**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

**Курсовая работа**

**по дисциплине «Программирование»**

Тема: **Обработка текстовой информации**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3312 |  | Мохно Даниил. |
| Преподаватель |  | Аббас Саддам |

Санкт-Петербург

2023

**Цель работы.**

Законченное поэтапное решение содержательной задачи (постановка задачи, спецификация, выбор структур данных и разработка алгоритма, программная реализация, тестирование).

**Задание (вариант 99)**

Разработать алгоритм и написать программу, заменяющую в тексте программы на Си комментарии в стиле С++ ( // comment text ) на комментарии в стиле Си ( /\* comment text \*/ ) с сохранением текста комментария.

**Постановка задачи и описание решения**

* При запуске программа очищает консоль, и запускает функцию вывода меню. Очистка консоли происходит при вызове функции system, c параметром “clear” на unix-подобных системах или “cls” на windows. Объявление макроса с использованием этой функции с одним из двух аргументов происходит за счёт директив условной компиляции.
* Функция меню – запускает цикл, в котором выводит на экран опции программы, а затем запрашивает у пользователя выбранную им опцию. Далее сравнивает значение с заложенными, с помощью оператора выбора. Если значение – 1, то вызывается функция помощи. Если значение – 2, то вызывается функция запуска с режимом чтения из файла. Если значение – 3, вызывается функция запуска с режимом ввода текса в консоль. Если значение – 0, то цикл прерывается и программа завершается. Если значение не совпадает ни с одним из заложенных, выводится сообщение о некорректном значении.
* Функция помощи, выводит сообщение с описанием программы.
* Функция запуска с режимом чтения из файла, запрашивает у пользователя имя открываемого файла, вычисляет его длину, и заменяет в имени символ перехода на новую строку на символ окончания строки (индекс этого символа – (длинна строки – 1). Далее открывает файл с переданным нами именем на чтение и, если файл существует, взывает функцию запуска процессов, в которую передаёт объект файла. Затем закрывает файл. Если при закрытии файла возникла ошибка, выводится «Error closing input file! Try again», а если файл даже не открылся, вероятно он не существует, поэтому выводим «Input opening error  
  Make sure the file exists»
* Функция запуска процессов, создаёт двумерный массив символов, который является нашим многострочным текстом, переменную, хранящую кол-во его строк, и массив длин всех строк. Далее вызывает функцию получения текста, в которую передаёт массив текста, массив с длинами строк, объект файла и указатель на кол-во строк в тексте, затем, в случае, когда параметром был передан стандартный файл ввода, через условную компиляцию на unix-подобных системах мы вызываем функцию пере открытия стандартного файла ввода, из-за того, что после завершения ввода командой ctrl+D файл ввода закрывается. Затем вызывается функция замены комментариев, принимающая параметрами массив текста, массив с длинами строк и кол-во строк, за ней вызывается функция вывода/записи текста, принимающая массив текста, объект файла/файл стандартного вывода и кол-во строк. И наконец функция сохранения текста в файл, принимающая массив текста и кол-во строк.
* В функции получения текста, перед началом цикла мы ставим флаг пропуска нулевой строки равным 1, он нужен, потому что, когда файлом для чтения является стандартный файл ввода (stdin), первой будет читаться пустая строка, даже если мы её не вводим. Далее функция проходится циклом, получая из файла строку, пока операция получения строки выполняется. После вычисляется длинна строки. Если наш файл – это стандартный файл ввода и флаг равен 1, то мы ставим флаг равным нулю, иначе она копирует строку в массив, добавляет длину строки по соответствующему значению указателя на кол-во строк и инкрементирет кол-во строк и итератор. Таким образом мы пропустим ненужную нам строку при вводе из консоли.
* Макрофункцию пере открытия стандартного файла ввода мы объявляем в условной компиляции на unix-подобных системах. Функция открывает в объект stdin файл /dev/tty, который является файл-устройством (COM-портом), к которому подключён наш терминал.
* Функция замены комментариев присваиваетфлагу, указывающему, выполняется ли многострочный блок комментариев 0, и перебирает циклом строки текста. В цикле функция проверяет, если первые два символа текущей и следующей строки – это / и флаг равен 0, то значит начался многострочный комментарий, второй символ строки мы заменяем на \*, а флаг меняем на 1**.**  Иначе функция проверяет чтобы флаг был равен 1, если да, то выполняется обработка многострочного комментария, первый элемент строки мы меняем на пробел, а второй на \*, если у следующей строки первые два символа не /, то мы обрабатываем последний комментарий многострочного комментария, если третий символ не \n то мы вызываем ф-ю закрытия комментариев передавая ей текст, индекс текущей строки и массив с длинами строк. В случае, когда третий символ – это \n, изначальный комментарий состоял всего из двух /, значит можно просто заменить \n на / и после него добавить \n. В любом случае после закрытия последнего комментария, флаг делаем равным 0. В том случае, когда комментарий не многострочный, то есть флаг не равен 1, происходит обработка однострочного комментария. Поставим флаг наличия комментария в строке 0 и пройдёмся циклом по строке. Если встречаем в строке два идущих подряд символа /, то второй меняем на \*, а флаг наличия комментария ставим равным 1. После прохода по строке проверяем, чтобы этот флаг был равен 1, тогда вызываем функцию закрытия комментария и передаём ей текст, индекс текущей строки и массив с длинами строк.
* Функция закрытия комментария получает по индексу текущей строки её длину из массива длин, значение этой длинны – это индекс последнего элемента строки. По нему мы можем вычислить предпоследний и следующие индексы символов строки. Так как последний символ \0, а предпоследний \n, заменяем предпоследний символ строки на пробел, последний на \*, и добавляем после ещё два символа – / и \n.
* Функция записи текса, проходится циклом по тексту записывая каждую строку в файл. Если в качестве файла был передан стандартный файл вывода (stdout), то все записанные на него строки просто выведутся в консоль.
* Функция сохранения текста в файл, спрашивает у пользователя хочет ли он сохранить текст в файл, если пользователь введёт Y или y, то у пользователя запрашивается имя файла в который мы сохраним текст, далее вычисляем длину имени файла и заменяем в нём символ перехода на новую строку (\n) символом окончания строки (\0) (индекс этого символа – (длинна строки – 1), и открываем файл с таким именем на запись. Если файла не существует, он создастся автоматически. Если файл открылся или создался, мы вызываем функцию записи текста, которой передаём массив текста, открытый нами файл, кол-во строк в тексте. За тем закрываем файл. Если закрытие прошло без ошибки выводим «Successfully preserved» в ином случае выводим «Output closing error». Если же файл даже не открылся и не создался, тогда выводим «Output opening error».
* Функция запуска программы с режимом ввода в консоль выводит сообщение о том, что пользователь может вводить текст, и для того, чтобы остановить ввод нужно нажать ctrl+Z на Windows или ctrl+D на unix-подобных системах. Затем вызывается функция запуска процессов, в которую передаём стандартный файл ввода (stdin).

**Описание переменных**

* **Функция вывода меню - menu()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | value | int | Значение параметра меню, введённое пользователем |

* **Функция для выполнения программы с вводом из файла - run\_With\_File\_Input()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | file | FILE\* | Обьект входного файла |
| 2 | input\_file\_name[] | char | Имя вводимого файла |
| 4 | len | int | Длина имени входного файла |

* **Функция, вызывающая функции для обработки введенного текста - text\_processing(FILE \*file)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | file | FILE\* | Файл для чтения пользователем или файл ввода из кнсоли stdin |
| 2 | text[][] | char | Текст из файла |
| 3 | rows\_lens[] | int | Массив длин каждой из строк текста |
| 4 | row\_numbers | int | количество строк в тексте |

* **Функция, для получения текста из файла - get\_text(char (\*text)[], int rows\_lens[], FILE \*file, int \*row\_numbers)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | file | FILE\* | Файл для чтения пользователем |
| 2 | text[][] | char | Текст из файла |
| 3 | rows\_lens[] | int | Массив длин каждой из строк текста |
| 4 | row\_numbers | int\* | Указатель на количество строк в тексте |
| 5 | char string[] | char | текущая строка текста из файла в цикле |
| 6 | stlen | int | текущая длина строки из текста в цикле |
| 7 | stdin\_flag | int | Флаг для исключения первой строки при чтении из консоли |
| 8 | i | int | Итератор |

* **Функция, для замены комментариев в тексте - comment\_replace(char (\*text)[ ], int rows, int rows\_lens[])**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | text[][] | char | Текст из файла |
| 2 | rows\_lens[] | int | Массив длин каждой из строк текста |
| 3 | row\_numbers | int | Количество строк в тексте |
| 4 | row | int | Итератор для строк в тексте |
| 5 | col | int | Итератор для столбцов в каждой строке |
| 6 | haveComment | int | Флаг, указывающий, есть ли комментарий в строке |
| 7 | isStillComment | int | Флаг, указывающий, выполняется ли многострочный блок комментариев |

* **Функция для закрытия блока комментариев - close\_comment(char (\*text)[], FILE \*file,int row, int rows\_lens[])**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | text[][] | char | Текст из файла |
| 2 | rows\_lens[] | int | Массив длин каждой из строк текста |
| 3 | row | int | Индекс строки, в которой нужно закрыть, комментарий |

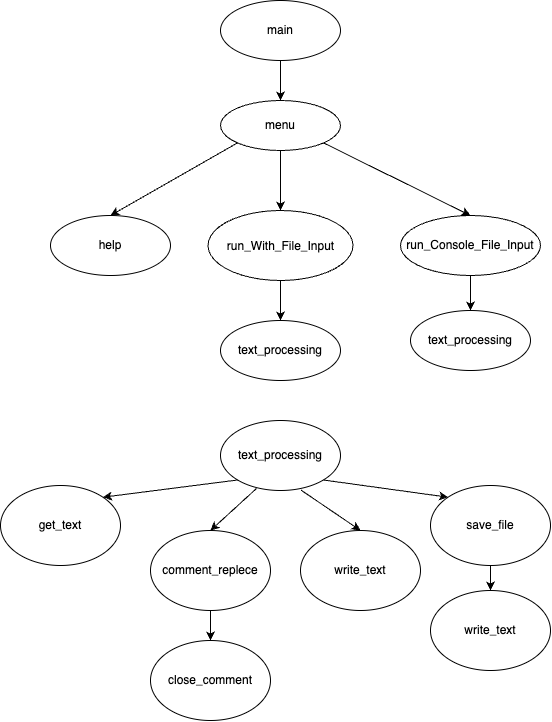
* **Функция для вывода текста на консоль - write\_text(char (\*text)[STRING\_SIZE], FILE file, int row\_numbers)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | file | FILE\* | Объект выходного файла (stdout для вывода на экран) |
| 2 | text[][] | char | Текст из файла |
| 3 | row\_numbers | int | Количество строк в тексте |

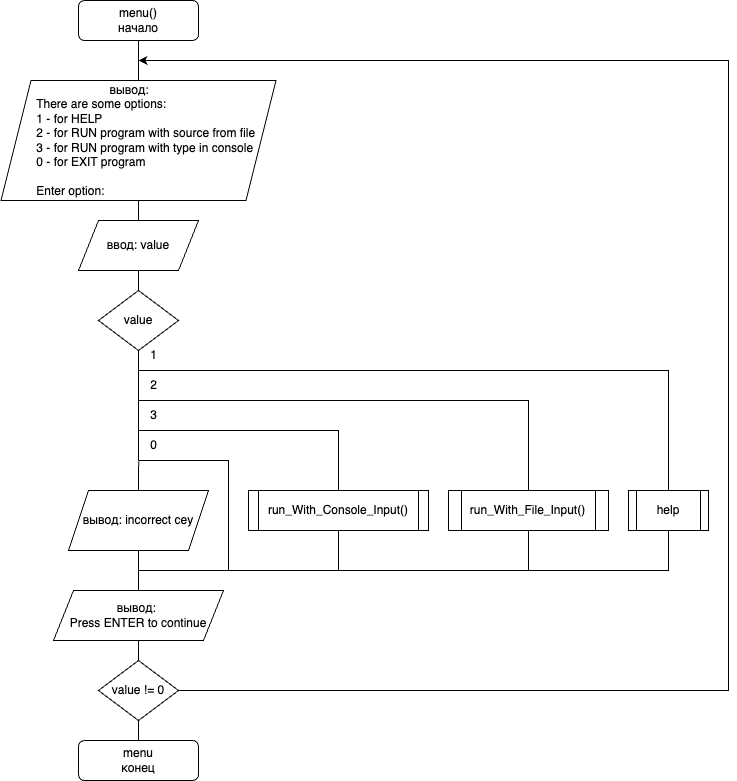
* **Функция для сохранения измененного текста в файл - save\_file(char (\*text)[STRING\_SIZE], int row\_numbers)**

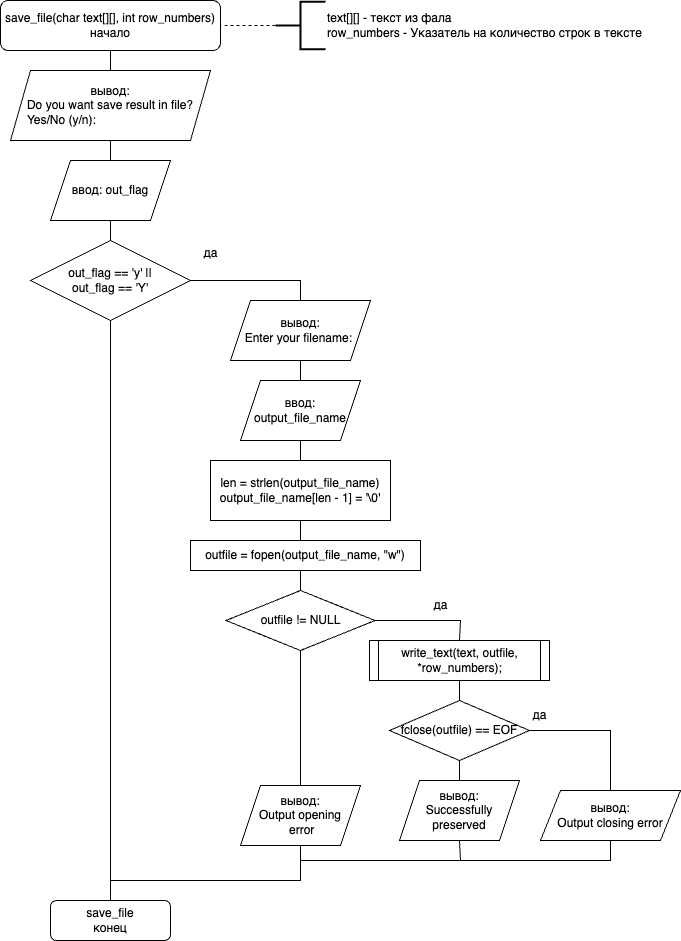
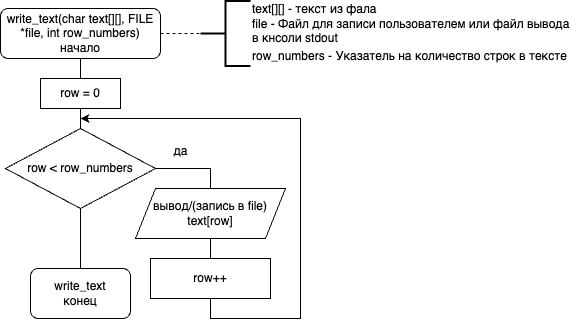
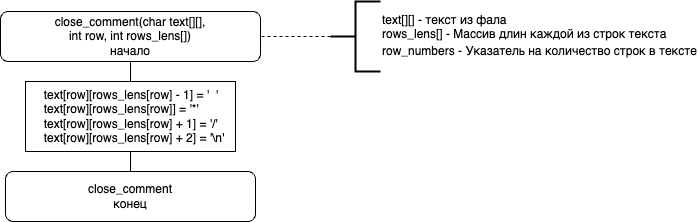
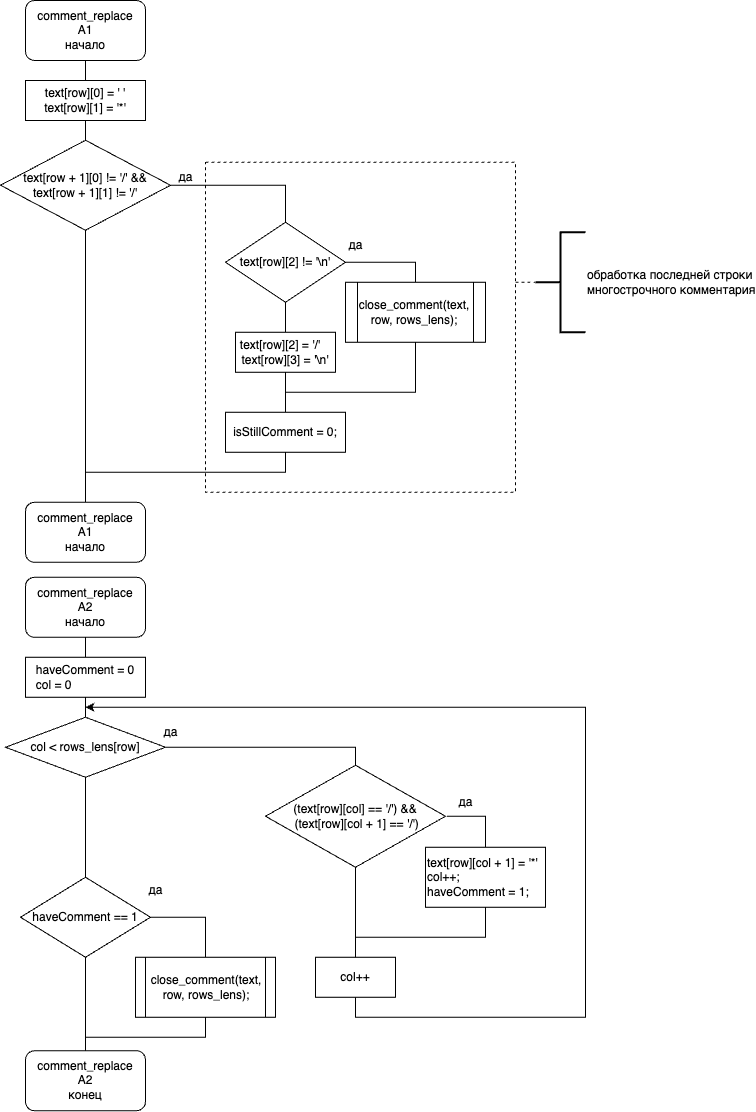
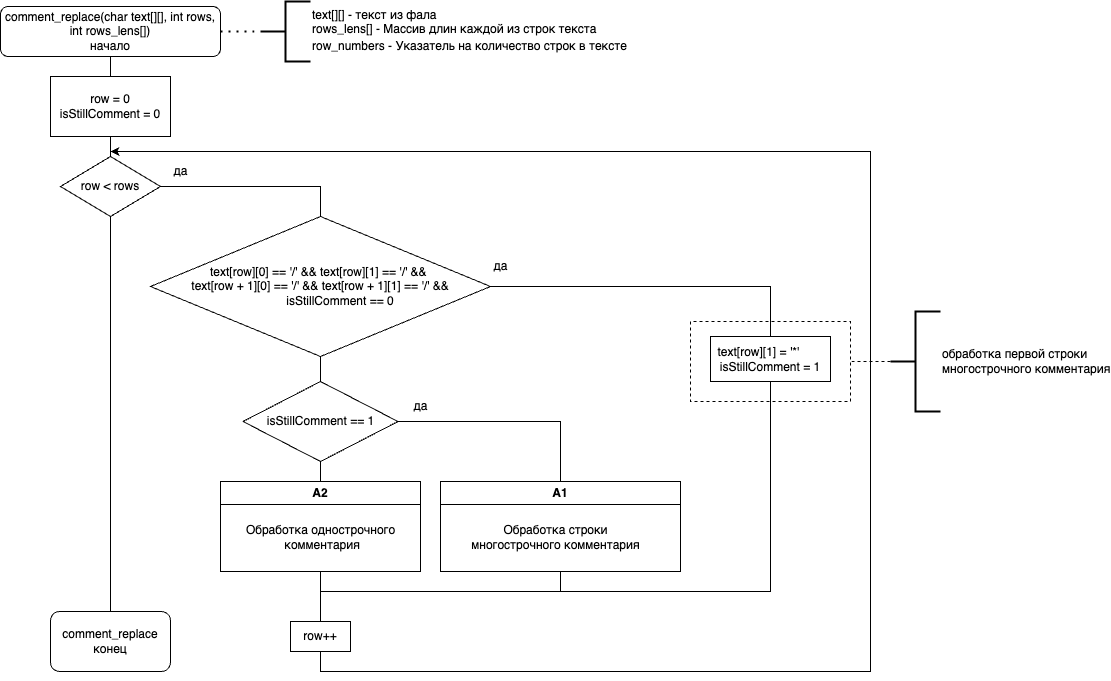
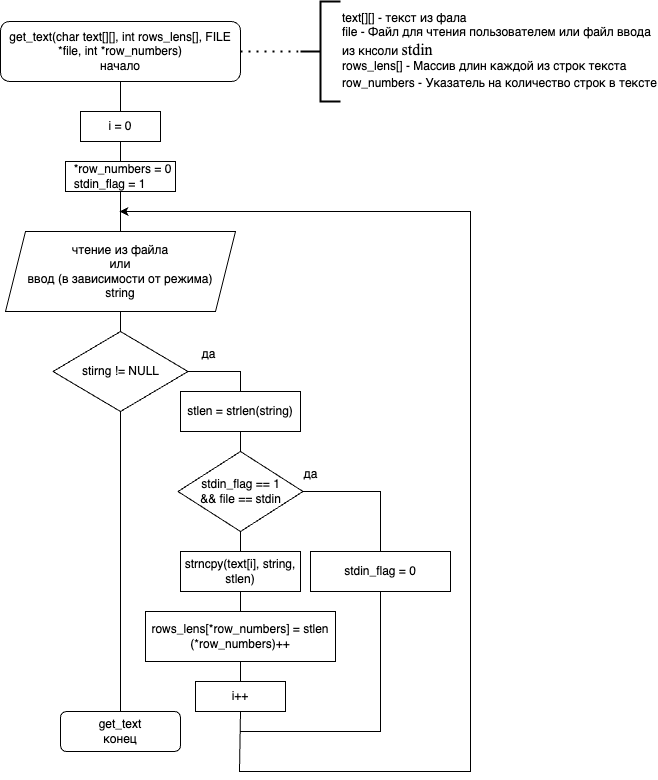
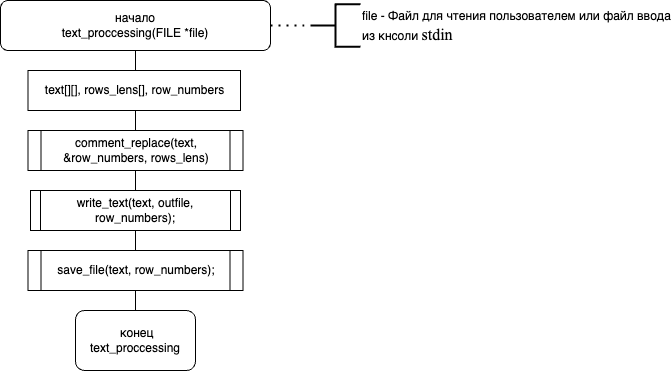
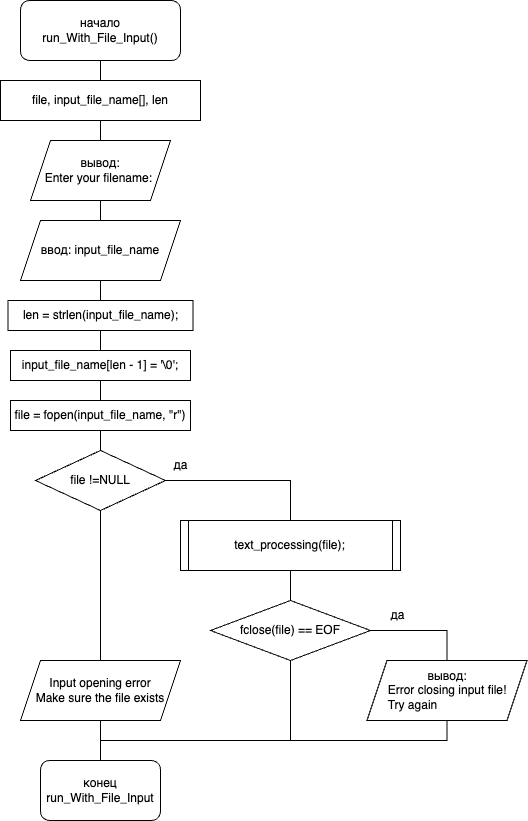
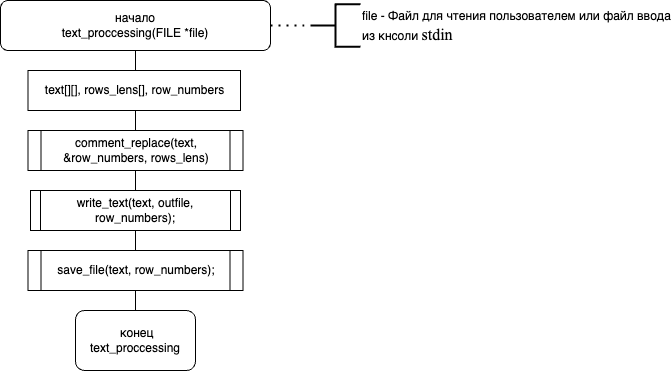
|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | outfile | FILE\* | Объект выходного файла |
| 2 | text[][] | char | Текст из файла |
| 3 | output\_file\_name[] | char | Имя выходного файла |
| 4 | out\_flag | char | Переменная для сохранения выбора пользователя относительно того, хочет ли он сохранить результат в файл |
| 5 | row\_numbers | int | Количество строк в тексте |
| 6 | len | int | Длина имени выходного файла |

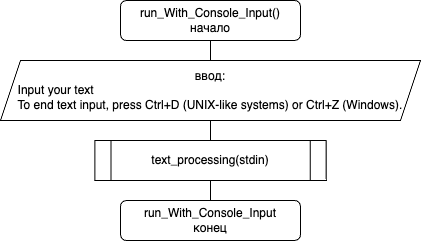
**Структура вызова функций**

****

**Схема алгоритма**

****

****

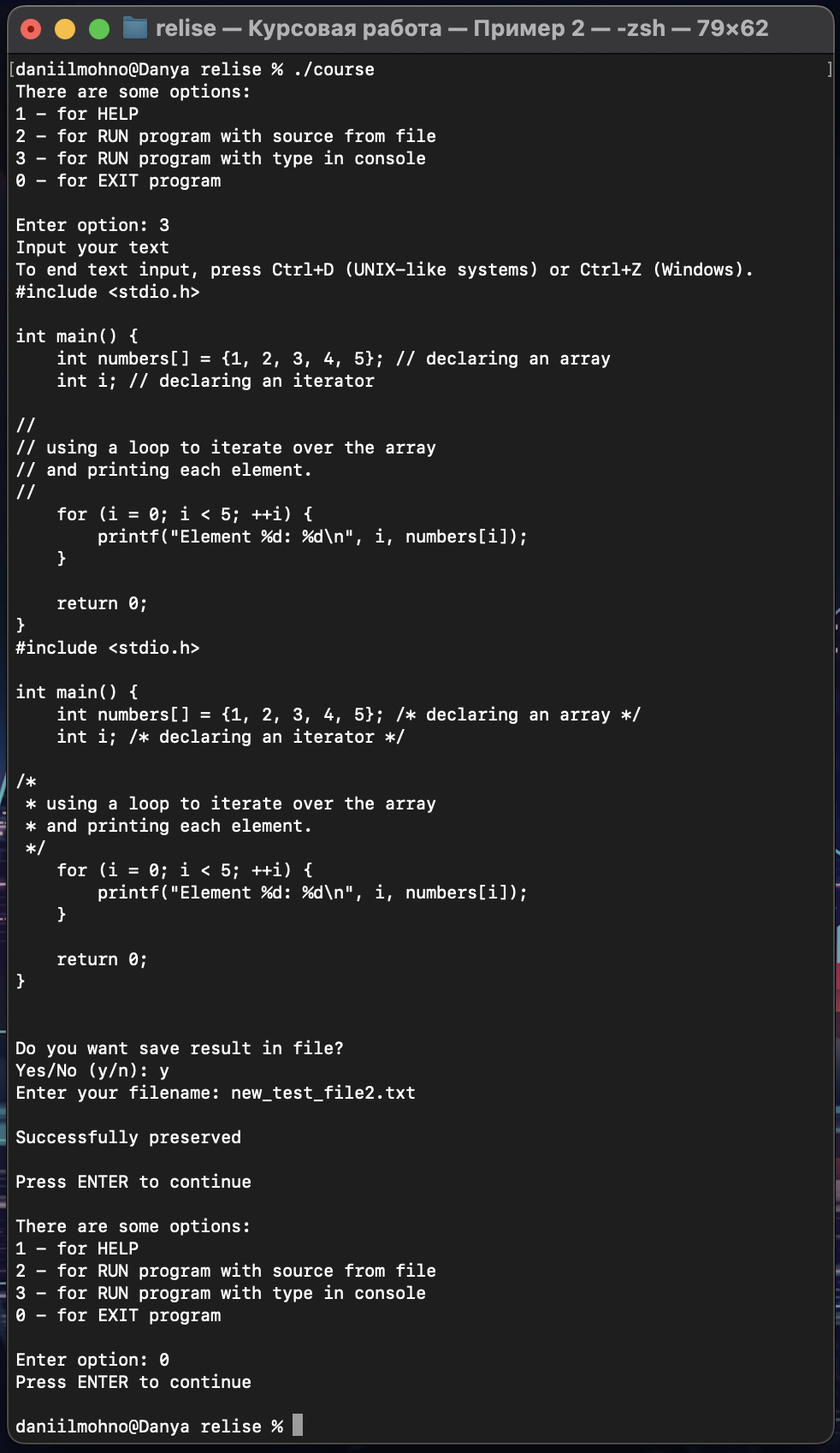
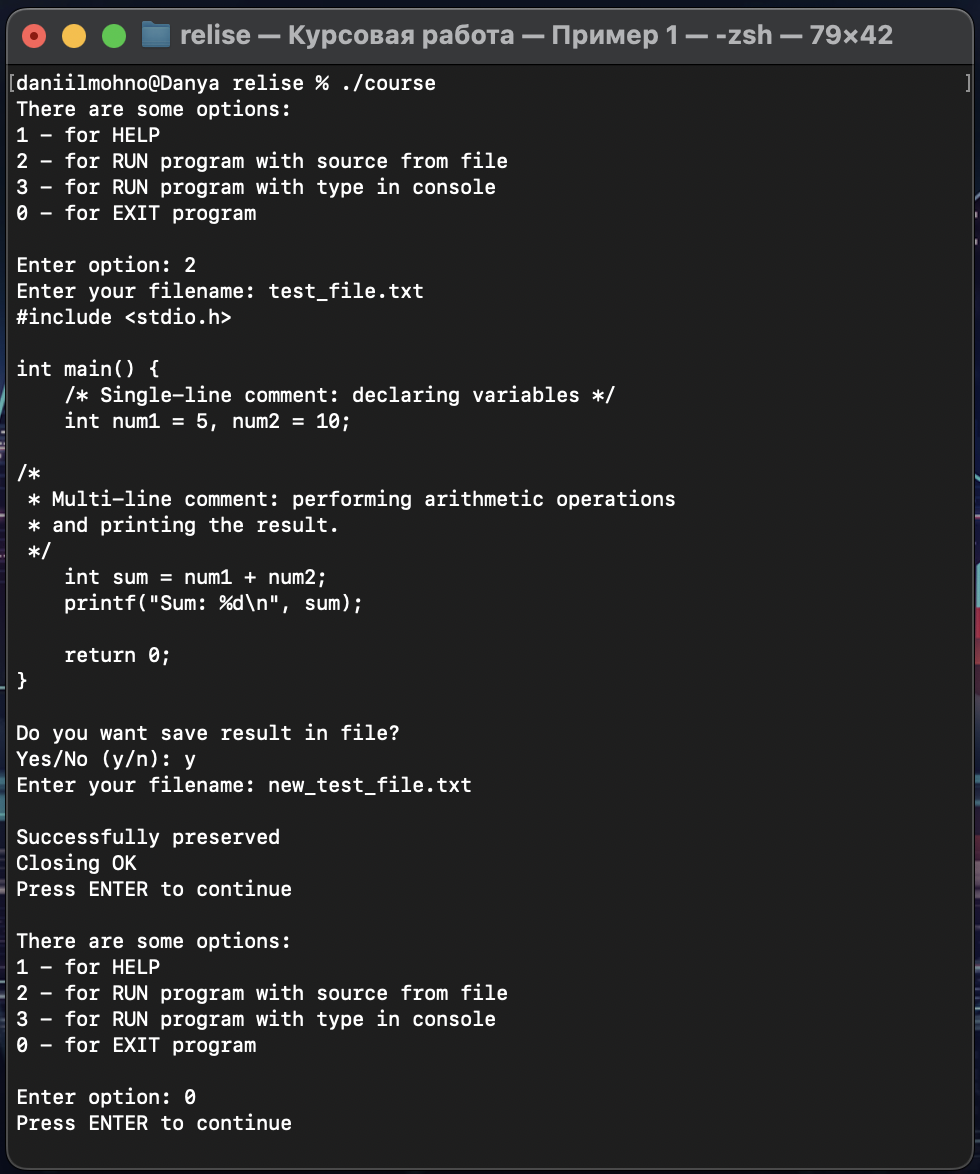
****

**Контрольные примеры**

|  |  |  |
| --- | --- | --- |
| № | Исходный текст | Полученный текст |
| 1 | #include <stdio.h>  int main() {  // declaring variables  int num1 = 5, num2 = 10;  //  // performing arithmetic operations  // and printing the result.  //  int sum = num1 + num2;  printf("Sum: %d\n", sum);  return 0;  } | #include <stdio.h>  int main() {  /\* declaring variables \*/  int num1 = 5, num2 = 10;  /\*  \* performing arithmetic operations  \* and printing the result.  \*/  int sum = num1 + num2;  printf("Sum: %d\n", sum);  return 0;  } |
| 2 | #include <stdio.h>  int main() {  int numbers[] = {1, 2, 3, 4, 5}; // declaring an array  int i; // declaring an iterator  //  // using a loop to iterate over the array  // and printing each element.  //  for (i = 0; i < 5; ++i) {  printf("Element %d: %d\n", i, numbers[i]);  }  return 0;  } | #include <stdio.h>  int main() {  int numbers[] = {1, 2, 3, 4, 5}; /\* declaring an array \*/  int i; /\* declaring an iterator \*/  /\*  \* using a loop to iterate over the array  \* and printing each element.  \*/  for (i = 0; i < 5; ++i) {  printf("Element %d: %d\n", i, numbers[i]);  }  return 0;  } |

**Текст программы**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define FILE\_NAME\_SIZE 50  #define STRING\_SIZE 200  #define MAX\_ROW\_NUMBER 1000  #ifdef WIN32  #define CLS system("cls")  #else  #define CLS system("clear")  #endif  /\* Function to display the main menu \*/  void menu();  /\* Function to display help information \*/  void help();  /\* Function to execute the program with input from a file \*/  void run\_With\_File\_Input();  /\* A function that calls functions to process the entered text, make the necessary changes and display the results \*/  void text\_processing(FILE \*file);  /\* Function to get text from a file \*/  void get\_text(char (\*text)[STRING\_SIZE], int rows\_lens[], FILE \*file, int \*row\_numbers);  /\* Reset standard input function on UNIX-like systems \*/  #ifndef WIN32  #define RESET\_STDIN() \  if ((stdin = fopen("/dev/tty", "r")) == NULL) \  perror("Input error");  #endif  /\* Function to replace comments in the text \*/  void comment\_replace(char (\*text)[STRING\_SIZE], int rows, int rows\_lens[]);  /\* Function to close a comment block \*/  void close\_comment(char (\*text)[STRING\_SIZE], int row, int rows\_lens[]);  /\* Function to print text to the console \*/  void write\_text(char (\*text)[STRING\_SIZE], FILE \*file, int row\_numbers);  /\* Function to save the modified text to a file \*/  void save\_file(char (\*text)[STRING\_SIZE], int row\_numbers);  /\* Function to execute the program with console input \*/  void run\_With\_Console\_Input();  int main()  {  CLS;  menu();  return 0;  }  void menu()  {  int value; /\* Value of menu parameter entered by user \*/  do  {  puts("There are some options:");  puts("1 - for HELP");  puts("2 - for RUN program with source from file");  puts("3 - for RUN program with type in console");  puts("0 - for EXIT program");  printf("\nEnter option: ");  scanf("%d", &value);  switch (value)  {  case 1:  {  CLS;  help();  getchar();  }  break;  case 2:  {  CLS;  run\_With\_File\_Input();  getchar();  }  break;  case 3:  {  CLS;  run\_With\_Console\_Input();  getchar();  }  break;  case 0:  {  getchar();  }  break;  default:  {  puts("Incorrect key!");  getchar();  }  }  puts("Press ENTER to continue");  getchar();  CLS;  } while (value != 0);  }  void help()  {  printf("\n\nPROGRAM DESCRIPTION\n\n"  "This program provides functionality for processing and modifying text files.\n"  "It allows you to replace inline comments in the text, format the output, and save the results to a file.\n\n"  "OPTIONS:\n"  "1 - HELP: Display this help message.\n"  "2 - for RUN program with source from file: Execute the program with input read from a file.\n"  "3 - for RUN program with type in console: Execute the program with input entered interactively in the console.\n"  "0 - EXIT program: Exit the program.\n\n"  "USAGE:\n"  "- Choose the appropriate option from the main menu.\n"  "- For 'RUN program with source from file', provide the filename when prompted.\n"  "- For 'for RUN program with type in console', input text interactively until finished.\n"  "- Results can be saved to a file upon request.\n\n"  "NOTE:\n"  "To end text input in console mode, press Ctrl+D (UNIX-like systems) or Ctrl+Z (Windows).\n");  }  void run\_With\_File\_Input()  {  FILE \*file; /\* Input file object \*/  char input\_file\_name[FILE\_NAME\_SIZE]; /\* Input file name \*/  int len; /\* Length of the input file name \*/  getchar();  printf("Enter your filename: ");  fgets(input\_file\_name, FILE\_NAME\_SIZE, stdin);  len = strlen(input\_file\_name);  input\_file\_name[len - 1] = '\0';  if ((file = fopen(input\_file\_name, "r")) != NULL)  {  text\_processing(file);  if (fclose(file) == EOF)  printf("Error closing input file!\nTry again\n");  }  else  {  printf("\nInput opening error\nMake sure the file exists\n\n");  }  }  void text\_processing(FILE \*file)  {  char text[MAX\_ROW\_NUMBER][STRING\_SIZE]; /\* text \*/  int rows\_lens[MAX\_ROW\_NUMBER]; /\* Lengths of each of the lines \*/  int row\_numbers; /\* number of lines in the text \*/  get\_text(text, rows\_lens, file, &row\_numbers);  #ifndef WIN32  if (file == stdin)  RESET\_STDIN()  #endif  comment\_replace(text, row\_numbers, rows\_lens);  write\_text(text, stdout, row\_numbers);  save\_file(text, row\_numbers);  }  void get\_text(char (\*text)[STRING\_SIZE], int rows\_lens[], FILE \*file, int \*row\_numbers)  {  char string[STRING\_SIZE]; /\* current line of text from the file in the loop \*/  int i; /\* iterator \*/  int stlen; /\* current length of a line of text in a loop \*/  int stdin\_flag; /\* Flag to exclude the first line when reading from stdin \*/  \*row\_numbers = 0;  stdin\_flag = 1;  i = 0;  while (fgets(string, STRING\_SIZE, file) != NULL)  {  stlen = strlen(string);  if (stdin\_flag && file == stdin)  {  stdin\_flag = 0;  }  else  {  strncpy(text[i], string, stlen);  rows\_lens[\*row\_numbers] = stlen;  (\*row\_numbers)++;  i++;  }  }  }  void comment\_replace(char (\*text)[STRING\_SIZE], int rows, int rows\_lens[])  {  int row; /\* Iterator for rows in the text \*/  int col; /\* Iterator for columns in each row \*/  int haveComment; /\* Flag to indicate whether a line has a comment \*/  int isStillComment; /\* Flag to indicate whether a multiline comment block is in progress \*/  isStillComment = 0;  for (row = 0; row < rows; row++)  {  if (text[row][0] == '/' && text[row][1] == '/' && text[row + 1][0] == '/' && text[row + 1][1] == '/' &&  !isStillComment)  {  text[row][1] = '\*';  isStillComment = 1;  }  else if (isStillComment)  {  text[row][0] = ' ';  text[row][1] = '\*';  if (text[row + 1][0] != '/' && text[row + 1][1] != '/')  {  if (text[row][2] != '\n')  {  close\_comment(text, row, rows\_lens);  }  else  {  text[row][2] = '/';  text[row][3] = '\n';  }  isStillComment = 0;  }  }  else  {  haveComment = 0;  for (col = 0; col < rows\_lens[row]; col++)  {  if ((text[row][col] == '/') && (text[row][col + 1] == '/'))  {  text[row][col + 1] = '\*';  col++;  haveComment = 1;  }  }  if (haveComment)  {  close\_comment(text, row, rows\_lens);  }  }  }  }  void close\_comment(char (\*text)[STRING\_SIZE], int row, int rows\_lens[])  {  text[row][rows\_lens[row] - 1] = ' ';  text[row][rows\_lens[row]] = '\*';  text[row][rows\_lens[row] + 1] = '/';  text[row][rows\_lens[row] + 2] = '\n';  }  void write\_text(char (\*text)[STRING\_SIZE], FILE \*file, int row\_numbers)  {  int row; /\* Iterator for rows in the text \*/  for (row = 0; row < row\_numbers; row++)  fputs(text[row], file);  }  void save\_file(char (\*text)[STRING\_SIZE], int row\_numbers)  {  FILE \*outfile; /\* Output file object \*/  char output\_file\_name[FILE\_NAME\_SIZE]; /\* Output file name \*/  char out\_flag; /\* User input flag for saving to a file \*/  int len; /\* Length of the output file name \*/  printf("\n\nDo you want save result in file?\nYes/No (y/n): ");  scanf("%c", &out\_flag);  if (out\_flag == 'y' || out\_flag == 'Y')  {  getchar();  printf("Enter your filename: ");  fgets(output\_file\_name, FILE\_NAME\_SIZE, stdin);  len = strlen(output\_file\_name);  output\_file\_name[len - 1] = '\0';  outfile = fopen(output\_file\_name, "w");  if (outfile != NULL)  {  write\_text(text, outfile, row\_numbers);  if (fclose(outfile) == EOF)  perror("\nOutput closing error");  else  printf("\nSuccessfully preserved\n");  }  else  perror("Output opening error");  }  }  void run\_With\_Console\_Input()  {  printf("Input your text\nTo end text input, press Ctrl+D (UNIX-like systems) or Ctrl+Z (Windows).\n");  text\_processing(stdin);  }  } |

**Примеры выполнения программы**

**Выводы.**

В результате выполнения работы изучены **oобработка текстовой информации, и работа с файлами на языке си** и получены практические навыки в программировании на этом языке.