**Отчет по рубежному контролю №3**

*РК6-16Б*

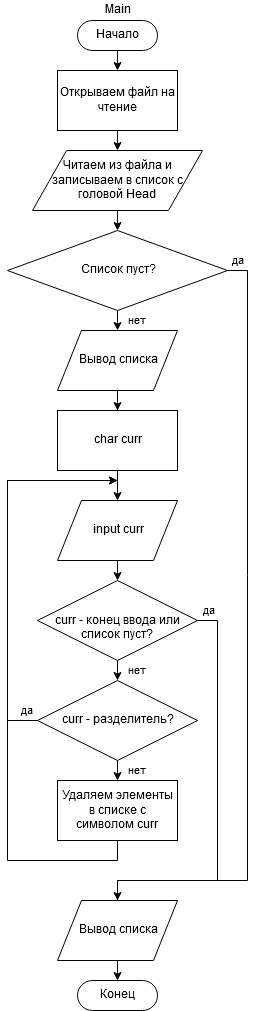
16. Петраков Станислав Альбертович

Преподаватель: Федорук Елена Владимировна

**Задача**

Разработайте программу, которая вводит из файла, имя которого определяется в командной строке, слова и помещает их в линейный односвязный список. Затем удаляет из списка все элементы, которые содержат те символы, которые вводятся со стандартного потока ввода. Слова из результирующего списка вывести в стандартный поток вывода.

**Блок схема**



**Код программы**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define BUFFER\_SIZE 1024

#define TRUE 1

#define FALSE 0

/\*

exits:

1: invalid arguments

2: Can't open file

3: memory allocation error

\*/

typedef struct element {

char\* str;

struct element\* next;

}element\_s;

element\_s\* Head = NULL;

void tempoutput();

void output();

void addElement(char\* word);

void input(FILE\* filename);

\_Bool findElement(char c);

void deleteElement(element\_s\*\* prev, element\_s\*\* curr);

void tempoutput()

{

printf("Temp output:\n");

if (Head == NULL)

{

printf("Empty list.\n\n\n");

return;

}

element\_s\* temp;

for (temp = Head; temp != NULL; temp = temp->next)

printf("%s\n", temp->str);

printf("\n\n");

}

void output()

{

printf("Result output:\n");

if (NULL == Head)

{

printf("Empty list.\n\n\n");

return;

}

element\_s\* temp;

while (NULL != Head)

{

printf("%s\n", Head->str);

temp = Head;

Head = Head->next;

free(temp);

}

}

void input(FILE\* filename)

{

char buffer[BUFFER\_SIZE];

char\* temp;

while (fscanf(filename, "%s", buffer) != EOF)

{

if (NULL == (temp = (char\*)malloc(sizeof(char) \* strlen(buffer))))

exit(3);

strcpy(temp, buffer);

addElement(temp);

}

}

void addElement(char\* word)

{

element\_s\* temp;

if (NULL == (temp = (element\_s\*)malloc(sizeof(element\_s))))

exit(3);

temp->str = word;

temp->next = Head;

Head = temp;

}

\_Bool findElement(char c)

{

element\_s\* curr = Head, \* prev = Head;

\_Bool isFounded =FALSE;

while (NULL != curr)

{

if (strchr(curr->str, c))

{

isFounded = TRUE;

deleteElement(&prev, &curr);

}

else

{

prev = curr;

curr = curr->next;

}

}

return isFounded;

}

void deleteElement(element\_s\*\* prev, element\_s\*\* curr)

{

printf("Found element: %s. Deleting...\n", (\*curr)->str);

if ((\*curr) == Head)

{

Head = Head->next;

free(\*curr);

\*curr = Head;

\*prev = Head;

return;

}

else

{

(\*prev)->next = (\*curr)->next;

free(\*curr);

\*curr = (\*prev)->next;

}

}

int main(int argc, char\* argv[])

{

//Preparation

if (argc != 2)

exit(1);

FILE\* filename;

if (NULL == (filename = fopen(argv[1], "r")))

exit(2);

//All preparation ready

input(filename);

fclose(filename);

//If the list is empty, then nothing

if (NULL != Head)

{

//temp output

tempoutput();

//Processing

char curr;

while (EOF != (curr = getchar()) && NULL != Head && '~' != curr)

if (' ' != curr && '\n' != curr && '\t' != curr)

if (Head != NULL)

{

printf("Searches for elements with symbol '%c'. After search, print list.\n", curr);

if(findElement(curr))

tempoutput();

else

printf("Not found.\n\n\n");

}

}

//Output

output();

return 0;

}

**Тестовые запуски**

|  |
| --- |
| Temp output:  q  q  q  q  q  q  Searches for elements with symbol 'q'. After search, print list.  Found element: q. Deleting...  Found element: q. Deleting...  Found element: q. Deleting...  Found element: q. Deleting...  Found element: q. Deleting...  Temp output:  Empty list.  Result output:  Empty list. |
| Пустой файл  Result output:  Empty list. |
| Temp output:  q  q  Searches for elements with symbol 'q'. After search, print list.  Found element: q. Deleting...  Temp output:  Empty list.  Result output:  Empty list. |
| Temp output:  wqe  sad  wqeqw  wejhr  rekjhw  i  quyew  w  Searches for elements with symbol 'w'. After search, print list.  Found element: wqe. Deleting...  Found element: wqeqw. Deleting...  Found element: wejhr. Deleting...  Found element: rekjhw. Deleting...  Found element: quyew. Deleting...  Temp output:  sad  i  ~  Result output:  sad  i |