

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: Линейные односвязные списки

Студент гр. 9305

Салауров Е. М.

Преподаватель

Перязева Ю. В.

Санкт-Петербург

2020

Оглавление

Введение.....	3
Задание	3
Постановка задачи и описание решения	3
Описание переменных	5
Контрольные примеры	10
Текст программы	10
Пример работы программы	22
Заключение.....	23

Введение

Задание

С использованием структуры, созданной при выполнении лабораторной работы №9 (по выбранной предметной области), создать односвязный линейный список и выполнить задание в соответствии с вариантом.

Вариант задания:

Разработать подалгоритм и написать функцию, вставляющую в односвязный список получаемые данные перед заданным по номеру элементом. Номер элемента задается с конца списка. При недостаточном количестве элементов в списке данные вставить в начало списка.

Постановка задачи и описание решения

Для решения этой задачи изначально нужно объявить структуру, я сделал это так:

```
typedef struct ZNAK {  
    char NAME[MAXLEN];  
    char NIK[MAXLEN];  
    int DATE[3];  
    int chislopole1;  
    int chislopole2;  
    float chislopole3;  
    float chislopole4;  
} group;
```

Структура была взята из прошлой лабораторной работы.

Также объявил структуру для списка:

```
struct node{  
    group *data;  
    struct node *next;  
};
```

Здесь data это поле, которое содержит информацию, которая описывается в структуре ZNAK. Далее в программе описываются функции.

Программа начинает свою работу с проверки на корректное открытие файла для чтения, после чего программа считает сколько строчек она должна взять из этого файла таким циклом:

```
while((fgets(s1,MAXLEN,df))!=NULL) n++;
```

Теперь программа формирует массив, в который записывает информацию из файла с помощью функции `simple_split()`. Эта функция считывает строку из файла и распределяет ее по ячейкам массива. И тут же программа делает проверку на корректное заполнение этого массива, после чего она заполняет структуру распределяя ячейки массива по полям структуры с помощью функции `struct_fill()`. После создания списка из файла, программа выводит список в виде таблицы.

Далее пользователю выводится меню из трех пунктов. Пользователю предлагается создать новый элемент списка, вывести таблицу еще раз для проверки ввода нового элемента списка и выйти из программы.

Подробнее опишу пункт меню с добавлением нового элемента списка. После выбора этого пункта пользователю предлагается ввести номер, под которым будет расположена новая структура в списке. После этого создается указатель типа `group` и под него выделяется память. Он заполняется с помощью функции `new_struct()`. Эта функция запрашивает данные у пользователя в каждую ячейку инициализированной структуры. После чего программа с помощью функции `insert_after()` вставляет структуру в список.

Описание структуры:

Структура ZNAK:

Название переменной	Тип переменной	Назначение
NAME	char	Поле с именем
NIK	char	Поле со знаком зоди
DATE	int	Массив содержащий дату рождения
chislopole1	int	Любимое число пользователя
Chislopole2	int	Счастливое число пользователя
Chislopole3	float	Процент удачи пользователя
Chislopole4	float	

Структура node:

Название переменной	Тип переменной	Назначение
data	group	Информационное поле
next	node	Указатель на следующую структуру в списке

Описание функций:

Функция `main()`:

Описание:

Точка входа в программу. Отвечает за открытие файла, содержащего данные для последующей работы.

Прототип:

```
int main()
```

Пример вызова:

```
main()
```

Описание переменных:

Название переменной	Тип переменной	Назначение
ch	group	Массив для чтения данных из файла
slen	int	Переменная для хранения длины массива
i		Переменная для работы оператора фор
n		Переменная хранящая количество строчек в файле
s2	char	Массив для заполнения структуры
s1	char	Так же массив для чтения данных из файла
sep	char	Переменная ограничитель для прохода по файлу
df	FILE	Файл
head	node	Указатель на голову списка
temp	node	Указатель для создания элемента списка
p	node	Указатель для создания элемента списка

Возвращает значение: 0, если работа программы завершена успешно.

Функция `simple_split()`:

Описание:

Функция разделения строки по заданному разделителю.

Каждая строка файла разделяется на элементы промежуточного массива строк s2 по разделителям с помощью функции? и в зависимости от типа поля элемента массива структур выполняется преобразование элемента массива строк в поле отдельной структуры.

Прототип:

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

Пример вызова:

```
s2=simple_split(s1,slen,sep);
```

Описание переменных:

Название переменной	Тип переменной	Назначение
str_array	char	Массив в который копируются данные из файла
i	int	Переменная для работы с циклом
j	int	Переменная для работы с циклом
k	int	Переменная для работы с массивом
m	int	Переменная для работы с массивом
key	int	Переменная индикатор
count	int	Переменная для очистки массива

Возвращаемое значение: массив строк.

Функция ClearStringArray():

Описание:

Функция очистки памяти для динамического массива строк.

Прототип:

```
void ClearStringArray(char **str, int n)
```

Пример вызова:

```
ClearStringArray(str_array,count);
```

Описание переменных:

Название переменной	Тип переменной	Назначение
i	int	Переменная для прохода по массиву
str	char**	Массив строк

Функция new_struct():

Название переменной	Тип переменной	Назначение
str0	group	Структура для заполнения новыми данными

Функция struct_fill():

Описание:

Функция заполнения структуры данными из файла. В массив строк вводятся полученные из simple_split данные.

Прототип:

```
cars *struct_fill(char **str)
```

Пример вызова:

```
ch[i]=struct_fill(s2);
```

Описание переменных:

Название переменной	Тип переменной	Назначение
str0	group	Структура для распределения в ней данных из массива

Функция create_node():

Описание:

Функция создает новый узел связного списка. Отводится память под новую

запись, устанавливаются значения полей, переданные в аргументах, ссылка на следующий элемент устанавливается в NULL

Прототип:

```
node *create_node(cars *data);
```

Пример вызова:

```
temp=create_node(ch[0]);
```

Описание переменных:

Название переменной	Тип переменной	Назначение
temp	node	Структура для вставки ее в список

Функция print_list():

Описание:

Функция вывода списка. Пока указатель на голову не NULL, выводит данные одной строки и переходит к следующей

Прототип:

```
void print_list(node *head)
```

Пример вызова:

```
print_list(head);
```

Описание переменных:

Название переменной	Тип переменной	Назначение
p	node	Указатель на голову списка

Функция insert_after():

Описание:

Прототип:

```
void insert_after(node *head, int index, group *value)
```

Пример вызова:

```
insert_after(head, a, str0)
```

Описание переменных:

Название переменной	Тип переменной	Назначение
i	int	Переменная для ориентирования в списке
p	node	Указатель на данные из головы массива
temp	node	Структура которую вставляют в список

Контрольные примеры

Пример 1:

Пользователь выбирает 5 индекс для новой структуры

С клавиатуры вводится поля структуры:

EVGENII S; RIBA; 2000; 12; 12; 123; 13; 0.043; 0.54

Пример 2:

Пользователь выбирает 9 индекс для новой структуры

С клавиатуры вводится поля структуры:

EVGENII CHOKOLEV; VODALEI; 2000; 23; 12; 2; 666; 0.043; 0.54

Алгоритм на естественном языке

Main

- 1) проверяет на правильность открытие файла
- 2) Если файл открыт успешно, то сосчитать количество строк в нём
- 3) Пока $i=0$ не дойдет до последней строки:
 - Вычисляем длину строки из файла
 - Вызываем функцию для сортировки массива строк по символуразделителю `simple_split`
 - Если массив строк $s2 \neq \text{NULL}$, то заполнение структуры функцией `struct_fill`
- 4) Создаем голову списка в нулевом элементе функцией `create_node`
 - Пока $i=1$ не дойдет до последней строки:
 - Создаем остальной список `create_node`
 - Удаляем последний элемент функцией `delete_node`
 - Функцией `print_list` выводим результат
- 5) Запуск функции `menu()`

menu

- 1) Следующие пункты выполняются пока пользователь не введет 0
- 2) выводиться список возможных действий.
 - Если пользователь вводит 1, то программа запрашивает у него данные для ввода в таблицу нового узла, то есть индекс нового узла, и если такой индекс существует в списке, то программа выполняет вставку.
 - Если такого индекса в структуре нет, то программа с помощью функции `add` вставляет структуру на первое место.
 - Если пользователь вводит 2, то программа выводит список на экран.

simple_split

- 1) Считаем количество строк

2) Выделяем память для массива строк

Если память выделилась удачно, то

Производится считывание каждой строки, пока не дойдем до последней

Ищем в каждой строке символ-разделитель и пропускаем его

После очищаем память функцией `ClearStringArray`

3) Возвращаем полученный массив строк

ClearStringArray

1) Пока $i=0$ не дойдет до последней строки, освобождаем память каждой строки из массива строк и присваиваем `NULL`

struct_fill

1) Выделяем память для массива структур

Если память выделилась удачно, то

Каждой ячейке присваивается значение из массива строк, полученного ранее

2) Возвращаем структуру

create_node

1) Создаем узел списка

2) Выделяем под него память и вносим данные из структуры

3) Возвращаем ссылку на узел списка

print_header

1) Вывод шапки структуры

Текст программы

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <ctype.h>
```

```
#define MAXLEN 256
```

```
typedef struct ZNAK {  
    char NAME[MAXLEN];  
    char NIK[MAXLEN];  
    int DATE[3];  
    int chislopole1;  
    int chislopole2;  
    float chislopole3;  
    float chislopole4;  
} group;
```

```
struct node{  
  
    group *data;  
    struct node *next;  
};  
typedef struct node node;
```

```
char **simple_split(char *str, int length, char sep);
```

```
void ClearStringArray(char **str, int n);
```

```
group *new_struct();
```

```
group *struct_fill(char **str);
```

```
node *create_node(group *data);
```

```
void print_list(node **head);
```

```
void add(node **head, group *data);
```

```
void add_last(node **tail, group *data);
```

```
void insert_after(node *head, int index, group *value);
```

```
int main(){
```

```
    group **ch=NULL;
```

```
    int slen,i,n,count;
```

```
    char **s2=NULL;
```

```
    char s1[MAXLEN];
```

```
    char sep;
```

```
    FILE *df;
```

```
    node *head = NULL;
```

```
    node *temp, *p;
```

```
    sep=';';
```

```
    df=fopen("struct-data-03.csv", "r");
```

```
    if(df!=NULL){
```

```
        n=0;
```

```
        while((fgets(s1,MAXLEN,df))!=NULL) n++;
```

```
        rewind(df);
```

```
        ch=(group**)malloc(n*sizeof(group*));
```

```
        puts("Initial array:");
```

```
        if(ch!=NULL){
```

```
            for(i=0,count=0;i<n;i++,count++){
```

```
                fgets(s1,MAXLEN,df);
```

```
                slen=strlen(s1);
```

```
                s1[slen-1]='\0';
```

```
                slen=strlen(s1);
```

```

s2=simple_split(s1,slen,sep);
if(s2!=NULL){
    ch[i]=struct_fill(s2);
}
else puts("Error at data reading!");
}
temp=create_node(ch[0]);
head = temp;
for(i = 1; i < 12; i++){
    p = create_node(ch[i]);
    temp -> next = p;
    temp = p;
}
print_list(&head);
}
i=9;
while(i!=0){
    printf("Menu:\n");
    printf("1 - Add stuct\n");
    printf("2 - Print struct\n");
    printf("0 - Exit\n");
    scanf("%d", &i);
    getchar();
    if(i == 1){
        int a = 0;
        printf("\n");
        printf("Index struct:\n");
        scanf("%d", &a);
        a--;
    }
}

```

```

        printf("bingo%d\n", a);

        group *str0=NULL;

        str0=(group*)malloc(sizeof(group));

        str0 = new_struct();

        insert_after(head, a, str0);

    }

    if(i == 2){

        print_list(&head);

    }

}

else puts("File not found!");

}

```

```

node *create_node(group *data){

    node *temp;

    temp = (node *)malloc(sizeof(node));

    temp -> data = data;

    temp -> next = NULL;

    return temp;

}

```

```

void print_list(node **head){

    node *p;

    p = *head;

    while(p != NULL){

        printf("|%20s |%10s|%2d|%2d|%2d|%5d|%5d|%10f|%10f|\n",

            p -> data ->NAME,

            p -> data ->NIK,

            p -> data ->DATE[0],

```



```

        p -> data -> DATE[1],
        p -> data -> DATE[2],
        p -> data -> chislopole1,
        p -> data -> chislopole2,
        p -> data -> chislopole3,
        p -> data -> chislopole4);
    p = p->next;
}
}

```

```

void add(node **head, group *data){
    node *temp = (node *)malloc(sizeof(node));
    temp -> data = (group**)malloc(sizeof(group*));
    temp -> next = *head;
    temp -> data = data;
    *head = temp;
}

```

```

void add_last(node **tail, group *data){
    node *temp = (node *)malloc(sizeof(node));
    temp -> next = NULL;
    temp -> data = data;
    (*tail)->next = temp;
    *tail = temp;
}

```

```

void insert_after(node *head, int index, group *value){
    int i;
    node *p = head;
    node *temp;

```

```

//p = (node*)malloc(sizeof(node));

temp = (node*)malloc(sizeof(node));

i = 0;

printf("bingo%d\n", index);

while (i < index - 1){printf("bingo\n");

    p = p->next;

    i++;

}

temp = create_node(value);

temp->next = p->next;

p->next = temp;

}

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++;
        }
    }
}
else{
    ClearStringArray(str_array,count);
}
}
return str_array;
}

```

```

void ClearStringArray(char **str, int n){
    int i;

    for(i=0;i<n;i++){
        free(str[i]);
        str[i]=NULL;
    }
    free(str);
    str=NULL;
}

```

```
}
```

```
group *struct_fill(char **str){  
    group *str0=NULL;  
  
    str0=(group*)malloc(sizeof(group));  
    if(str0!=NULL){  
        strcpy(str0 -> NAME, str[0]);  
        //temp -> data -> NAME = s2[0];  
        strcpy(str0 -> NIK, str[1]);  
        //temp -> data -> NIK = s2[1];  
        str0-> DATE[0] = atoi(str[2]);  
        str0 -> DATE[1] = atoi(str[3]);  
        str0 -> DATE[2] = atoi(str[4]);  
        str0 -> chislopole1 = atoi(str[5]);  
        str0 -> chislopole2 = atoi(str[6]);  
        str0 -> chislopole3 = atof(str[7]);  
        str0 -> chislopole4 = atof(str[8]);  
    }  
    return str0;  
}
```

```
group *new_struct(){  
  
    group *str0=NULL;  
  
    str0=(group*)malloc(sizeof(group));  
    if(str0!=NULL){  
        getchar();  

```

```

    puts("Enter name:");
    fgets((*str0).NAME,MAXLEN,stdin);
    puts("Enter nik:");
    fgets((*str0).NIK,MAXLEN,stdin);
    puts("Enter day:");
    scanf("%d",&(*str0).DATE[0]);
    puts("Enter mounth:");
    scanf("%d",&(*str0).DATE[1]);
    puts("Enter year:");
    scanf("%d",&(*str0).DATE[2]);
    puts("Enter like chislo:");
    scanf("%d",&(*str0).chislopole1);
    puts("Enter like chislo:");
    scanf("%d",&(*str0).chislopole2);
    puts("Enter like chislo:");
    scanf("%f",&(*str0).chislopole3);
    puts("Enter like chislo:");
    scanf("%f",&(*str0).chislopole4);
    str0->NAME[strlen(str0->NAME)-1]='\0';
    str0->NIK[strlen(str0->NIK)-1]='\0';
}
return str0;
}

```

Пример работы программы

```
Initial array:
|      Last James | vodolei|2000| 3| 2|   1| 12| 0.120000| 0.234000|
|      Mae Vanessa | lev|1990| 2| 1| 123| 123| 0.220000| 0.124000|
|      Chang Jackie | oven|1999| 1| 4| 213| 34| 0.110000| 0.414000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      McCartney Pol | ckorpion|1999| 8| 2|   3| 1| 0.880000| 0.284000|
|      Howston Witney | vodolei|1987| 6| 1|   1| 3| 0.660000| 0.164000|
|      Stone Sharonne | riba|2001| 5| 3| 13| 2| 0.550000| 0.354000|
|      McCartney Pol | lev|1990| 2| 1| 23| 342| 0.220000| 0.124000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|

Menu:
1 - Add stuct
2 - Print struct
0 - Exit
1

Index struct:
5
Enter name:
Евгений Салауров
Enter nik:
ZzanZzan
Enter year:
2000
Enter mounth:
12
Enter day:
12
Enter like chislo:
3
Enter like chislo:
4
Enter like chislo:
5
Enter like chislo:
6
Menu:
1 - Add stuct
2 - Print struct
0 - Exit
2
|      Last James | vodolei|2000| 3| 2|   1| 12| 0.120000| 0.234000|
|      Mae Vanessa | lev|1990| 2| 1| 123| 123| 0.220000| 0.124000|
|      Chang Jackie | oven|1999| 1| 4| 213| 34| 0.110000| 0.414000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      Евгений Салауров | ZzanZzan|2000|12|12|   3| 4| 5.000000| 6.000000|
|      McCartney Pol | ckorpion|1999| 8| 2|   3| 1| 0.880000| 0.284000|
|      Howston Witney | vodolei|1987| 6| 1|   1| 3| 0.660000| 0.164000|
|      Stone Sharonne | riba|2001| 5| 3| 13| 2| 0.550000| 0.354000|
|      McCartney Pol | lev|1990| 2| 1| 23| 342| 0.220000| 0.124000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|
|      Stone Sharonne | riba|2001| 5| 3| 234| 3| 0.550000| 0.354000|

Menu:
1 - Add stuct
2 - Print struct
0 - Exit
```

Заключение

Выводы:

При выполнении лабораторной работы были получены практические навыки в разработке алгоритма и написании программы на языке Си. Были получены основные знания о синтаксисе языка Си, в частности, о программировании задач со структурами, а также правилах написания кода на языке Си.