Зарегистрировано №_____
«___»_____2019 г.

_____ _____
подпись (расшифровка подписи)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ
Кафедра прикладной информатики и информационных технологий

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МАШИНЫ ПОСТА

Курсовой проект
по дисциплине «Программирование»

студента очной формы обучения
направления подготовки 38.03.05 «Бизнес-информатика»

1 курса группы 12001131
Клименко Надежды Анатольевны

Допущена к защите
«___»_____2019 г.

_____ _____
Подпись (расшифровка подписи)

Оценка_____
«___»_____2019 г.

_____ _____
Подпись (расшифровка подписи)

Научный руководитель:
к.т.н., доцент

Асадуллаев Р.Г.

БЕЛГОРОД 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Язык Паскаль и его современные версии остаются одними из самых популярных языков программирования в мире Этому способствуют, с одной стороны, простота лежащего в ее основе языка программирования Паскаль, а с другой постоянные модификации, ведущие к его улучшению и совершенствованию. Придуманный швейцарским ученым Никласом Виртом как средство для обучения студентов программированию, язык Паскаль стараниями превратился в мощную современную профессиональную систему программирования, которой по плечу любые задачи - от создания простых программ, предназначенных для решения несложных вычислительных задач, до разработки сложнейших реляционных систем управления базами данных [2].

Абстрактная вычислительная машина - теоретическое построение, с помощью которого вводится строгое, математическое определение алгоритма. Существуют различные виды абстрактных машин, рассмотрим некоторые из них.

Автомат – разновидность абстрактной вычислительной машины, которая определяется:

- множеством входных и выходных сигналов;
- множеством состояний;
- функцией, задающей переходы из одних состояний в другие;
- функцией, определяющей выходные сигналы в зависимости от входного сигнала и текущего состояния.

Автомат предназначен для формальной переработки последовательностей символов.

Конечный автомат - математическая модель устройства с конечной памятью. Конечный автомат перерабатывает множество входных дискретных сигналов в множество выходных сигналов. Различают синхронные и асинхронные конечные автоматы.

Машина Поста - математическое построение, предназначенное для уточнения понятия алгоритма. Машина Поста состоит:

– из неограниченной в обе стороны ленты, разделенной на ячейки;

– из головки чтения/записи, которая может перемещаться вдоль ленты и управляется программой на специальном языке из шести команд.

Машина Тьюринга - математическое построение, предназначенное для уточнения понятия алгоритма. Машина Тьюринга состоит:

– из неограниченной в обе стороны ленты, разделенной на ячейки;

– из головки чтения/записи, которая может перемещаться вдоль ленты.

Программа для машины Тьюринга, задается в виде таблицы, определяющей команды для головки.

Нормальный алгоритм Маркова - математическое построение, предназначенное для уточнения понятия алгоритм. Нормальный алгоритм Маркова:

– задается алфавитом и нормальной схемой подстановок, выполняемых по заранее определенной схеме;

– определяет преобразование строк.

Доказано, что совпадает класс нормальных алгоритмов Маркова и класс алгоритмов, представленных в форме машины Тьюринга, [5].

Целью данного курсового проекта является разработка и реализация программы, которая бы моделировала работу простейшей машины Поста, а также визуализировала работу разработанной модели.

Для достижения цели необходимо решить следующие задачи:

– изучить и описать принцип работы машины Поста;

– разработать алгоритма программной реализации;

– произвести программную реализацию модели машины Поста;

– произвести тестирование разработанной программы и проанализировать результаты её работы;

1 Описание работы машины поста

Абстрактная машина, которая в дальнейшем стала именоваться, как машина Поста, была предложена американским математиком Эмилем Постом в 1936 г. Абстрактная (т.е. существующая не реально, а лишь в воображении) машина Поста, предназначена для доказательств различных утверждений о свойствах программ. Данная машина представляет собой универсальный исполнитель, являющийся полностью детерминированным, позволяющим «вводить» начальные данные, и после выполнения программ «читать» результат. Машина Поста менее популярна машины Тьюринга, хотя она значительно проще машины Тьюринга. С ее помощью можно вести обучение первым навыкам составления программ для ЭВМ [1].

Машина Поста представляет собой бесконечную ленту, разделенную на одинаковые клетки, каждая из которых может быть либо пустой, либо заполненной меткой «V», и головки, которая может перемещаться вдоль ленты на одну клетку вправо или влево, наносить в клетку ленты метку, если этой метки там ранее не было, стирать метку, если она была, или проверять наличие в клетке метки. Информация о заполненных метками клетках ленты характеризует состояние ленты, которое может меняться в процессе работы машины. В каждый момент времени головка («-») находится над одной из клеток ленты. Информация о местоположения головки вместе с состоянием ленты характеризует состояние машины Поста (рисунок 1.1).

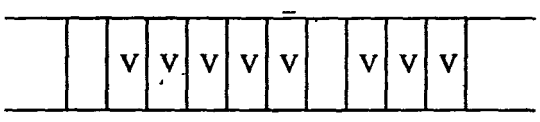


Рисунок 1.1 - Абстрактная машина Поста

Команда машины Поста имеет следующую структуру: *n Km,* где *n* - порядковый номер команды, *K*-действие, выполняемое головкой, *m* - номер следующей команды, подлежащей выполнению.

Существует всего шесть команд машины Поста, они представлены на рисунке 1.2.

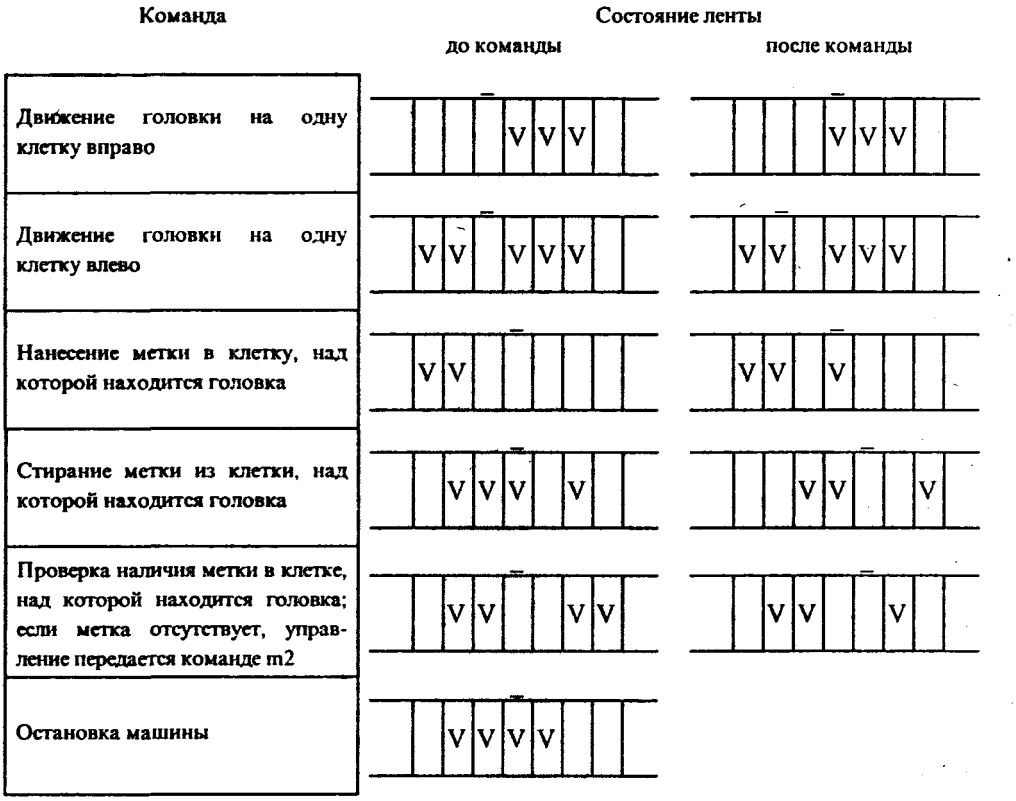


Рисунок 1.2 - Команды машины Поста

Ситуации, в которых головка должна наносить метку там, где она уже имеется, или, наоборот, стирать метку там, где ее нет, являются аварийными (недопустимыми) [3].

Программой для машины Поста называется непустой список команд, такой что 1) на *n*-м месте команда с номером *n*; 2) номер *m* каждой команды совпадает с номером какой-либо команды списка.

С точки зрения свойств алгоритмов, изучаемых с помощью машины Поста, наибольший интерес представляют причины останова машины при выполнении программы:

1) останов по команде «стоп»; такой останов называется результативным и указывает на корректность алгоритма (программы);

2) останов при выполнении недопустимой команды; в этом случае останов называется безрезультативным;

3) машина не останавливается никогда; в этом и в предыдущем случае мы имеем дело с некорректным алгоритмом (программой) [3].

Число k представляется на ленте машины Поста идущими подряд $k + 1$ метками (одна метка означает число «О»). Между двумя числами делается интервал как минимум из одной пустой секции на ленте.

Рассмотрим типичный элемент программы машины Поста.

Пусть задано исходное состояние головки и требуется на пустой ленте написать две метки: одну в секцию под головкой, вторую справа от нее. Это можно сделать по программе, представленной на рисунке 1.3 (справа от команды показан результат ее выполнения) [1]:

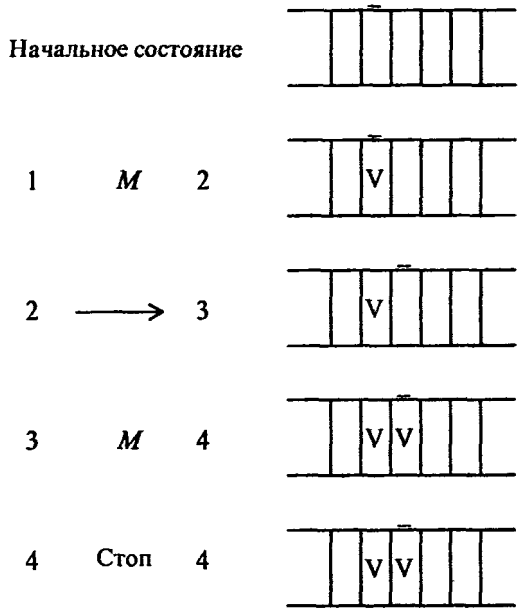


Рисунок 1.3 - Пример элемента программы машины Поста

2 Разработка алгоритма решения задачи

Разработанная в результате выполнения курсового проекта программа должна будет представлять собой модель машины Поста на ЭВМ. Программа будет разрабатываться на языке Турбо Паскаль и будет реализовывать шесть команд машины Поста. Начальное состояние ленты, место головки и последовательность выполнения команд будут считываться программой из текстового файла. Первая строка текстового файла содержит последовательность нулей и единиц, и задает тем самым состояние ленты, вторая строка содержит число, указывающее номер позиции головки на ленте, а начиная с третьей строки, через пробел, указываются номера команд (на каждой строке одна команда).

Для команды, реализующей условный переход необходимо записать в текстовый файл еще два значения (номер строки следующей команды, если условие выполнится, и номер строки в противном случае). Следовательно, в текстовом файле, на строке, содержащей номер команды условного оператора будет располагаться три цифры: первая содержит информацию о номере команды, вторая указывает номер строки, на которую необходимо перейти если условие истинно, и третья указывает номер строки перехода если условие ложно.

Так как текстовый файл считывается сверху вниз и строка не может считаться дважды, в программе будет реализован двумерный массив размера *Nx3*, в который переписывается текст программы машины Поста из файла. Дальнейшую обработку записанной программы Паскаль реализует, считывая ее из массива.

Лента в программе будет описана строковым типом, а номер позиции головки храниться в переменной целочисленного типа. Выполнение недопустимой команды (запись в ячейку с меткой и стирание пустой ячейки) приведет к аварийному выходу из программы. Для наглядности хода выполнения программы реализованы задержки.

Для проверки работоспособности программы выполняется алгоритм сложения двух целых неотрицательных чисел *A* и *B* на машине Поста, когда головка находится над числом *A,* а число *B* находится правее числа *A* на некоторое число клеток (рисунок 2.1). Программа реализует следующий алгоритм: первое число постепенно придвигается ко второму до их слияния, а потом стирается одна метка (иначе результат оказался бы на единицу больше верного) [4].

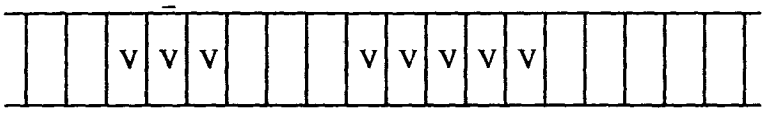


Рисунок 2.1 - Пример записи двух чисел 2 и 4 на ленте машины Поста

Алгоритм данной программы выглядит следующим образом:

1) переход головки влево;
2) если в ячейке «1» то переход на 1) иначе на 3);
3) переход головки вправо;
4) запись «0»;
5) переход головки вправо;
6) если в ячейке «1» то переход на 5) иначе на 7);
7) запись «1»;
8) переход головки вправо;
9) если в ячейке «1» то переход на 10) иначе на 1);
10) переход головки влево;
11) если в ячейке «1» то переход на 10) иначе на 12);
12) запись «0»;
13) остановка программы.

Результатом работы данной программы является сумма двух чисел.

В программе будет принято следующие обозначения команд машины Поста, используемых при составлении алгоритма и записи его в файл и при использовании оператора «условного выбора»:

1 – переход вправо;

2 – переход влево;

3 – запись единицы;

4 – запись нуля;

5 – переход по условию;

6 – остановка выполнения программы.

3 Разработка программы

Информация о последовательности выполнения команд машины Поста хранится в двумерном массиве Matrix типа array [1..N, 1..M] of Integer размера M*N, где константы N=100 и M=3. В первом столбце массива хранится информация о номере выполняемой команды, второй и третий столбцы содержат информацию о том, на какую ячейку массива необходимо перейти при выполнении условия условного оператора и в противном случае, следовательно, они заполняются только при наличии в первом столбце числа 5.

Значение ленты Поста хранится в переменной Lenta типа String. Переменная Pos служит для хранения номера позиции головки на ленте, I – хранит номер строки массива при записи из файла, L – используется в качестве индекса при работе с лентой, X и Y – служат в качестве координат при выводе на экран строки и курсора, имитирующего головку. Файловая переменная TF типа Text, используется при чтении информации из файла. Переменная Flag типа Boolean используется в команде останова программы. Значение переменных Driver, Mode типа Integer используются при настройке графического режима.

Так же подключены модули Crt, Graph. Crt необходим для использования процедуры GotoXY, ClrScr, Delay. Graph нужен для работы в графическом режиме и использования процедур вырисовывания прямоугольников, задания цвета линий.

В программе написаны две процедуры Jump и ReadFile.

Процедура Jump, описывает алгоритм работы команды условного перехода машины Поста (рисунок 3.1). Входные переменным I1, L1, Lenta1 передается значение из основной программы, а после завершения работы процедура возвращает глобальным переменным измененное значение. Локальным переменным A, B присваивается из массива значение номеров строк, на которые необходимо перейти при выполнении условия *A* и B противном случае *B*. Перед завершением работы, процедура Jump уменьшает значение I1. Это связано с тем, что в главной программе организован цикл считывания

номера команды из массива и увеличение значения I на «1», а процедура выдает значение ячейки, с которой необходимо считать номер. Следовательно, процедура уменьшает номер, а программа увеличивает, тем самым происходит компенсация их работы. Работа процедуры заключается в следующих случаях: если текущее значение ленты равно «1» то в переменную I записывается номер строки массива *A,* в противном случае номер строки массива *B*.

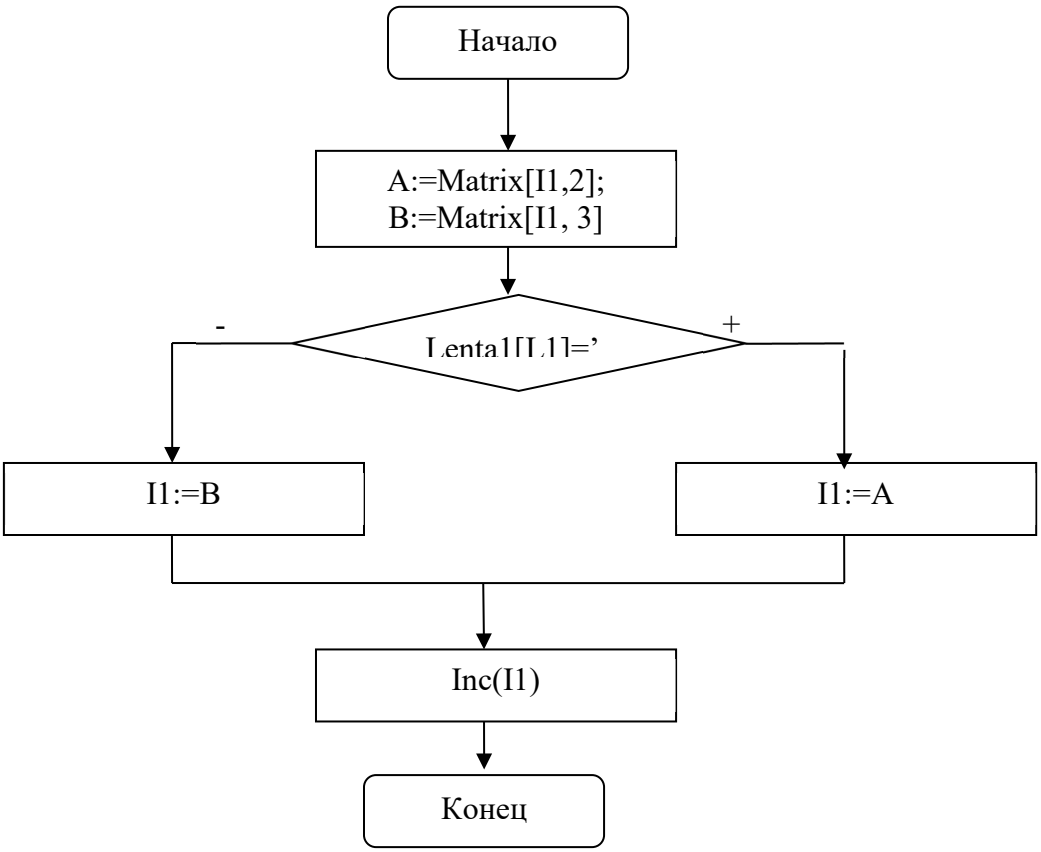


Рисунок 3.1 – Блок-схема алгоритма процедуры Jump

Процедура ReadFile (рисунок 3.2) предназначена для чтения файла и записи его строк в массив и переменные. В данной подпрограмме используются только глобальные переменные.

После того как подпрограмма связала файловую переменную TF с файлом «Prog.txt» (файл должен находится там же, где и файл программы, в противном случае придется указывать путь к текстовому файлу) и открыла его для чтения Reset(TF) она записывает первую строку файла в переменную Lenta,

вторую в переменную Pos, а затем организует цикл считывания до конца файла while not EOF(TF) do. Внутри данного цикла происходит считывание всех первых цифр строк в первый столбец массива Matrix и проверяется условие если значение первого столбца массива равно «5», то считать остальные две цифры строки файла в противном случае перейти на следующую строку файла. Перед завершением работы процедура закрывает файл и передает управление в главную программу.

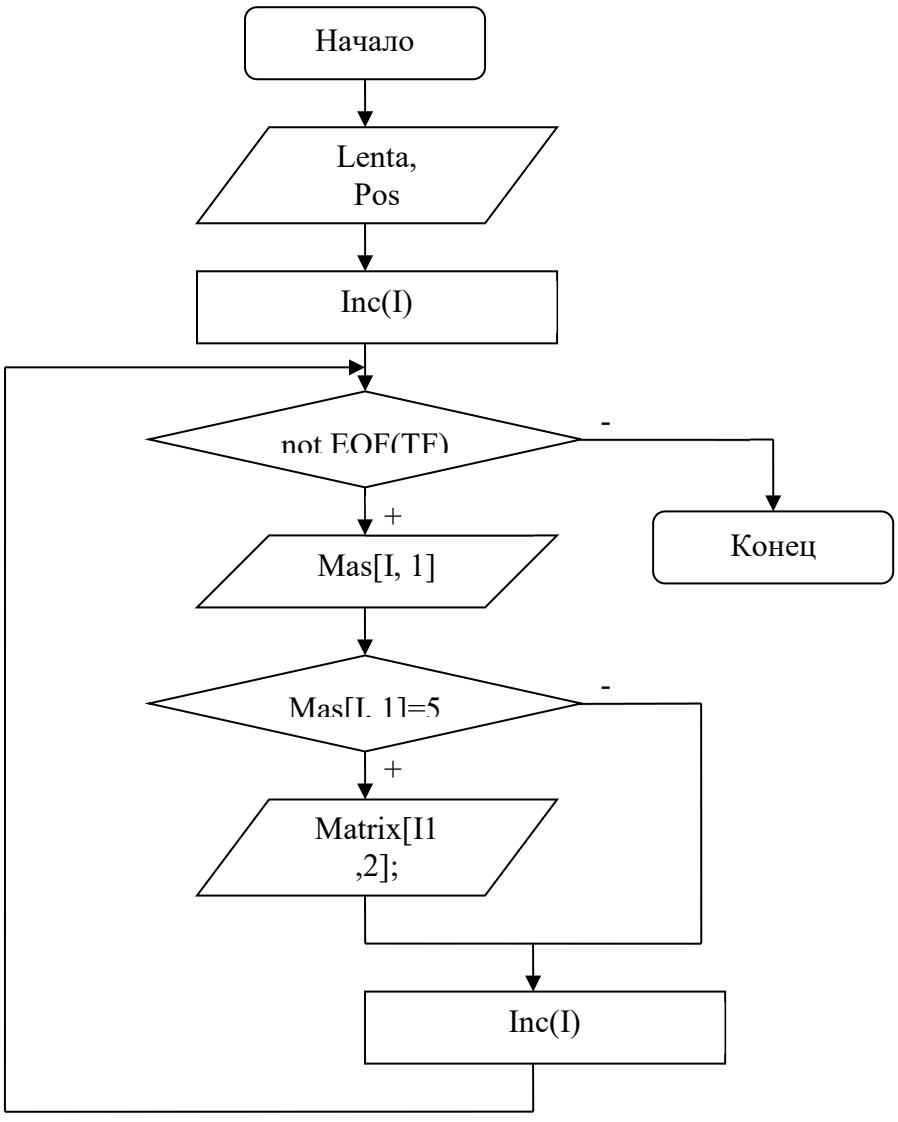
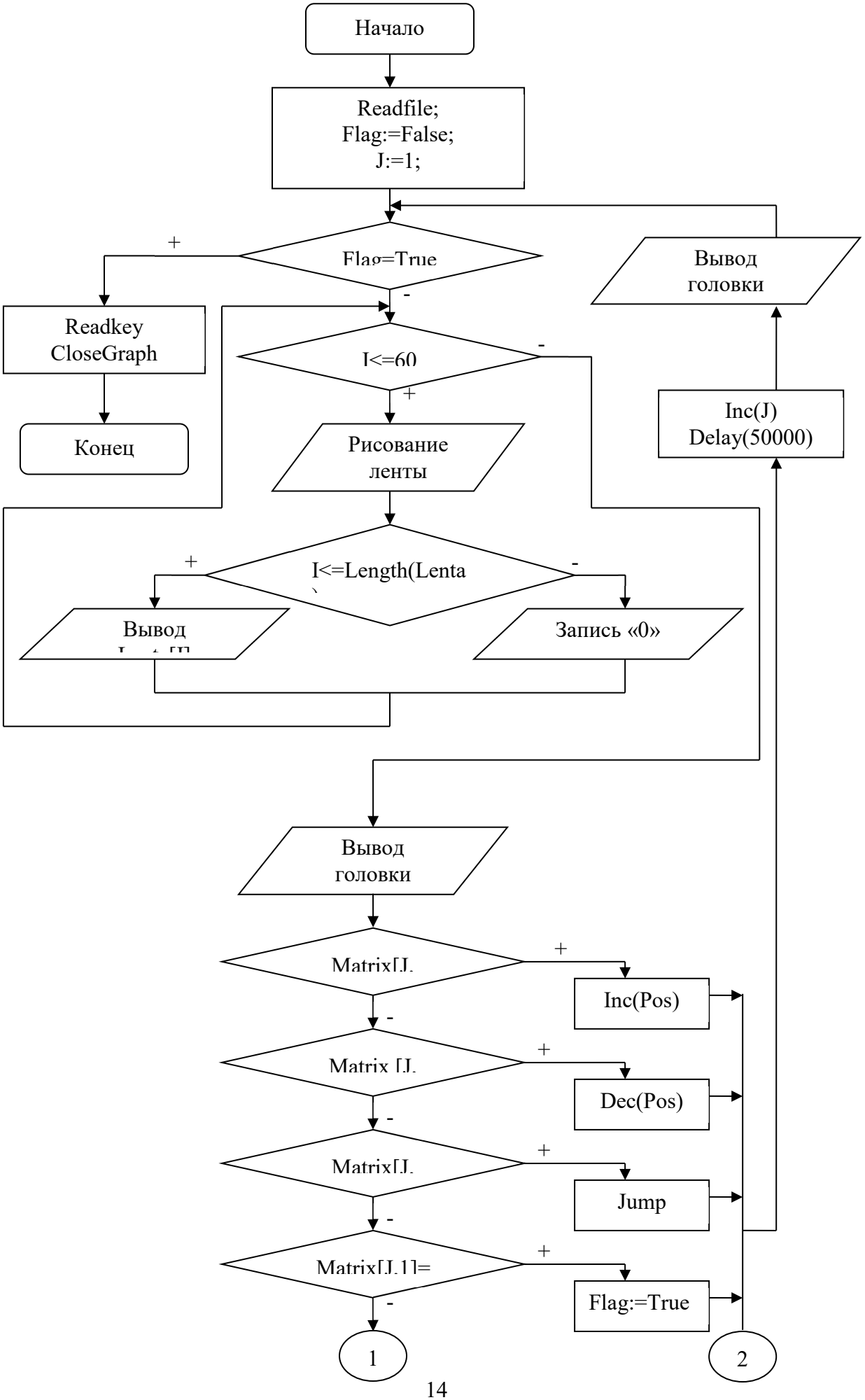


Рисунок 3.2 - Блок-схема алгоритма процедуры ReadFile


Начало
Readfile;
Flag:=False;
J:=1;
Flag=True
+
-
Readkey
CloseGraph
Конец
I<=60
-
+
Рисование
ленты
I<=Length(Lenta
+
-
Вывод
Запись «0»
Вывод
головки
Inc(J)
Delay(50000)
Вывод
головки
Matrix[J,
+
Inc(Pos)
-
Matrix [J,
+
Dec(Pos)
-
Matrix[J,
+
Jump
-
Matrix[J,1]=
+
Flag:=True
-
1
2

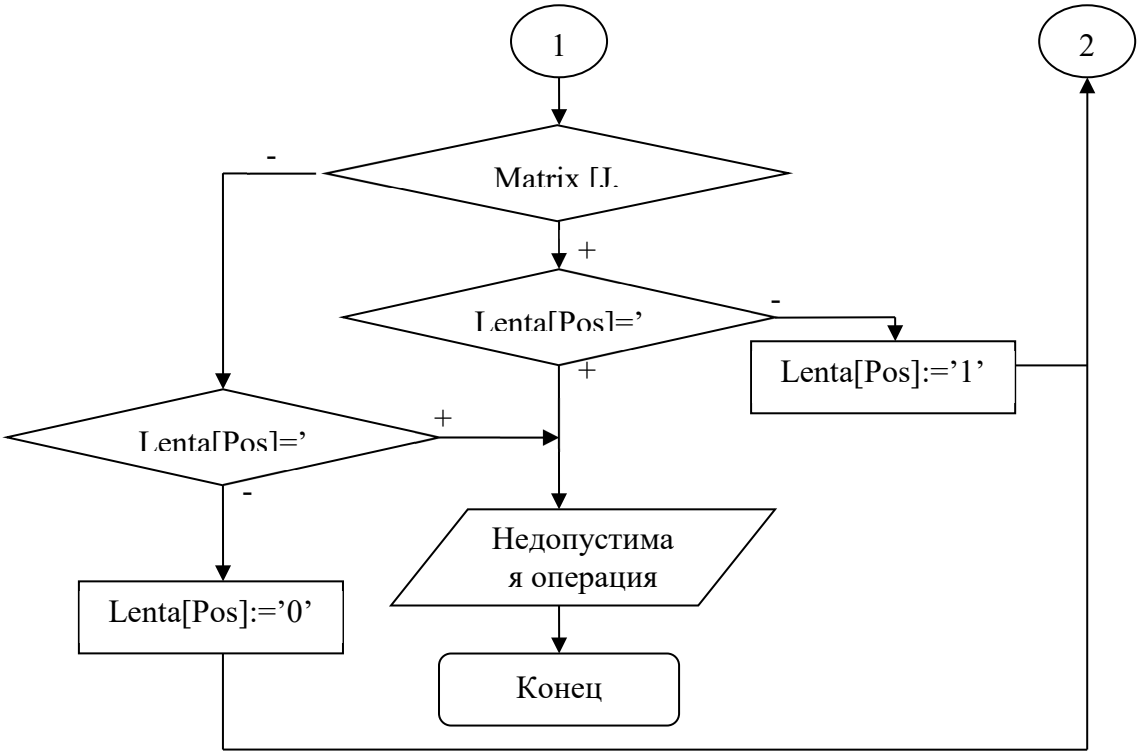


Рисунок 3.3 – Блок-схема главной программы

Главная программа, осуществляющая процесс обработки алгоритма работы машины Поста, представлена на рисунке 3.3.

Первой запускается процедура ReadFile, которая считывает информацию с файла, затем переменная Flag устанавливается на значение ложь, так как значение истина является признаком останова программы. Затем инициализируется графический режим работы Driver:=Detect – этим мы указываем автоматическое определение драйвера, InitGraph(Driver, Mode, '').

Основной частью главной программы является цикл repeat until, который выполняется до тех пор, пока не выполнится условие истинности Flag. Первым шагом цикла является очистка экрана в графическом режиме ClearBevice и задание белого цвета SetColor(White) вырисовываемых линий.

Цикл for I:=1 to 60 do вырисовывает прямоугольники Rectangle((I-1)*10,190, I*10, 210), изображая ленту, и заносит в них значения OutTextXY((I-1)*10+2,200, Lenta[I]). Когда длина переменной Lenta исчерпана, остальная часть нарисованной ленты заполняется нулями OutTextXY((I-1)*10+2, 200, '0').

Затем вырисовывается прямоугольник имитирующий головку машины поста Rectangle((Poz-1)*10, 213, (Poz*10), 214), и устанавливается красный цвет

вывода сообщений о том, какая команда машины Поста выполняется в данный момент SetColor(Red).

Затем идет оператор выбора, который считывает очередное значение массива и в соответствии выполняет одну из 6 команд машины Поста:

1: сдвиг головки вправо, то есть в переменной лента смещается указатель вправо Inc(Pos) и вывод сообщения OutTextXY(170, 170, 'Сдвиг вправо');

2: сдвиг головки влево, аналогично предыдущей команде, только смещение влево Dec(Pos) и вывод сообщения OutTextXY(170, 170, 'Сдвиг влево');

3: Запись единицы, переменной, описывающей ленту, в ячейку присваивается единица Lenta[Pos]:='1' и выводится сообщение OutTextXY(170, 170, 'Запись «1»'). Здесь так же осуществляется проверка условия, что бы ячейка изначально не содержала «1», иначе выводится сообщение о некорректности алгоритма и выполнение работы программы завершается;

4: Запись нуля, переменной, описывающей ленту, в ячейку присваивается нуль Lenta[Pos]:='0' и выводится сообщение OutTextXY(170, 170, 'Запись «0»'). Здесь так же осуществляется проверка на то, что бы ячейка изначально не содержала 0, иначе выводится сообщение о некорректности алгоритма с последующим завершением работы программы;

5: переход по условию, вызывается выше описанная процедура Jump(J, Pos, Lenta), которая и осуществляет переход, после чего выводится сообщение OutTextXY(170, 170, 'Переход');

6: остановка выполнения программы, признаком остановки программы является Flag:=True, после чего цикл заканчивает свою работу и выводит сообщение OutTextXY(170, 170, 'Стоп').

После выполнения команды производится задержка Delay(50000), величина которой выбирается в зависимости от производительности ЭВМ. Устанавливается черный цвет SetColor(Black) для того чтобы зарисовать старое значение головки Rectangle((Poz-1)*10, 213, (Poz*10), 214) и увеличивается значение индекса массива для считывания следующей команды Inc(J).

В конце программы стоит ReadKey, ожидающий нажатия клавиши который позволяет пользователю просмотреть результаты работы программы и закрытие графического режима CloseGraph.

4 Описание интерфейса пользователя и тестирование программы

Записанная программа в блокноте выглядит следующим образом (рисунок 4.1)

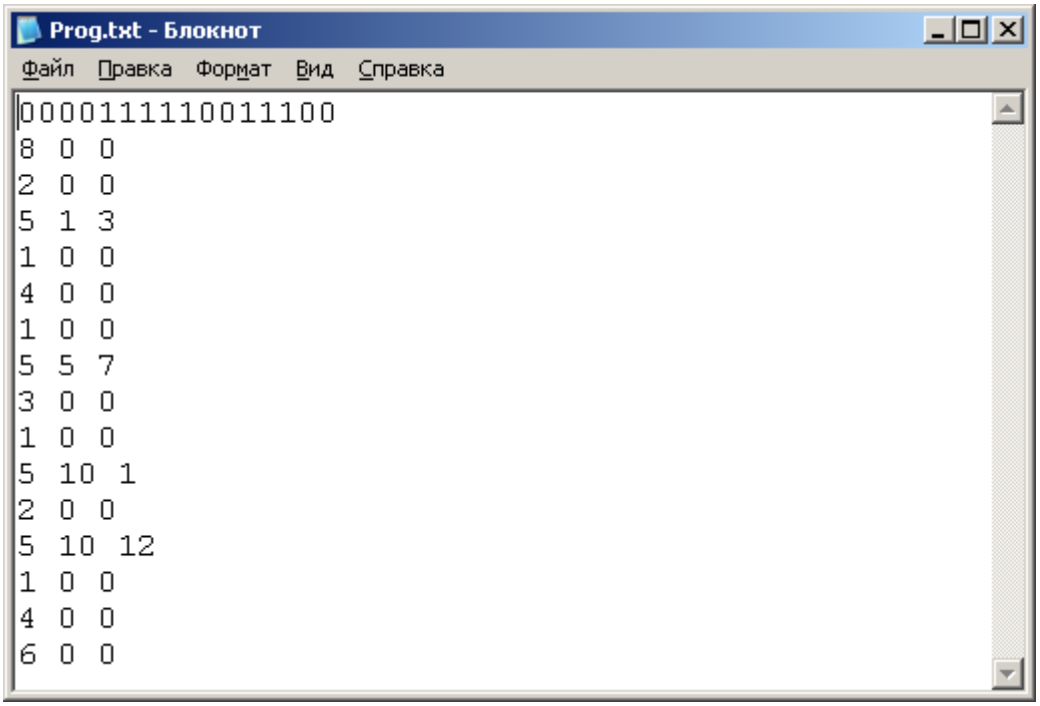


Рисунок 4.1 – Вид файла с программой для машины Поста в режиме блокнот

На рисунке 4.2 изображен вид программы после запуска. Курсор позиционируют по ленте с задержками, указывая на то, в какой именно ячейке происходит действие. Как видно из рисунка, начальными данными являются два числа («4» и «2»)

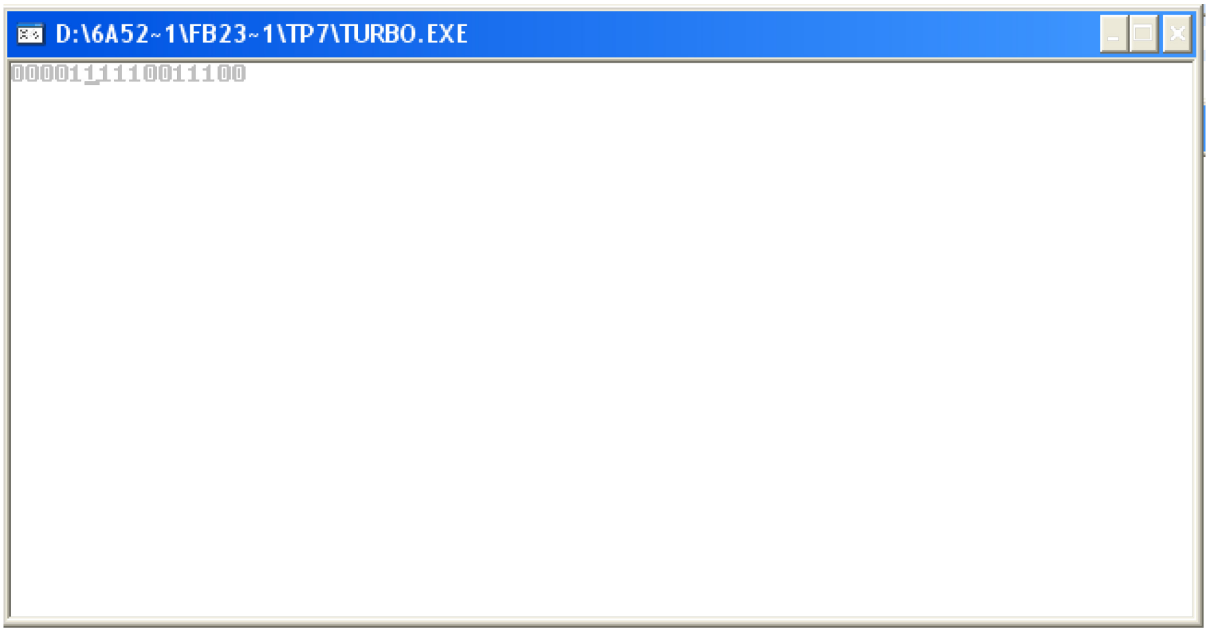


Рисунок 4.2 - Начало работы программы

Рисунок 4.3 показывает результат работы программы. Как видно результат сложения равен 6.

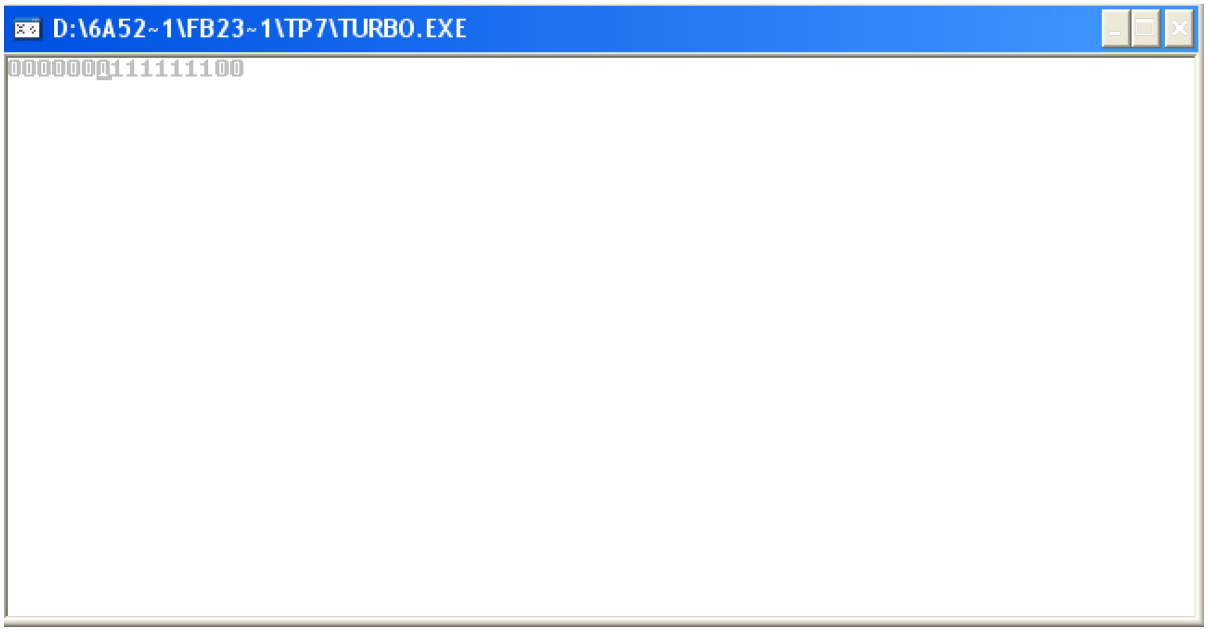


Рисунок 4.3 - Результат работы программы

Проверим правильность работы программы при внесении ошибки в алгоритм Поста. Занесем в пятую строку программы вместо команды стереть команду записать (рисунок 4.4)

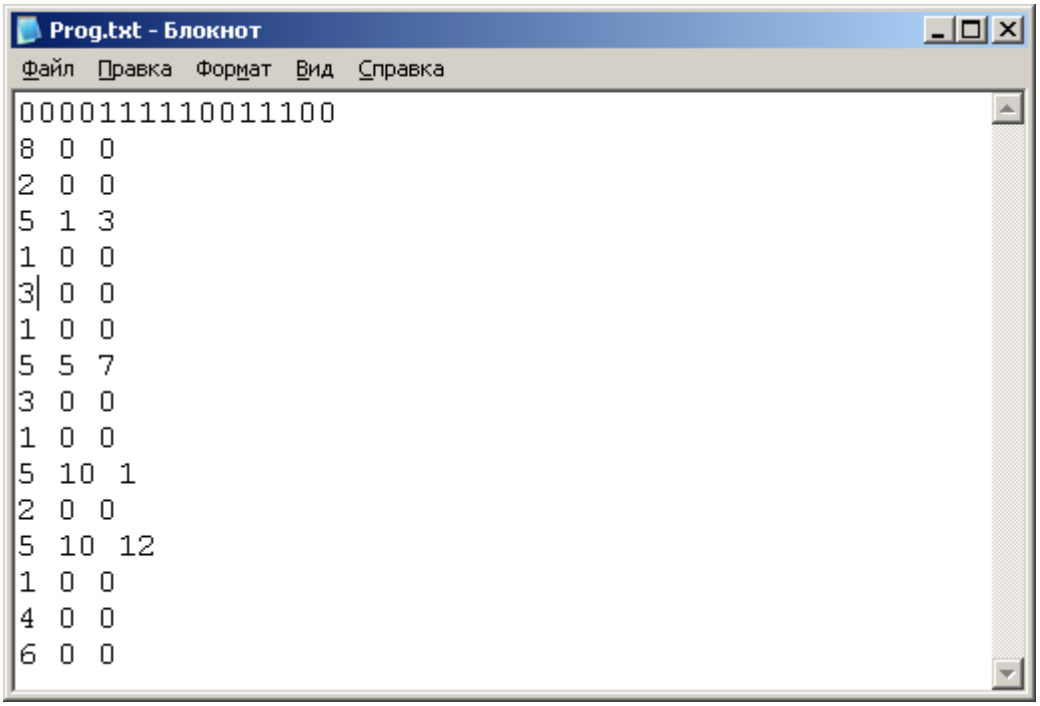


Рисунок 4.4 - Документ программы Поста с ошибкой в 6 строке

В результате выполнение программы прервется и на экран высветится сообщение об ошибке (рисунок 4.5)

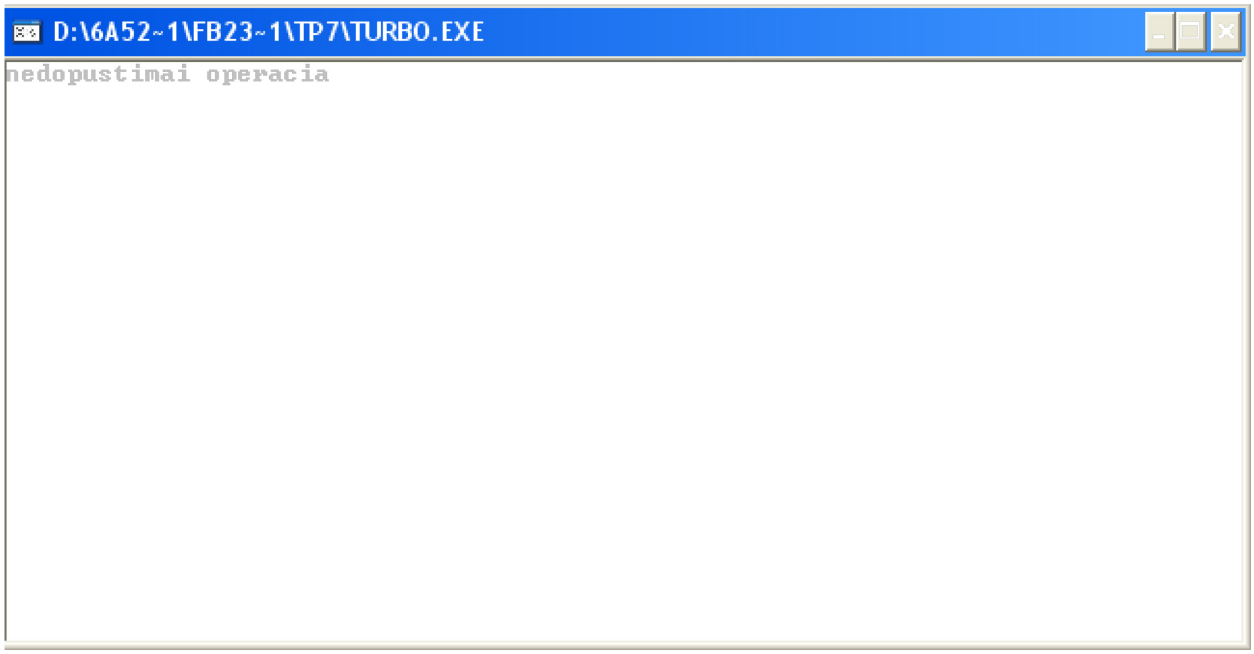


Рисунок 4.5 - Ошибка выполнения программы

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта стала программная реализация машины Поста при помощи средств языка программирования Турбо Паскаль.

Разработана программа, имитирующая работу машины Поста, а именно ленту бесконечной длины, головку машины Поста и шесть команд, доступных при работе данной машины. Построены блок схемы главной программы и подпрограмм Jump и ReadKey, на основе которых можно реализовать алгоритм работы машины Поста на других языках программирования.

В конце работы было проведено тестирование программы на основе примера сложения двух чисел, которое показало корректность написанной программы.

В качестве недостатков программы можно отметить ограниченность размеров входной программы, а также режим работы программы, который является «режимом реального времени» и забирает значительные ресурсы ЭВМ. Устранить вышеописанные недостатки можно переписав программу на более высоком языке программирования, например, Delphi.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Могилев, А.В. Информатика / А.В. Могилев, Н.И. Пак, Е.К. Хеннер. – М.: Академия, 2016. – 848 с.

2) Фаронов, В.В. Турбо Паскаль 7.0 / В.В. Фаронов – М.: «ОМД Групп», 2013. – 616 с.

3) URL: http://it.kgsu.ru/TI_5/falg_003.html

4) URL: http://inf.1september.ru/articlef.php?ID=200700104

5) URL: http://www.glossary.ru/cgi -bin/gl_sch2.cgi?RAhxywgqyt: l!i:,oxroylr;t:l!sg@ot:

ПРИЛОЖЕНИЕ

Текст программы на языке Turbo Pascal

```
program Post_Machine;

uses Graph, Crt;

const
 M = 3;
 N = 100;

var
 Lenta: String;
 Driver, Mode, Pos, I, L, J, X, Y: Integer;
 Matrix: array[1..N, 1..M] of Integer;
 TF: Text;
 Flag: Boolean;

procedure Jump(var I1, L1: Integer; var Lenta1: String);
var
 A, B: Integer;
begin
  A:=Matrix[I1, 2];
  B:=Matrix[I1, 3];
  if Lenta1[L1]='1' then I1:=A else I1:=B;
  Dec(I1);
end;

procedure ReadFile;
begin
 Assign(TF, 'Prog.txt');
 Reset(TF);
 Readln(TF, Lenta);
 Readln(TF, Pos);
 I:=1;
 while not EOF(TF) do
  begin
    Read(TF, Matrix[I,1]);
    if Matrix[I,1]=5 then
      begin
       Read(TF, Matrix[I,2]);
       Readln(TF, Matrix[I,3]);
      end
     else ReadLn(TF);
    Inc(I);
   end;
 Close(TF);
```

```
end;

Begin
 ReadFile;
 Flag:=False;
 J:=1;
 Driver:=Detect;
 InitGraph(Driver, Mode, '');
 Repeat
  ClearDevice;
  SetColor(White);
  for I:=1 to 60 do
   begin
    Rectangle((I-1)*10,190, I*10, 210);
    if I<=Length(Lenta) then OutTextXY((I-1)*10+2,200, Lenta[I])
     else OutTextXY((I-1)*10+2, 200, '0');
   end;
  Rectangle((Pos-1)*10, 213, (Pos*10), 214);
  SetColor(Red);
  case Matrix[J,1] of
   1:
    begin
     Inc(Pos);
     OutTextXY(170, 170, 'SDVIG VPRAVO');
    end;
   2:
    begin
     Dec(Pos);
     OutTextXY(170, 170, 'SDVIG VLEVO');
    end;
   3:
    if Lenta[Pos]='1' then
      begin
       OutText('NAIDENA NEDOPUSTIMAI OPERACIA');
       ReadKey;
       Halt;
      end
     else
      begin
       Lenta[Pos]:='1';
       OutTextXY(170, 170, 'ZAPIS 1')
      end;
   4:
    if Lenta[Pos]='0' then
      begin
       OutText('NAIDENA NEDOPUSTIMAI OPERACIA');
       ReadKey;
       Halt;
      end
     else
      begin
       Lenta[Pos]:='0';
       OutTextXY(170, 170, 'ZAPIS 0');
```

```
      end;
     5:
      begin
       jump(J, Pos, Lenta);
       OutTextXY(170, 170, 'PEREHOD');
      end;
     6:
      begin
       Flag:=True;
       OutTextXY(170, 170, 'STOP');
      end;
     end;
    Delay(65535);
    SetColor(Black);
    Rectangle((Pos-1)*10, 213, (Pos*10), 214);
    Inc(J);
   until Flag;
 Readkey;
 CloseGraph;
End.
```