Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по курсовой работе**

**Дисциплина**: Прикладное программирование

**Тема**: Симулятор машины Тьюринга с односторонней лентой

Выполнил студент гр. 23531/3 Е.С. Шумеева

(подпись)

Преподаватель М.А. Беляев

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Санкт-Петербургский государственный политехнический университет

**ЗАДАНИЕ**

**НА ВЫПолнение курсовой работы**

студенту группы 23531/3 Шумеевой Екатерине Сергеевне

*(номер группы) (фамилия, имя, отчество)*

***1. Тема работы:*** Симулятор машины Тьюринга с односторонней лентой

***2. Срок сдачи законченного проекта (работы)*** 2 июля 2018 г.

***3. Исходные данные к проекту (работе)***: *описание необходимого функционала программы, описание метода получения исходных данных и представления результата программы.*

***4. Содержание пояснительной записки*** (перечень подлежащих разработке вопросов): введение, основная часть (*метод решения, коды ошибок программы, методика испытаний, моделирование ошибок*), заключение, список использованных источников, приложения.

***Дата получения задания***: «21» марта 2018 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Беляев

*(подпись) (инициалы, фамилия)*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.С. Шумеева

*(подпись студента) (инициалы, фамилия)*

\_\_\_\_\_\_\_\_\_\_\_\_

*(дата)*

**Содержание**

[Введение 4](#_Toc517972953)

[1. Метод решения 5](#_Toc517972954)

[2. Коды ошибок 5](#_Toc517972955)

[3. Методика испытаний. Моделирование ошибок 6](#_Toc517972956)

[Заключение 11](#_Toc517972957)

[Список использованных источников 11](#_Toc517972958)

[Приложение 1 12](#_Toc517972959)

[Приложение 2 13](#_Toc517972960)

[Приложение 3 16](#_Toc517972961)

[Приложение 4 16](#_Toc517972962)

[Приложение 5 17](#_Toc517972963)

[Приложение 6 18](#_Toc517972964)

## Введение

В данном курсовом проекте будет реализован симулятор односторонней машины Тьюринга. Программный язык реализации - С.

Машина Тьюринга с односторонней лентой характеризуется тем, что головка машины никогда не сдвигается левее начальной ячейки (т.е. всегда находится в ячейках с положительными номерами). В случае попытки выхода за левую границу ленты будет выдана ошибка.

Входные данные считываются из двух входных файлов.

Первый файл содержит таблицу переходов. Команды переходов должны быть заданы в следующем виде: Aqn-Bqk|M, где:

A и B – символы алфавита, хранящиеся в ячейке и записываемые в нее, соответственно.

qn и qk – состояния машины, в котором она находится в данный момент и в которое необходимо перейти, соответственно. В начальный момент времени машина находится в состоянии 1. Для остановки машины зарезервировано состояние 0.

M – направление перемещения головки. Может принимать одно из следующих значений:

R - сдвиг головки вправо;

L - сдвиг головки влево;

H - головка остается на месте.

Второй файл должен содержать начальное состояние ленты и положение головки в виде:

\_\_\_v\_\_\_\_\_\_\_\_\_\_\_\_

. . . 0111 . . . . . . . . .

Если ввести более одного положения головки, то будет выведена ошибка о некорректном задании входной ленты.

Будет реализовано 2 режима работы программы: стандартный режим и режим отладки. Флаг «-а» будет указывать, что необходимо запустить программу в стандартном режиме. Флаг «-о» - в режиме отладки. Для стандартного режима можно дополнительно указать количество шагов. В режиме отладки будут реализованы следующие возможности: выполнить один шаг (s), прервать выполнение программы (b), продолжить выполнение программы в обычном режиме (g).

Полученный результат будет записываться в файл вывода и отображаться в командной строке. В выходном файле содержится начальное состояние ленты и все промежуточные состояния и команды, выполненные в процессе работы.

Вид командной строки:

Turing.exe inputTable.txt inputTape.txt outFile.txt -a [-p]/ -o

В процессе создания продукта будут предусмотрены возможные проблемы взаимодействия пользователя с программой. Все эти случаи будут генерировать определённые ошибки.

# Метод решения

Программа состоит из 6 файлов. В одном из них – «help\_function.c» – содержатся вспомогательные функции общего предназначения. Например, проверка, что в строке содержится число.

Универсальный заголовочный файл («header») – input.h – используется во всех оставшихся пяти файлах.

Исполнение программы начинается с файла «main.c». В него передаются аргументы из командной строки и, если они соответствуют заданному формату, то происходит запуск файла «input.c».

Далее в файле «input.c» происходит разбор и проверка входных данных ленты и таблицы состояний на соответствие заданному формату. Если данные заданы неверно, то программа выдает соответствующую ошибку с пояснением возникшей проблемы.

Следующим этапом выполняется запуск файла «step.c». В данном файле происходит пошаговое выполнение команд. Также в данном файле описывается взаимодействие пользователя с программой в режиме отладки. При достижении определенного результата программа выполняет передачу соответствующих данных в файл «output.c».

В файле «output.c» содержатся функции для вывода результатов в командную строку, а также в указанный файл вывода.

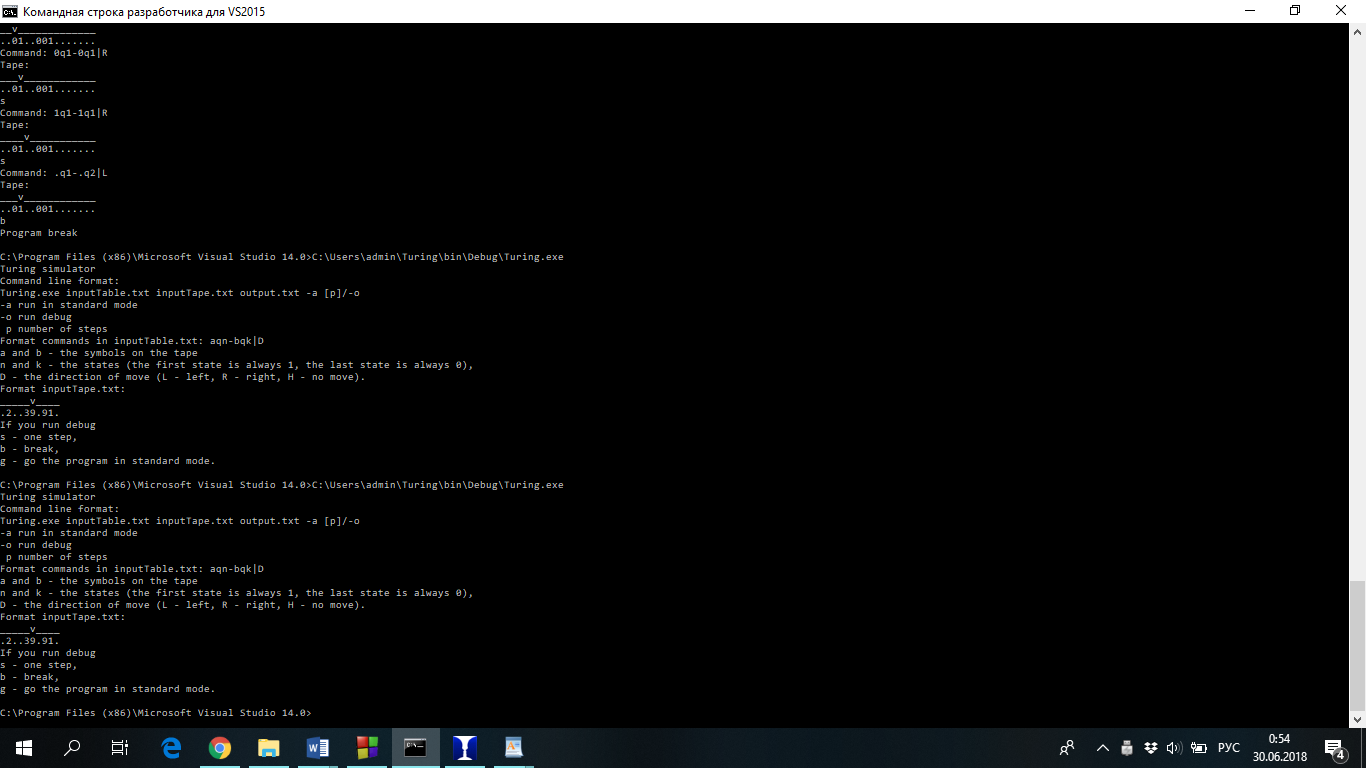
В программе используются только стандартные подключаемые библиотеки и методы. Все действие основано на обработке строк и числовых массивов.

# Коды ошибок

|  |  |  |
| --- | --- | --- |
| **Код ошибки** | **Сообщение** | **Комментарий** |
| 10 | Can't find file | Введенные входные файлы отсутствуют |
| 11 | Wrong number of arguments | Неверный формат данных в командной строке. Необходимо повторить ввод в соответствии с указанным стандартом |
| 12 | Wrong number of input chars | Команда в таблице переходов не соответствует указанному формату. Необходимо повторить ввод в соответствии с указанным стандартом |
| 13 | Incorrect command in line - "" instead of the expected "" | Неверный символ в команде в таблице переходов |
| 14 | Wrong move command | Неверно задано направление перемещения головки |
| 15 | The state can not be negative | Состояния перехода могут иметь только целые неотрицательные значения |
| 16 | No input data in tape | Пустой файл |
| 17 | Invalid character in tape line 1 | Первая строка в файле ленты может задержать лишь символы «\_» и «v». |
| 18 | Can not find the head \ Several head | Неверно задано начальное положение головки на ленте(отсутствует или же задано несколько) |
| 20 | Memory allocation error | Произошла проблема выделения памяти. Требуется больше свободного места. Рекомендуется ограничить входные данные |
| 19 | Lines length mismatch in the file | Необходимо, чтобы строка с указателем и лента имели одинаковый размер |
| 21 | No transition of the symbol «» from state «» | В таблице переходов отсутствует команда для данного значения ленты и состояния. Необходимо дополнить таблицу. |
| 22 | Head index out of bounds | Головка не должны заходить за левую границу ленты (нулевое значение) |

# Методика испытаний. Моделирование ошибок

При запуске программы без аргументов происходит вывод информационного окна



Моделирование ошибок:

* Ошибка с кодом 10: по заданному адресу не существует файла table21.txt

|  |
| --- |
| Входные данные: |
| Выходные данные: |

* Ошибка с кодом 11: не указан выходной файл

|  |
| --- |
| Входные данные: |
| Выходные данные: |

* Ошибка с кодом 12: указан лишний символ «к» в первой строке

|  |  |
| --- | --- |
| Входные данные:  table.txt | Выходные данные: |

* Ошибка с кодом 13: вместо символа «q» стоит «m»

|  |  |
| --- | --- |
| Входные данные:  table.txt | Выходные данные: |

* Ошибка с кодом 14: в направлении перемещения головки указан символ «N»

|  |  |
| --- | --- |
| Входные данные:  table.txt | Выходные данные: |

* Ошибка с кодом 15: в первой строке номер состояния равен «-1»

|  |  |
| --- | --- |
| Входные данные:  table.txt | Выходные данные: |

* Ошибка с кодом 17: входной файл tape.txt пуст

|  |  |
| --- | --- |
| Входные данные:  tape.txt | Выходные данные: |

* Ошибка с кодом 18: в первой строке содержится символ «у»

|  |  |
| --- | --- |
| Входные данные:  tape.txt | Выходные данные: |

* Ошибка с кодом 18: отсутствует указатель

|  |  |
| --- | --- |
| Входные данные:  tape.txt | Выходные данные: |

* Ошибка с кодом 19: не совпадает длина строк

|  |  |
| --- | --- |
| Входные данные:  tape.txt | Выходные данные: |

* Ошибка с кодом 21: в таблице переходов отсутствует команда для значения «1», когда машина находится в состоянии 1

|  |  |
| --- | --- |
| Входные данные:  tape.txt    table.txt | Выходные данные: |

* Ошибка с кодом 22: в соответствии с командой указатель всегда будет перемещаться влево

|  |  |
| --- | --- |
| Входные данные:  tape.txt    table.txt | Выходные данные: |

Для демонстрации корректной работы программы выполним циклический сдвиг двоичного числа на заданное число разрядов.

inputTable.txt

|  |
| --- |
| .q1-.q2|L  0q1-0q1|R  1q1-1q1|R  0q2-0q12|L  1q2-0q3|R  0q3-1q3|R  .q3-.q4|R  0q4-0q5|R  1q4-1q7|R  .q4-.q4|R  0q5-0q5|R  1q5-1q5|R  .q5-.q6|L  0q6-0q9|L  1q6-0q10|L  0q7-0q7|R  1q7-1q7|R  .q7-.q8|L  0q8-1q9|L  1q8-1q10|L  0q9-0q9|L  1q9-0q10|L  .q9-.q11|L  0q10-1q9|L  1q10-1q10|L  .q10-.q11|L  0q11-0q2|H  1q11-1q2|H  .q11-.q11|L  0q12-0q12|L  1q12-0q3|R  .q12-.q0|H |

inputTape.txt

\_\_v\_\_\_\_\_\_\_\_\_\_\_\_\_

. . 01. . 001 . . . . . . .

Результат работы output.txt

|  |
| --- |
| \_\_v\_\_\_\_\_\_\_\_\_\_\_\_\_  . . 01. . 001 . . . . . . .  Command: 0q1-0q1|R  Tape:  \_\_\_v\_\_\_\_\_\_\_\_\_\_\_\_  . . 01. . 001 . . . . . . .  Command: 1q1-1q1|R  Tape:  \_\_\_\_v\_\_\_\_\_\_\_\_\_\_\_  . . 01. . 001 . . . . . . .  Command: .q1-.q2|L  Tape:  \_\_\_v\_\_\_\_\_\_\_\_\_\_\_\_  . . 01. . 001 . . . . . . .  Command: 1q2-0q3|R  Tape:  \_\_\_\_v\_\_\_\_\_\_\_\_\_\_\_  . . 00. . 001 . . . . . . .  Command: .q3-.q4|R  Tape:  \_\_\_\_\_v\_\_\_\_\_\_\_\_\_\_  . . 00. . 001 . . . . . . .  Command: .q4-.q4|R  Tape:  \_\_\_\_\_\_v\_\_\_\_\_\_\_\_\_  . . 00. . 001 . . . . . . .  Command: 0q4-0q5|R  Tape:  \_\_\_\_\_\_\_v\_\_\_\_\_\_\_\_  . . 00. . 001 . . . . . . .  Command: 0q5-0q5|R  Tape:  \_\_\_\_\_\_\_\_v\_\_\_\_\_\_\_  . . 00. . 001 . . . . . . .  Command: 1q5-1q5|R  Tape:  \_\_\_\_\_\_\_\_\_v\_\_\_\_\_\_  . . 00. . 001 . . . . . . .  Command: .q5-.q6|L  Tape:  \_\_\_\_\_\_\_\_v\_\_\_\_\_\_\_  . . 00. . 001 . . . . . . .  Command: 1q6-0q10|L  Tape:  \_\_\_\_\_\_\_v\_\_\_\_\_\_\_\_  . . 00. . 000 . . . . . . .  Command: 0q10-1q9|L  Tape:  \_\_\_\_\_\_v\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . .  Command: 0q9-0q9|L  Tape:  \_\_\_\_\_v\_\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . .  Command: .q9-.q11|L  Tape:  \_\_\_\_v\_\_\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . .  Command: .q11-.q11|L  Tape:  \_\_\_v\_\_\_\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . .  Command: 0q11-0q2|H  Tape:  \_\_\_v\_\_\_\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . .  Command: 0q2-0q12|L  Tape:  \_\_v\_\_\_\_\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . .  Command: 0q12-0q12|L  Tape:  \_v\_\_\_\_\_\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . .  Command: .q12-.q0|H  Tape:  \_v\_\_\_\_\_\_\_\_\_\_\_\_\_\_  . . 00. . 010 . . . . . . . |

## Заключение

В ходе выполнения курсовой работы были закреплены знания о машине Тьюринга. Также более детально были изучены особенности программирования на языке C.

Была реализована поставленная задача работы – машина Тьюринга с односторонней лентой, а затем проведена экспериментальная проверка правильности работы созданной программы.

## Список использованных источников

1. Справочник функций языка С [Электронный ресурс] // URL: <https://prog-cpp.ru/>
2. Машина Тьюринга [Электронный ресурс] // URL: <https://ru.wikipedia.org/wiki/Машина_Тьюринга>
3. Язык программирования Си [Электронный ресурс] // URL: <http://cppstudio.com/cat/271/>
4. Система сборки CMake [Электронный ресурс] // URL: http://sysdev.me/cmake-and-ctest/

## Приложение 1

Файл «main.c»

|  |
| --- |
| #include "input.h"  **void** inf(){  printf("Turing simulator\nCommand line format:\nTuring.exe inputTable.txt inputTape.txt output.txt -a [p]/-o"  "\n-a run in standard mode\n-o run debug\n p number of steps"  "\nFormat commands in inputTable.txt: aqn-bqk|D\na and b - the symbols on the tape\n"  "n and k - the states (the first state is always 1, the last state is always 0),\n"  "D - the direction of move (L - left, R - right, H - no move)."  "\nFormat inputTape.txt:\n\_\_\_\_\_v\_\_\_\_\n.2..39.91.\n"  "If you run debug\ns - one step,\nb - break,\ng - go the program in standard mode.\n");  }  **int** main (**int** argc, **char**\* argv[]){  FILE \*inputTable;  FILE \*inputTape;  FILE \*output;  **if**(argc == 1){  inf(); //информация о входных данных  exit(0);  }  **if**((argc == 5) && (strcmp(argv[4], "-a") == 0 || strcmp(argv[4], "-o") == 0) ||  ((argc == 6) && (strcmp(argv[4], "-a") == 0) && (checkInteger(argv[5]) == 1))){  inputTable = fopen(argv[1], "r");  **if**(inputTable == NULL){  printf("Can't find file %s", argv[1]);  exit(10);  }  inputTape = fopen(argv[2], "r");  **if**(inputTape == NULL){  printf("Can't find file %s", argv[2]);  exit(10);  }  output = fopen(argv[3], "w");  }**else**{  printf("Wrong number of arguments");  exit(11);  }  input(inputTable, inputTape, output, argc, argv);  **return** 0;  } |

## Приложение 2

Файл «input.c»

|  |
| --- |
| #include "input.h"  **int** input(FILE \*inputTable, FILE \*inputTape, FILE \*output, **int** argc, **char** \*argv[]){  // чтение таблицы состояний из файла  **char** lastChar; // символ на ленте  **char** lastState;  **long int** lastStateNum; // состояние в котором находится лента  **char** dash;  **char** newChar; // символ, который нужно записать  **char** newState;  **long int** newStateNum; // состояние, в которе должна перейти лента  **char** pipeline;  **char** move; // сдвиг указателя  **int** maxArraySize = 10, realArraySize = 0; //размер табилицы состояний  **struct** State \*arrayTable;  arrayTable = (**struct** State\*)malloc(maxArraySize\***sizeof**(**struct** State));  **if**(arrayTable == NULL){  printf("Memory allocation error");  exit(20);  }  **int** j = 0;  **while**((j = fscanf(inputTable, " %c %c %d %c %c %c %d %c %c", &lastChar, &lastState, &lastStateNum, &dash, &newChar, &newState, &newStateNum, &pipeline, &move)) != EOF){  **if**(j != 9){  printf("Wrong number of input chars");  exit(12);  }  **if**(lastState != 'q'){  errorCommand(argv[1], realArraySize, lastState, 'q');  }  **if**(newState != 'q'){  errorCommand(argv[1], realArraySize, newState, 'q');  }  **if**(dash != '-'){  errorCommand(argv[1], realArraySize, dash, '-');  }  **if**(pipeline != '|'){  errorCommand(argv[1], realArraySize, pipeline, '|');  }  **if**(move != 'L' && move != 'R' && move != 'H') {  printf("Wrong move command \"%c\"", move);  exit(14);  }  **if**(lastStateNum < 0 || newStateNum < 0){  printf("The state can not be negative");  exit(15);  }  **if**(realArraySize == maxArraySize){  maxArraySize = maxArraySize + 10;  arrayTable = (**struct** State\*)realloc(arrayTable,maxArraySize \* **sizeof**(**struct** State));  **if**(arrayTable == NULL){  printf("Memory allocation error");  exit(20);  }  }  arrayTable[realArraySize].inpCh = lastChar;  arrayTable[realArraySize].inpQ = lastStateNum;  arrayTable[realArraySize].outCh = newChar;  arrayTable[realArraySize].outQ = newStateNum;  arrayTable[realArraySize].act = move;  realArraySize++;  }  **if**(realArraySize == 0){  printf("No input data in %s\n", argv[1]);  exit(16);  }  fclose(inputTable);  //определение расположения указателя на ленте  **int** headSize = 0, numberHeads = 0, head;  **char** symbol;  **while**(((symbol = fgetc(inputTape)) != '\n')){  **if**(symbol == EOF){  printf("No input data in %s\n", argv[2]);  exit(16);  }  **if**(symbol != '\_' && symbol != 'v'){  printf("Invalid character \"%c\" in %s line 1\n", symbol, argv[2]);  exit(17);  }  **if**(symbol == 'v'){  numberHeads++;  head = headSize;  }  headSize++;  }  **if**(numberHeads != 1){  **if**(numberHeads == 0){  printf("Can not find the head in %s\n", argv[2]);  }**else**{  printf("Several head in %s\n", argv[2]);  }  exit(18);  }  //чтение ленты из файла  **int** maxTapeSize = 10, realTapeSize = 0; //длина ленты  **char** \*tape;  tape = (**char**\*)malloc(maxTapeSize\***sizeof**(**char**));  **if**(tape == NULL){  printf("Memory allocation error");  exit(20);  }  **while**((tape[realTapeSize] = fgetc(inputTape)) != EOF){  **if**(realTapeSize == maxTapeSize - 1){  maxTapeSize++;  tape = (**char**\*)realloc(tape,maxTapeSize \* **sizeof**(**char**));  **if**(tape == NULL){  printf("Memory allocation error");  exit(20);  }  }  realTapeSize++;  }  **if**(headSize != realTapeSize){  printf("Lines length mismatch in the file %s\n", argv[2]);  exit(19);  }  fclose(inputTape);  printTape(tape, head, realTapeSize, output);  **int** numberSteps;  **if**(argc == 6){  numberSteps = strtol(argv[5], NULL, 10);  }**else**{  numberSteps = 1000;  }  **int** flag = 0;  **if**(strcmp(argv[4], "-o") == 0){  flag = 1;  }  step(head, tape, arrayTable, realArraySize, realTapeSize, maxTapeSize, numberSteps, output, flag);  fclose(output);  free(arrayTable);  free(tape);  } |

## Приложение 3

Файл «input.h»

|  |
| --- |
| #ifndef INPUT\_H\_INCLUDED  #define INPUT\_H\_INCLUDED  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <malloc.h>  **struct** State{  **char** inpCh;  **long int** inpQ;  **char** outCh;  **long int** outQ;  **char** act;  };  **int** input(FILE \*inputTable, FILE \*inputTape, FILE \*output, **int** argc, **char** \*argv[]);  **int** checkInteger(**char** step[]);  **int** searchState(**struct** State \*arrayTable, **int** arraySize, **long int** lastState, **char** \*tape, **int** head);  **void** printTape(**char** \*tape, **int** head, **int** tapeSize, FILE \*output);  **void** errorCommand(**char** \*arg, **int** line, **char** symbolError, **char** symbol);  **void** printCommand(**struct** State state, FILE \*output);  **void** step(**int** head, **char** \*tape, **struct** State \*arrayTable, **int** arraySize, **int** realTapeSize, **int** maxTapeSize, **int** numberSteps,FILE \*output,**int** flag);  #endif // INPUT\_H\_INCLUDED |

## Приложение 4

Файл «help\_function.c»

|  |
| --- |
| #include "input.h"  **int** checkInteger(**char** step[]){  **if**(strtol(step, NULL, 10) != NULL){  **int** j = strlen(step);  **int** i = 0;  **while** (i < j){  **if**(atoi(&step[i]) == NULL){  **return** 0;  }  i++;  }  **if**(strtol(step, NULL, 10) < 0){  **return** 0;  }  **return** 1;  }**else**{  **return** 0;  }  }  **void** errorCommand(**char** \*arg, **int** line, **char** symbolError, **char** symbol){  printf("Incorrect command in %s line %d - \"%c\" instead of the expected \"%s\"", arg, line + 1, symbolError, symbol);  exit(13);  }  **int** searchState(**struct** State \*arrayTable, **int** arraySize, **long int** lastState, **char** \*tape, **int** head){  **int** i = 0;  **while**(arrayTable[i].inpQ != lastState || arrayTable[i].inpCh != tape[head]){  i++;  **if**(i == arraySize){  printf("No transition of the symbol %c from state %d", tape[head], lastState);  exit(21);  }  }  **return** i;  } |

## Приложение 5

Файл «step.c»

|  |
| --- |
| #include "input.h"  **void** step(**int** head, **char** \*tape, **struct** State \*arrayTable, **int** arraySize, **int** realTapeSize, **int** maxTapeSize, **int** numberSteps,FILE \*output,**int** flag){  **long int** lastState = 1;  **while**(lastState != 0 && numberSteps != 0){  **int** i = searchState(arrayTable, arraySize, lastState, tape, head);  tape[head] = arrayTable[i].outCh;  lastState = arrayTable[i].outQ;  **if**(arrayTable[i].act == 'L'){  head --;  }**else if**(arrayTable[i].act == 'R'){  head++;  }  **if**(head < 0){  printf("Head index out of bounds\n");  exit(22);  }  **if**(head == realTapeSize){  tape[head] = '.';  realTapeSize++;  **if**(head == maxTapeSize){  maxTapeSize++;  tape = (**char**\*)realloc(tape,maxTapeSize \* **sizeof**(**char**));  }  }  numberSteps--;  printCommand(arrayTable[i], output);  printTape(tape, head, realTapeSize, output);  **if** (flag == 1) {  **char** comand[2]; //сюда будет записана команда  **while** (1) {  scanf(" %2s", comand);  fflush(stdin); //очистка буфера  **if** (!strcmp(comand, "b")) {  printf("Program break\n");  fclose(output);  exit(0);  }  **if** (!strcmp(comand, "s")) {  **break**;  }  **if** (!strcmp(comand, "g")) {  flag = 0;  **break**;  } **else** printf("No such instruction. Enter other\n");  }  }  }  } |

## Приложение 6

Файл «output.c»

|  |
| --- |
| #include "input.h"  **void** printTape(**char** \*tape, **int** head, **int** tapeSize, FILE \*output){  printf("Tape:\n");  fprintf(output, "Tape:\n");  **int** i = 0;  **while**(i < tapeSize){  **if**(i == head){  printf("v");  fprintf(output, "v");  }**else**{  printf("\_");  fprintf(output, "\_");  }  i++;  }  printf("\n");  fprintf(output,"\n");  i = 0;  **while**(i < tapeSize){  printf("%c", tape[i]);  fprintf(output, "%c", tape[i]);  i++;  }  printf("\n");  fprintf(output,"\n");  }  **void** printCommand(**struct** State state, FILE \*output){  printf("Command: ");  fprintf(output, "Command: ");  printf("%cq%d-%cq%d|%c", state.inpCh, state.inpQ, state.outCh, state.outQ, state.act);  fprintf(output, "%cq%d-%cq%d|%c", state.inpCh, state.inpQ, state.outCh, state.outQ, state.act);  printf("\n");  fprintf(output,"\n");  } |