

МИНОБРНАУКИ РОССИИ  
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
«ЛЭТИ» им.В.И.УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Отчет по лабораторной работе № 10  
по дисциплине «Программирование»  
Тема: «Односвязный список»

Студент гр. 9305

Любаневич Р.О.

Преподаватель

Перязева Ю.В.

Санкт-Петербург  
2020

## Содержание

Цель.....	3
Задание .....	3
Постановка задачи и описание решения.....	3
Описание переменных .....	4
Контрольные примеры .....	4
Схема алгоритма.....	5
Текст программы.....	6
Пример работы программы .....	7
Исходные данные.....	7
Вывод программы .....	7
Выводы .....	8

## Цель

Получить практические навыки в разработке алгоритма и написании программы на языке Си для работы с односвязным списком.

## Задание

Разработать подалгоритм удаления в односвязном списке предпоследнего элемента. При недостаточном количестве элементов в списке вывести соответствующее сообщение.

## Постановка задачи и описание решения

Имеется файл, содержащий строки данных, которые должны стать значениями полей структур, на каждой строке — новая структура. Значения полей на одной строке отделены друг от друга специальным символом, который будет индикатором окончания считывания значения одного поля структуры и сигналом к началу считывания значения другой. Структура имеет указатель на следующий элемент, что позволяет нам сделать односвязный список. Элемент списка делается с помощью ф-ии `create_node` и записываем его в список начиная с головы. Чтобы удалить предпоследний элемент, и соответственно решить задачу лабораторной была написана функция `delete_penultimate`

Функция `simple_split`:

Одномерный массив `s1` просматривается для подсчёта количества специальных символов-разделителей `m`, количество этих разделителей равно количеству полей структуры, которые предстоит заполнить. Это значит, что если в двумерном массиве символов `s2` будет храниться одно значение поля структуры на одной строке, то в двумерном массиве `s2` нужно ровно `m` строк.

Затем одномерный массив `s1` просматривается ещё раз, мы записываем значение первого поля структуры в первую строку массива `s2` до тех пор, пока не будет встречен символ-разделитель, в конец считанной первой строки добавляем нуль-терминатор.

Пропуская символ-разделитель, мы просматриваем строку далее, считывая значение второго поля структуры во вторую строку массива `s2` до следующего символа-разделителя, и так до конца строки.

Мы помещаем в каждое из полей структуры соответствующее ему значение строки двумерного массива `s2`. Важно обязательно выделить

нужное (равное длине строки массива s2) количество памяти для тех полей структуры, которые являются массивами.

Когда одни поля одной структуры заполняются, из файла считывается следующая строка s1, содержащая данные уже для другой структуры и т.д.

Функция struct\_out:

На вход поступает структура и функция просто выводит её элементы.

Функция node\_out:

На вход поступает голова, и проходя по элементам списка выводит структуру с помощью node\_out, пока не встретит NULL.

Функция create\_node:

На вход поступает массив строк, разделённый с помощью simple\_split, каждому элементу структуры присваивается строка, а так же номер. Указатель next = NULL.

Функция make\_head:

Функция создаёт голову списка, её же и возвращает. Работает только с первым элементом списка.

Функция delete\_penultimate:

На вход поступает голова списка, общее количество элементов списка и его элемент. Если голова списка равна его «хвосту», то программа пишет о том что нужно больше элементов.

Элементу списка присваивается первый элемент, дальше если элементов > 2, то доходим до предпоследнего элемента, очищаем следующий, то есть предпоследний и ему присваиваем «хвост», предпоследний элемент соответственно пропадает и на его место встаёт последний. И уменьшаем номер хвоста на 1.

## Описание переменных

Таблица 1. Описание переменных.

simple_split		
str_array	char**	Двумерный массив
i	int	Параметр цикла
j	int	Параметр цикла
k	int	Параметр цикла
m	int	Количество строк в предполагаемом двумерном массиве, параметр цикла
key	int	“флаг» успешного выделения памяти для строки двумерного массива
count	int	Количество строк двумерного массива, уже получивших значения
Main		
node	animals*	Элемент для создания списка
p	animals*	Элемент списка
ddd	Head*	Голова списка
slen	int	Длина строки
i	int	Итератор
n	int	Количество структур
count	int	Номер структуры
choice	int	Переменная-выбор

s1	char	Строка из элементов структуры
s2	char**	Разделённая строка s1
sep	char	Символ-разделитель
df	FILE*	Файл
make_head		
ddd	Head*	Голова списка
Create_node		
node	animals*	Элемент списка
node_out		
node	animals*	Элемент списка
free_nodes		
p	animals*	Элемент списка

### Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define OS_TYPE linux
#ifdef OS_TYPE
#define maxlen 256
```

```

#endif // OS_TYPE

struct animals
{
    int numb;
    char *name;
    char *kind;
    int lifespan;
    int quantity;
    float weight;//average
    float height;//average
    int numbers[3];
    struct animals *next;
};
/* define types */
typedef struct animals animals;

typedef struct Head
{
    int inc;
    animals *nose;
    animals *tail;
} Head;

/* function to split string to array by separator */
char **simple_split(char *str, int length, char
sep);//разделяем текст для записи его в структуру

/* function to print header string without data */
void print_header();

/* function to output structure fields on console */
void struct_out(animals *str0);//выводим структуру

animals *create_node(char **str, int numb);//создаём элемент
списка

Head *make_head();//создаём голову списка

void node_free(Head **head);

void node_out(Head *head);//выводим список

void delete_penultimate(Head *head, int n, animals
*p);//удаляем предпоследний элемент

```

```
void free_nodes(Head *head); //отчистка памяти
```

```
int main()
{
    animals *node = NULL, *p = NULL;
    Head *ddd = NULL;
    int slen, i, n, count, choice;
    char s1[maxlen];
    char **s2 = NULL;
    char sep;
    FILE *df;
    choice = 1;

    printf("Enter your separator symbol: \n");
    scanf("%c", &sep);

    if (sep != ';')
    {
        while (sep != ';')
        {
            puts("Enter right separator!");
            scanf("%s", &sep);
        }
    }
    df = fopen("csv.txt", "r");
    if (df != NULL)
    {
        n = 0;
        ddd = make_head(); //создаём голову списка
        while (fgets(s1, maxlen, df) != NULL) //считаем кол-
во строк
            n++;

        rewind(df); //после подсчёта строчек переводим
указатель на начало

        for (i = 0, count = 1; i < n; i++, count++)
        {
            fgets(s1, maxlen, df);
            slen = strlen(s1);
            s1[slen - 1] = '\0';
            slen = strlen(s1);
            s2 = simple_split(s1, slen, sep);
            node = create_node(s2, count);
            //строчки до for считывают строчки, разделяют
текст и создают из него элемент списка

```



```

        if (i != 0)//если элемент не первый, то
записываем его не в голову
        p -> next = node;
    else
    {
        ddd -> nose = node;
        ddd -> inc = count;
    }

    p = node;
}
}
else puts("File error!");
ddd -> tail = node;
ddd -> inc = node -> numb;

node_out(ddd);
printf("\n");
while (choice == 1)
{
    puts("wanna delete penultimate struct?\n1 - Yes\n2 -
No\n");
    scanf("%d", &choice);
    puts("\n");
    if (choice == 1)
    {
        if (n != 1)
        {
            delete_penultimate(ddd, n, p);
            n--;
            node_out(ddd);
        }
        else
        {
            puts("Need more elements!\nError!");
            break;
        }
    }
    else
        puts("Bye-bye");
}
free_nodes(ddd);
puts("Press any key when ready");
getchar();
getchar();
return 0;

```

```

}

Head *make_head()
{
    Head *ddd = NULL;
    ddd = (Head *) malloc(sizeof(Head));

    if (ddd != NULL)
    {
        ddd -> inc = 0;
        ddd -> nose = NULL;
        ddd -> tail = NULL;
    }
    return ddd;
}

animals *create_node(char **str, int numb)
{
    animals *node = NULL;
    node = (animals *) malloc(sizeof(animals));
    node -> numb = numb;
    node -> name = str[0];
    node -> kind = str[1];
    node -> lifespan = atoi(str[2]);
    node -> quantity = atoi(str[3]);
    node -> weight = atof(str[4]);
    node -> height = atof(str[5]);
    node -> numbers[0] = atoi(str[6]);
    node -> numbers[1] = atoi(str[7]);
    node -> numbers[2] = atoi(str[8]);
    node -> next = NULL;

    return node;
}

void node_out(Head *head)
{
    print_header();
    animals *node = NULL;
    node = head -> nose;

    while (node != NULL)
    {
        struct_out(node);
        node = node -> next;
    }
}

```



```

        str0 -> numb, str0 -> name, str0 -> kind, str0 ->
lifespan, str0 -> quantity, str0 -> weight, str0 -> height,
str0 -> numbers[0], str0 -> numbers[1], str0 -> numbers[2]);
}

```

```

char **simple_split(char *str, int length, char sep)
{

```

```

    char **str_array = NULL;
    int i, j, k, m;
    int key, count;
    for(j = 0, m = 0; j < length; j++)
    {
        if(str[j] == sep)
            m++;
    }

```

```

    key = 0;
    str_array = (char**)malloc((m + 1) * sizeof(char*));
    if(str_array != NULL)
    {
        for(i = 0, count = 0; i <= m; i++, count++)
        {
            str_array[i] = (char*)malloc(length *
sizeof(char));

```

```

                if(str_array[i] != NULL)
                    key=1;

                else
                {
                    key=0;
                    i=m;
                }
        }

```

```

    if(key)
    {
        k = 0;
        m = 0;

        for(j = 0; j < length; j++)
        {
            if(str[j] != sep)
                str_array[m][j - k] = str[j];

            else

```

```

        {
            str_array[m][j - k] = '\0';
            k = j + 1;
            m++;
        }
    }
}
return str_array;
}
void free_nodes(Head *head)
{
    animals *p;
    if (head -> nose == head -> tail)
        free(head -> nose);
    else
    {
        while(head -> nose -> next != NULL)
        {
            p = head -> nose -> next;
            free(head -> nose);
            head -> nose = p;
        }
    }
}

```

## Пример работы программы:

### Исходные данные:

Из файла CSV:

chicken;birds;3;4;5;6;7;8;7;

lion;cats;2;5;6;3;5;7;8;

panda;bears;2;0;9;8;5;6;7;

dog;canidae;4;3;2;5;6;8;2;

horse;equidae;3;8;5;7;7;2;5;

### Вывод программы:

```
Enter your separator symbol:
d
Enter right separator!
;
num|          fullname |      kind |lifespan |  quantity |  weight |   height | Species| Countries|Continents|
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 |         chicken |     birds |      3 |         4 | 5.000000 | 6.000000 |      7 |         8 |         7 |
| 2 |          lion |     cats |      2 |         5 | 6.000000 | 3.000000 |      5 |         7 |         8 |
| 3 |         panda |     bears |      2 |         0 | 9.000000 | 8.000000 |      5 |         6 |         7 |
| 4 |          dog |   canidae |      4 |         3 | 2.000000 | 5.000000 |      6 |         8 |         2 |
| 5 |         horse |   equidae |      3 |         8 | 5.000000 | 7.000000 |      7 |         2 |         5 |

Wanna delete penultimate struct?
1 - Yes
2 - No
1

num|          fullname |      kind |lifespan |  quantity |  weight |   height | Species| Countries|Continents|
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 |         chicken |     birds |      3 |         4 | 5.000000 | 6.000000 |      7 |         8 |         7 |
| 2 |          lion |     cats |      2 |         5 | 6.000000 | 3.000000 |      5 |         7 |         8 |
| 3 |         panda |     bears |      2 |         0 | 9.000000 | 8.000000 |      5 |         6 |         7 |
| 4 |         horse |   equidae |      3 |         8 | 5.000000 | 7.000000 |      7 |         2 |         5 |

Wanna delete penultimate struct?
1 - Yes
2 - No
2

Bye-bye

Process returned 0 (0x0)   execution time : 7.159 s
Press any key to continue.
_
```

```

|num|          fullname |      kind |lifespan | quantity | weight | height | Species| Countries|Continents|
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 |          chicken |    birds |    3 |         4 |5.000000 | 6.000000 |      7 |         8 |         7 |
| 2 |          horse |   equidae |    3 |         8 |5.000000 | 7.000000 |      7 |         2 |         5 |
Wanna delete penultimate struct?
1 - Yes
2 - No
1

|num|          fullname |      kind |lifespan | quantity | weight | height | Species| Countries|Continents|
+---+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 |          horse |   equidae |    3 |         8 |5.000000 | 7.000000 |      7 |         2 |         5 |
Wanna delete penultimate struct?
1 - Yes
2 - No
1

Error!
Program need more structures!
Bye-bye!

Process returned -1073741819 (0xC0000005)   execution time : 7.203 s
Press any key to continue.

```

### Выводы:

При выполнении лабораторной работы были получены практические навыки в написании программы на языке Си для работы с односвязным списком.