цены, встроенный компилятор Паскаля также был очень высокого качества. Приставка «Turbo» намекала как на скорость компиляции, так и на скорость производимого им исполняемого кода. Turbo Pascal создавал машинный код за один проход, без шага компоновки.

Для того времени это была потрясающая среда разработки. Она была проста и интуитивно понятна, с хорошо организованным меню. В более поздних версиях появилась возможность быстро получить определение ключевого слова языка, просто поставив курсор на ключевое слово и нажав клавишу справки. Справочные статьи часто включали примеры кода, использующего данное ключевое слово. Это позволяло неопытным программистам изучать Паскаль даже без помощи книг, используя лишь среду разработки. В поставку входило большое количество исходных текстов демонстрационных и прикладных программ. В их числе были даже шахматы.

Среда позволяла легко встраивать в код на Паскале вставки на языке ассемблера. Пользователь имел возможность проходить программу шаг за шагом; при переходе на ассемблерный блок это также работало. В любой момент пользователь мог добавить переменную или регистр в удобно

расположенное окно для наблюдения за ними. При построчной отладке программ, использующих графические режимы IBM PC, происходило корректное переключение между графическим режимом программы и текстовым режимом среды разработки.

Помимо всего этого, имелось средство профилирования. Книги, включённые в поставку Borland Pascal, давали детальное описание языка ассемблера Intel вплоть до указания количества тактовых циклов, необходимых для выполнения каждой инструкции. В общем и целом, система давала превосходные возможности для оптимизации кода; пользователю не требовалось пользоваться чем-либо кроме среды разработки. Всё было сделано так идеально, что даже школьник мог этим пользоваться. Эти качества позволили версии Паскаля от Borland стать стандартом языка Паскаль de facto.

С начала 1990-х Pascal используется в университетах для изучения фундаментальных концепций программирования.

В течение нескольких лет Borland улучшал не только среду разработки, но и язык. В версии 5.5 в него были введены передовые возможности объектно-ориентированного программирования. Последней выпущенной версией была версия 7.

1.3. Этапы развития Turbo Pascal.

- Turbo Pascal 1.0, 1983 год. Компилирует непосредственно в машинный код. Требует 32 килобайта оперативной памяти. Стоит меньше 50 долларов. Имеет интегрированный компилятор/редактор, высокую скорость компиляции. Позволяет размещать динамические данные в «куче»heap) — динамической области памяти.
- Turbo Pascal 2.0, 1984 год. Увеличен размер создаваемой программы — позволяет использовать до 64 килобайт для кода, стека и данных. Версия для DOS поддерживает арифметический сопроцессор и двоично-десятичную арифметику.

- Turbo Pascal 3.0, 1985 год. Поддержка графических режимов. Специальные подпрограммы формирования изображений для IBM PC совместимых компьютеров. Инструментальные пакеты.
- Turbo Pascal 4.0, 1987 год. Раздельная компиляция модулей. Размер программы ограничен только объёмом оперативной памяти. Управляемая с помощью меню интегрированная среда разработки. Интеллектуальная компоновка модулей. Автономный компилятор командной строки. Контекстно-чувствительная система помощи.
- Turbo Pascal 5.0, 1988 год. Встроенный отладчик. Отдельный отладчик. Эмуляция арифметического сопроцессора. Поддержка графических драйверов BGI (Borland Graphics Interface).
- Turbo Pascal 5.5, 1989 год. Объектно-ориентированное программирование. Возможность копирования в программу примеров из справочной системы. Электронный учебник на диске. Turbo Profiler — профилировщик, позволяющий оптимизировать код программы.
- Turbo Pascal 6.0, 1990 год. Библиотека Turbo Vision. Новая IDE, переписанная с использованием Turbo Vision, поддерживающая мышь и редактирование нескольких файлов одновременно в разных окнах. Встроенный ассемблер BASM позволяющий в исходном тексте программы делать ассемблерные вставки. Работающий в защищённом режиме компилятор командной строки.
- Borland Pascal 7.0, 1992 год. Выпущен Borland Pascal 7.0, включающий в себя более дешёвый и менее мощный Turbo Pascal 7.0 который поставлялся также отдельно. BP 7.0 позволял создавать программы под реальный и защищённый 16-битный режим DOS и Windows. Была введена поддержка открытых массивов. Открыты исходные тексты системных библиотек и функций времени выполнения.

Глава 2. Алгоритмы.

2.1. Общие данные.

Появление алгоритмов связывают с зарождением математики. Более 1000 лет назад (в 825 году) ученый из города Хорезма Абдулла (или Абу Джафар) Мухаммед бен Муса аль-Хорезми создал книгу по математике, в которой описал способы выполнения арифметических действий над многозначными числами. Само слово «алгоритм» возникло в Европе после перевода на латынь книги этого среднеазиатского математика, в которой его имя писалось как «Алгоритми».

В старой трактовке алгори́тм — это точный набор инструкций, описывающих последовательность действий исполнителя для достижения результата решения задачи за конечное время. По мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок». Это связано с тем, что какие-то действия алгоритма должны быть выполнены только друг за другом, но какие-то могут быть и независимыми.

Ранее часто писали «алгорифм», сейчас такое написание используется редко.

Часто в качестве исполнителя выступает некоторый механизм (компьютер, токарный станок, швейная машина), но понятие алгоритма необязательно относится к компьютерным программам, так, например, чётко описанный рецепт приготовления блюда также является алгоритмом, в таком случае исполнителем является человек.

Единого определения понятия «алгоритм» нет.

1)«Алгоритм — это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность». (Д. Э. Кнут)

2)«Алгоритм — это всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи». (А. Колмогоров)

3)«Алгоритм — это последовательность действий, либо приводящая к решению задачи, либо поясняющая, почему это решение получить нельзя».

4)«Алгоритм — строго детерминированная последовательность действий, описывающая процесс преобразования объекта из начального состояния в конечное, записанная с помощью понятных исполнителю команд». (Николай Дмитриевич Угринович, учебник «Информатика и информ. технологии»)

5) «Алгоритм — это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату». (А. Марков)

Помимо этих поянений термина «алгоритм» существует множество других, имеющих свои особенности и дополнения.

Особую роль выполняют прикладные алгоритмы, предназначенные для решения определенных прикладных задач. Алгоритм считается правильным, если он отвечает требованиям задачи (например, даёт физически правдоподобный результат). Алгоритм (программа) содержит ошибки, если для некоторых исходных данных он дает неправильные результаты, сбои, отказы или не дает никаких результатов вообще. Последний тезис используется в олимпиадах по алгоритмическому программированию, чтобы оценить составленные участниками программы.

Важную роль играют рекурсивные алгоритмы (алгоритмы, вызывающие сами себя до тех пор, пока не будет достигнуто некоторое условие возвращения). В последнее время активно разрабатываются параллельные алгоритмы, предназначенные для вычислительных машин, способных выполнять несколько операций одновременно.

Алгоритм — это точно определённая инструкция, последовательно применяя которую к исходным данным, можно получить решение задачи. Для

каждого алгоритма есть некоторое множество объектов, допустимых в качестве исходных данных. Например, в алгоритме деления вещественных чисел делимое может быть любым, а делитель не может быть равен нулю.

Алгоритм служит, как правило, для решения не одной конкретной задачи, а некоторого класса задач. Так, алгоритм сложения применим к любой паре натуральных чисел. В этом выражается его свойство массовости, то есть возможности применять многократно один и тот же алгоритм для любой задачи одного класса.

Для разработки алгоритмов и программ используется алгоритмизация — процесс систематического составления алгоритмов для решения поставленных прикладных задач. Алгоритмизация считается обязательным этапом в процессе разработки программ и решении задач на ЭВМ. Именно для прикладных алгоритмов и программ принципиально важны детерминированность, результативность и массовость, а также правильность результатов решения поставленных задач.

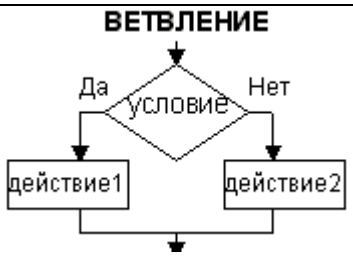
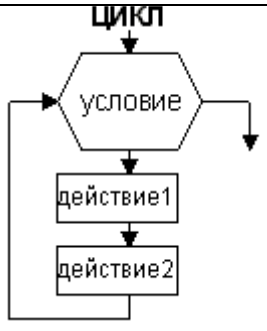
Алгоритм может быть записан словами и изображён схематически. Обычно сначала (на уровне идеи) алгоритм описывается словами, но по мере приближения к реализации он обретает всё более формальные очертания и формулировку на языке, понятном исполнителю (например, машинный код). Например, для описания алгоритма применяются блок-схемы. Другим вариантом описания, не зависимым от языка программирования, является псевдокод.

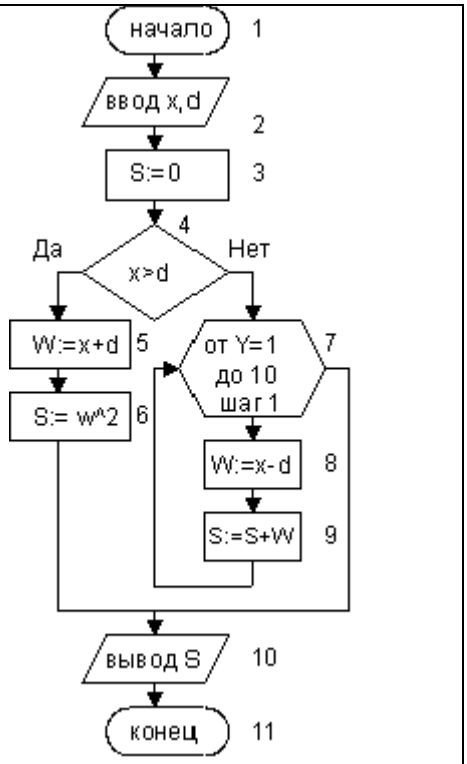
Хотя в определении алгоритма требуется лишь конечность числа шагов, требуемых для достижения результата, на практике выполнение даже хотя бы миллиарда шагов является слишком медленным. Также обычно есть другие ограничения (на размер программы, на допустимые действия). В связи с этим вводят такие понятия как сложность алгоритма (временная, по размеру программы, вычислительная и др.).

Для каждой задачи может существовать множество алгоритмов, приводящих к цели. Увеличение эффективности алгоритмов составляет одну из

задач современной информатики. В 50-х гг. XX века появилась даже отдельная её область — быстрые алгоритмы. В частности, в известной всем с детства задаче об умножении десятичных чисел обнаружился ряд алгоритмов, позволяющих существенно ускорить нахождение произведения.

2.2. Виды и свойства алгоритмов.

Виды алгоритмов	**Ключевые слова**	**Структура**
Линейный- описание действий, которые выполняются однократно в заданном порядке.	нет	
Разветвляющийся- алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.	если….то…иначе; при (в значении если…)…	
Циклический	от…до…; …раз; пока…; если (в значении пока…)	

Комбинированный	если….то…иначе; при (в значении если…)…; от…до…; …раз; пока…; если (в значении пока…); и т.д.	

Для правильного и продуктивного выполнения, алгоритм должен быть построен с учётом следующих особенностей:

1) дискретность (от лат. discretus — разделенный, прерывистый) – это разбиение алгоритма на ряд отдельных законченных действий (шагов);

2) детерминированность (от лат. determinate — определенность, точность) - любое действие алгоритма должно быть строго и недвусмысленно определено в каждом случае (например, если к остановке подходят автобусы разных маршрутов, то в алгоритме должен быть указан конкретный номер маршрута - 5. Кроме того, необходимо указать точное количество остановок, которое надо проехать, — скажем, три);

3) конечность - каждое действие в отдельности и алгоритм в целом должны иметь возможность завершения. Алгоритм должен иметь предел, то есть быть конечным;

4) массовость - один и тот же алгоритм можно использовать с разными исходными данными;

Например: алгоритм приготовления любого бутерброда.

1. Отрезать ломтик хлеба.

2. Намазать его маслом.

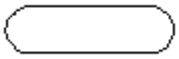
3. Отрезать кусок любого другого пищевого продукта (колбасы, сыра, мяса).

4. Наложить отрезанный кусок на ломоть хлеба.

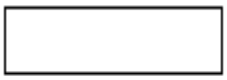
5) результативность - в алгоритме не должно быть ошибок;

Алгоритм может быть записан различными способами: на естественном языке в виде описания; в виде графических блок-схем; на специальном алгоритмическом языке Запись алгоритмов на родном языке доступна и удобна. Примеров таких записей множество, хотя бы книга кулинарных рецептов есть не что иное, как сборник алгоритмов, написанных на родном языке. Существенным недостатком такой записи является недостаточная наглядность, что особенно сказывается, когда алгоритм имеет много ветвлений. Поэтому, для наглядности используют специальные блок-схемы.

Элементы блок-схемы и их описания:

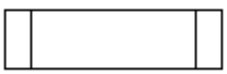


начало или конец. Внутри фигуры пишут «начало» или «конец» соответственно;



прямоугольником обозначается операция. Например, присваивание. Внутри блока пишут операции, которые выполняются на данном шаге алгоритмом;

ромбом обозначается оператор ветвления. Внутри ромба пишутся проверяемые условия. Например, "a < b»;



вызов подпрограммы. Внутри блока пишут имя вызываемой подпрограммы и передаваемые ей параметры;

параллелограмм обозначает операции ввода-вывода данных;

цикл с известным числом итераций. Внутри обычно указывают счетчик цикла, начальное, конечное значение и шаг цикла.

Глава 3. Типовые алгоритмы языка программирования TUBO PASCAL.

3.1. Циклы.

Основа алгоритма (если есть)	**Текст программы**

Обычный цикл
for…
	begin
	for n:=1 to 10 do
	writeln (sqr(n));
	end.

Цикл с предусловием
while условие do…
	while Abs(a)>epsilon do begin
	  sum:=sum+a;
	  a:=-a/2;
	end;

Цикл с постусловием
repeat … untill условие;
	repeat
	a:=a+1;
	untill a<100;

Цикл с параметром **for параметр:=… to … do …;**	for i:=2 to n div 2 do writeln (n);

3.2. Массивы.

<u>Случайное заполнение массива</u>
randomize;
for i:=1 to … do
a[i]:=random(…);
	var a: array [1..10] of integer;
	i:integer;
	begin
	randomize;
	for i:=1 to 10 do
	a[i]:= random(100);
	end.

<u>Вывод многомерного массива</u>
for i:=1 to … do begin
for j:=1 to … do
write (a[i,j],' ');
writeln;
end;
	a: array [1..n, 1..m] of integer;
	i,j: integer;
	begin
	randomize;
	for i:=1 to n do
	for j:=1 to m do
	a[i,j]:=random(q);
	for i:=1 to n do begin
	for j:=1 to m do
	write (a[i,j],' ');
	writeln;
	end;

Сортировка методом «пузырька»
for i:=1 to n-1 do
for j:=n downto i+1 do
if a[j]<a[j-1] then begin
b:=a[j];
a[j]:=a[j-1];
a[j-1]:=b
end;
	var
	a:array [1..n] of integer;
	i,b,j:integer;
	begin
	for i:=1 to n-1 do
	for j:=n downto i+1 do
	if a[j]<a[j-1] then begin
	b:=a[j];
	a[j]:=a[j-1];
	a[j-1]:=b;
	end;

Поиск MAX элемента массива
Imax:=1;
max:=a[1];
for i:=2 to n do
if max<a[i] then begin
max:=a[i];
Imax:=i;
end;
	var
	a: array [1..n] of integer;
	max: integer;
	Imax: integer;
	i: integer;
	begin
	Imax:=1;
	max:=a[1];
	for i:=2 to n do
	if max<a[i] then begin
	max:=a[i];
	Imax:=i;
	end;

Поиск отрицательного элемента	a: array [1..n] of integer;
	i, fl: integer;
	begin
	fl:=0;
	for i:=1 to n do
	if a[i]<0 then
	fl:=i;
	if fl>0 then

Вывод матрицы многомерного массива на экран
for i:=1 to … do begin
for j:=1 to … do
write (a[i,j],' ');
	a: array [1..n, 1..m] of integer;
	i,j: integer;
	begin
	for i:=1 to n do begin
	for j:=1 to m do
	write (a[i,j],' ');
	writeln;
	end;

Поиск количества чётных элементов	var
	a: array [1..n] of integer;
	i: integer;
	s: integer; {kol-vo elementov}
	begin
	s:=0;
	for i:=1 to n do
	if a[i] mod 2 = 0 then
	s:=s+1;
	writeln (Количество чётных элементов=',s);
	end.

s:=0;
for i:=1 to … do
if a[i] mod 2 = 0 then
s:=s+1;	

3.3. Строки.

Является ли строка палиндромом?
f:=true;
i:=1;
while f and (i<=length(s) div 2) do begin
if s[i] <> s[length(s)-i+1] then f:=false;
i:=i+1;
end;
	var
	s:string;
	i:integer;
	f:boolean;
	begin
	readln (s);
	i:=length(s);
	while i>=1 do begin
	if s[i]=' '
	then delete (s,i,1)
	else s[i]:=UpCase(s[i]);
	i:=i-1;
	end;
	f:=true;
	i:=1;
	while f and (i<=length(s) div 2) do begin
	if s[i] <> s[length(s)-i+1] then f:=false;
	i:=i+1;
	end;

Разложение строка на числа	s,s_L,s_r,s_Str,s1: string;
	k,p :integer;
	l: longint;
	r: real;
	begin
	readln (s);
	while s[1]=' ' do
	delete(s,1,1);
	while s[Length(s)]=' ' do
	delete(s,Length(s),1);
	while Pos(' ',s)<>0 do
	delete(s,Pos(' ',s),1);
	s_L:=' '; s_R:=' '; s_Str:=' ';
	while s<>'' do begin
	k:=Pos(' ',s);
	if k=0 then k:=Length(s)+1;
	s1:=Copy(s,1,k-1);
	Val(s1,l,p);
	if p=0 then s_l:=s_L+s1+' '
	else begin
	Val (s1,r,p);
	if p=0 then s_R:=s_R+s1+' '

	else s_Str:=s_Str+s1+' ';
	end;
	delete(s,1,k);
	end;
	Dec(s_l[0]); Dec(s_r[0]);
	Dec(s_Str[0]);
	Writeln;
	Writeln('Целые числа: ',s_L);
	Writeln('Действительные числа: ',s_r);
	Writeln('Слова: ',s_Str);
	end.

Замена слов в строке
Len:=Length(a);
Position:=Pos(a,b);
while Position<>0 do begin
delete(b,n,h);
insert(c,b,n);
Position:=Pos(a,b);
end;
	str, word1, word2: string;
	len, position: byte;
	begin
	write ('Введите слово-образец: ');
	readln (word1);
	write ('Введите слово заменитель`: ');
	readln (word2);
	write ('Введите исходную строку: ');
	readln (str);
	
	Len:=Length(Word1);
	Position:=Pos(Word1,Str);
	while Position<>0 do begin
	delete(Str,Position,Len);
	insert(Word2,Str,Position);
	Position:=Pos(Word1,Str);
	end;
	writeln (Str);
	end.

3.4. Работа с экраном.

Управление курсором
case c of
#72: if WhereX<>0 then
GotoXY (WhereX,WhereY-1);
#75: if WhereX<>1 then
GotoXY (WhereX-1,WhereY);
#77: if WhereX<>80 then
GotoXY (WhereX+1,WhereY);
#80: if WhereY<>25 then
GotoXY (WhereX,WhereY+1);
	uses Crt;
	var
	c:char;
	n:integer;
	begin
	ClrScr;
	c:=ReadKey;
	repeat
	if c=#0 then begin
	c:=ReadKey;

end;	case c of
	#72: if WhereX<>0 then
	GotoXY (WhereX,WhereY-1);
	#75: if WhereX<>1 then
	GotoXY (WhereX-1,WhereY);
	#77: if WhereX<>80 then
	GotoXY (WhereX+1,WhereY);
	#80: if WhereY<>25 then
	GotoXY (WhereX,WhereY+1);
	end;
	end;
	c:=ReadKey;
	until c=#27; **{пока не нажат ESC}**
	end.

Двигающийся текст
for i:=1 to 24 do begin
DelLine;
Delay (60000);
end;
for i:=1 to 24 do begin
InsLine;
Delay (60000);
end;
	uses Crt;
	var
	s:string[80];
	i:integer;
	begin
	ClrScr;
	Writeln('Введите текст');
	Read (s);
	ClrScr;
	GotoXY ((80-Length(s)) div 2-1, 25);
	write (s);
	GotoXY (1, 1);
	while not KeyPressed do begin
	for i:=1 to 24 do begin
	DelLine;
	Delay (60000);
	end;
	for i:=1 to 24 do begin
	InsLine;
	Delay (60000);
	end;
	end;
	end.

3.4. Файлы.

Вывод на экран содержимого текстового файла

```
while no Eof (a) do begin
readln (a, b);
writeln (b);
```

```
fil : text;
str : string;
begin
assign (fil, ‘work.txt’);
reset (fil);
while no Eof (fil) do begin
readln (fil, Str);
writeln (str);
end;
close (fil);
end.
```

Создание и копирование файла

а)создание нового файла

```
var i:integer
begin
        assign (a, ‘xxx.txt’);
        rewrite (a);
        for i:=1 to 3 do begin
        readln (c);
        writeln (a, c);
        end;
        close (a);
end;
```

б)копирование файлов

```
begin
        assign (a, ‘xxx.txt’)
reset (a);
assign (b, ‘yyy.txt’);
rewrite (b);
while not Eof (a) do begin
        readln (a, c);
writeln (b, c);
end;
close (a);
close (b);
end;
```

```
fil_1, fil_2: text;
str: string;
{процедура создания нового файла}
procedure New_file;
var i:integer
begin
        assign (fil_1, ‘work.txt’);
        rewrite (fil_1);
        for i:=1 to 3 do begin
        readln (Str);
        writeln (fil_1, str);
        end;
        close (fil_1);
end;
{процедура копирования файла}
procedure Copy_File;
begin
        assign (Fil_1, ‘work.txt’)
reset (fil_1);
assign (fil_2, ‘user.txt’);
rewrite (fil_2);
while not Eof (fil_1) do begin
        readln (Fil_1, str);
writeln (Fil_2, str);
end;
close (fil_1);
close (fil_2);
end;
begin
        new_file;
        copy_file;
end.
```

Запись матрицы вещественных чисел 5x4 в текстовый файл
For i:=1 to m do begin
For j:=1 to n do begin
A:=random(100);
Write (x, a:5:3, ‘ ‘);
End;
Writeln (x);
End;
	Const
	M=5; n=4; {размер матрицы}
	var
	fil:= text;
	A: real;
	S: char;
	I, j: integer;
	Begin
	Assign (fil, ‘matrix.txt’);
	Rewrite (fil);
	Randomize;
	For i:=1 to m do begin
	For j:=1 to n do begin
	A:=random(100);
	Write (fil, a:5:3, ‘ ‘);
	End;
	Writeln (fil);
	End;
	Close (fil);
	end.

Чтение матрицы с файла и вывод на экран
While not Eof (xxx) do begin
While not eoln(xxx) do begin
Read (xxx, a);
Write (a:5:3);
Read (xxx, s);
Write (s);
End;
Writeln;
Readln (xxx);
End;
	var
	fil:= text;
	a: real;
	s: char;
	begin
	assign (fil, ‘matrix.txt’);
	reset (fil);
	While not Eof (fil) do begin
	While not eoln(fil) do begin
	Read (fil, a);
	Write (a:5:3);
	Read (fil, s);
	Write (s);
	End;
	Writeln;
	Readln (fil);
	End;
	Close (fil);
	Close (fil);
	end.

3.5. Модуль Graph.

Заполнение экрана разноцветными точками.
Randomize;
Repeat;
color:= random(15);
PutPixel (random(100), Random (100), Clolor);
Delay (xxx);
	Uses graph, crt;
	var
	gd, gm: integer;
	color: byte;
	begin
	gd:= Detect;
	InitGraph (gd, gm, ‘’);
	If GraphResult <> grOK then
	Halt(1);
	Randomize;
	Repeat;
	color:= random(15);
	PutPixel (random(100), Random (100), Clolor);
	Delay (10);
	Untill key pressed;
	CloseGraph
	end.

Рисование линий в цикле
randomize;
repeat
SetColor (random(15));
line (random(200), random(200),random(200), random(200));
delay (10);
	uses Graph, crt;
	var
	gd, gm: integer;
	color: byte;
	begin
	gd:= detect;
	InitGraph (gd, gm, ‘’);
	if GraphResult <> grOK then
	halt(1);
	randomize;
	repeat
	SetColor (random(15));
	line (random(200), random(200),random(200), random(200));
	delay (10);
	until keypressed;
	readln;
	CloseGraph
	end.

Рисование разноцветных концентрических окружностей
randomize;
for r:=1 to 10 do begin
SetColor(random(16));
Circle(320,240,r*5);
end;
	uses Graph, crt;
	var
	gd, gm, r: integer;
	color: byte;
	begin
	gd:= detest;
	InitGraph (gd, gm, ‘’);
	if GraphResult <> grOK then
	halt(1);
	randomize;
	for r:=1 to 10 do begin
	SetColor(random(16));
	Circle(320,240,r*5);
	end;
	CloseGraph;
	end.

Заключение

Изобретение языка программирования позволило нам общаться с машиной, понимать её (если конечно Вам знаком используемый язык), как понимает американец немного знакомый с русским языком древнюю азбуку Кириллицы. Если мы обратим внимание на темпы роста и развития новейших технологий в области программирования, то можно предположить, что в ближайшем будущем, человеческие познания в этой сфере, помогут произвести на свет языки, умеющие принимать, обрабатывать и передавать информации в виде мысли, слова, звука или жеста. Так и хочется назвать это детище компьютеризированного будущего: «языки программирования "высочайшего" уровня».

Размышляя над этим, хочется верить в прогресс науки и техники, в высоко - компьютеризированное будущее человечества, как единственного существа на планете, пусть и не использующего один, определенный разговорный язык, но способного так быстро прогрессировать и развивать свой интеллект, что и перехода от многоязыковой системы к всеобщему пониманию долго ждать не придется.

В своей годовой работе я собрал самые важные и интересные алгоритмические конструкции языка программирования TURBO PASCAL, которые могли бы помочь другим ученикам в изучении информатики в целом и программирования в частности.

Кроме этого в моей годовой работе содержится самая важная информация относительно языка программирования TURBO PASCAL и алгоритмов, которая должна помочь учащимся в понимании сути программирования.

Единственный способ изучать новый язык программирования – писать на нём программы.

Брайэн Керниган

Список используемой литературы

1. Безменов Н. Turbo Pascal 7.0 руководство программиста. – М.: Эксмо, 2006. – 160 с.
2. Дональд Кнут Искусство программирования. Основные алгоритмы — М.: Вильямс, 2006. — с. 720.
3. Моргун А. Н. Справочник по Turbo Pascal для студентов. - М.: Диалектик», 2006. — с. 608.
4. Нэйл Рубенкинг Turbo Pascal для Windows. — М.: Мир, 1994. — с. 535.
5. Порублев И.Н., Ставровский А.Б. Алгоритмы и программы. Решение олимпиадных задач. — М.: Вильямс, 2007. — с. 480.
6. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ. — М.:Вильямс, 2006. — с. 1296.
7. Ушакова Д.М., Юркова Т.А. Паскаль для школьников. – М.: Питер, 2008. – 256 с.
8. Эллиот Б. Коффман Turbo Pascal Web Update. — М.: Вильямс, 2005. — с. 896.
9. http://pascalstudy.narod.ru/
10. http://ru.wikipedia.org/wiki/Turbo_Pascal
11. http://lessons-tva.info/edu/e-inf1/e-inf1-4-2.html
12. http://bukvezkaya.narod.ru/
13. http://www.miit.ru/prt/mmz/cd/paskal.htm
14. http://www.nesterova.ru/bibl/algorithm_lang/