

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ВТ**

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: «Разработка электронной картотеки»

Студент гр. 9305

Архипов Н. Д.

Преподаватель

Перязева Ю. В.

Санкт-Петербург

2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Архипов Н. Д.

Группа 9305

Тема работы: разработка электронной картотеки

Исходные данные:

файл разрешения «.csv»

Содержание пояснительной записи:

«Содержание», «Введение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записи:

Не менее 20 страниц.

Дата выдачи задания: 01.04.2020

Дата сдачи реферата: 08.06.2020

Дата защиты реферата: 08.06.2020

Студент

Архипов Н. Д.

Преподаватель

Перязева Ю. В.

АННОТАЦИЯ

Картотека реализована на линейном односвязном списке. Записи картотеки хранят данные о совершенных транзакциях. Пользователю доступны следующие функции: сортировка картотеки по дате и по сумме; поиск записей по диапазону дат; удаление и редактирование транзакции по индексу; добавление новой транзакции.

Программный код представлен на [github](#), ссылка приведена в приложении B.

SUMMARY

The card library is implemented on a linear, single-linked list. Card library records store data about completed transactions. The following functions are available to the user: sorting the card file by date and amount; searching for records by date range; deleting and editing a transaction by index; adding a new transaction.

The program code is provided on [github](#), with a link below.

СОДЕРЖАНИЕ

Введение	4
1. Картотека	6
1.1. Предметная область, описание структуры	6
1.2. Главое меню	6
2. Программная реализация	7
2.1. Описание решения	7
2.2. Описания структур	7
2.3. Описания функций	9
2.4. Пример работы программы	15
Заключение	17
Список использованных источников	18
Приложение А. Схема вызова функций	19
Приложение В. Ссылка на репозиторий	20

ВВЕДЕНИЕ

Цель работы

Написание программы, осуществляющей работу с электронной картотекой в соответствии с выбранной предметной областью.

Задачи

Изучить поставленную задачу.

Спланировать путь решения, разработать алгоритм.

Написать на языке Си работоспособную программу, удовлетворяющую поставленной цели.

Сделать вывод о полученном результате.

1. КАРТОТЕКА

1.1. Предметная область, описание структуры

Картотека основана на структурах, хранящих данные о транзакциях. Это дата, сумма, тип (доход или расход), наименование банка и наименование категории дохода или расхода.

Описание структуры:

```
typedef struct transaction_card
{
    float sum;          // sum in rubles
    char* category;    // category of transaction
    char* card_name;   // name of payment card or bank
    unsigned type:1;    // 0 -- outcome; 1 -- income
    struct _date *date; // date of transaction
} transaction;
```

1.2. Главное меню

- 0) Инфо
- 1) Добавление
- 2) Редактирование
- 3) Удаление
- 4) Вывод таблицы
- 5) Поиск
- 6) Сортировка
- 7) Выход

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. Описание решения

Картотека основана на односвязном списке.

Вывод производится только в порядке от первого к последнему элементу.

Редактирование и удаление осуществляется с любым элементом по индексу вне зависимости от его положения.

Сортировка реализована только по дате и по сумме.

Поиск работает для диапазона дат.

2.2. Описания структур

Ввиду простоты и наглядности программного кода структуры представлены не в табличном виде, а в форме демонстрации кода с последующим описанием.

2.2.1. Структура элемента списка.

Программный код:

```
typedef struct list_element
{
    int index;
    struct transaction_card *data;
    struct list_element *next;
} l_elem;
```

Описание:

Хранит индекс (int), указатель на информационное поле (struct transaction_card*), указатель на следующий элемент (struct list_element*).

2.2.2. Структура головы списка.

Программный код:

```
typedef struct
{
    int count; // number of elements in list
    struct list_element *first;
```

```
    struct list_element *last;  
} l_head;
```

Описание:

Хранит счетчик количества элементов, указатели на первый и последний элемент списка.

2.2.3. Структура информации о транзакции

Программный код:

```
typedef struct transaction_card  
{  
    float sum;          // sum in rubles  
    char* category;   // category of transaction  
    char* card_name;  // name of payment card or bank  
    unsigned type:1;   // 0 -- outcome; 1 -- income  
    struct _date *date; // date of transaction  
} transaction;
```

Описание:

Структура, хранящая сумму sum (float), категорию транзакции category (char*), наименование банка card_name (char*), тип операции type (unsigned ...:1), указатель на структуру даты date (struct _date *date).

2.2.4. Структура даты

Программный код:

```
typedef struct _date  
{  
    unsigned year:12; // 4095 is max  
    unsigned month:4; // 12 is max  
    unsigned day:5;   // 31 is max  
} date;
```

Описание:

Хранит данные о дне, месяце и году, представленные битовым полем.

2.3. Описания функций

2.3.1. Функция **split**

Описание:

Функция разделяет строки по заданному разделителю и возвращает массив строк. Память под массив выделяется по предварительному подсчету количества разделителей. Память под элементы выделяется по аргументу **length**.

Прототип:

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

Пример вызова:

```
tmp_str_array = split(tmp_str, maxlen, SEP);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	str	char*	Исходная строка
Формальный аргумент	length	int	Длина строки
Формальный аргумент	SEP	char	Символ-разделитель
Локальная	sep_array	char**	Массив строк
Локальная	i	int	Счётчик цикла
Локальная	j	int	Счётчик цикла
Локальная	k	int	Счётчик цикла
Локальная	m	int	Счётчик строки в массиве строк
Локальная	key	int	Флаг выделения памяти
Локальная	count	int	Счетчик количества элементов, используется для очистки памяти при ошибке

Возвращаемое значение:

Массив строк

2.3.2. Функция `clear_str_array`

Описание:

Функция очистки памяти для массива строк.

Прототип:

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

Пример вызова:

```
clear_str_array(sep_str,count);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	str	char**	Массив строк
Формальный аргумент	n	int	Количество строк
Локальная	i	int	Счётчик цикла

2.3.3. Функция `struct_create`

Описание:

Функция по аргументу (массиву строк) заполняет информационные поля структуры. Для преобразования к типам *int* и *float* используются функции *atoi* и *atof*. Память под структуру выделяется внутри функции, возвращается указатель.

Прототип:

```
transaction *struct_create(char **str)
```

Пример вызова:

```
tmp_lt = struct_create(tmp_str_array);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	str	char**	Массив строк
Локальная	lt	transaction*	Указатель на созданную

Возвращаемое значение:
Указатель на созданную структуру.

2.3.4. Функция **struct_clear**

Описание:

Функция очищает память для структуры.

Прототип:

```
void struct_clear(transaction *t)
```

Пример вызова:

```
struct_clear(t);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	t	transaction*	Указатель на структуру для очистки памяти

Возвращаемое значение:

Заполненная структура.

2.3.5. Функция **struct_print**

Описание:

Функция печатает строку таблицы с содержимым полей структуры.

Прототип:

```
void struct_print(transaction *t)
```

Пример вызова:

```
struct_print(tmp->data);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
----------------	----------------	-----	------------

Формальный аргумент	t	transaction*	Структура к печати
---------------------	---	--------------	--------------------

2.3.6. Простейшие функции

Описание:

Функция print_header печатает заголовок таблицы, print_line – разделительную линию таблицы.

Прототипы:

```
void print_header()
```

```
void print_line()
```

Пример вызова:

```
print_header();
```

```
print_line();
```

2.3.7. Функция list_init

Описание:

Инициализирует ЛОС: выделяет память под первый элемент списка, устанавливает указатель на следующий элемент в NULL, сохраняет указатель на структуру с информационными полями; устанавливает счетчик головы в единицу, указатели на первый и последний элемент указателем на этот элемент.

Прототип:

```
l_elem *list_init(l_head *hd, transaction *lt)
```

Пример вызова:

```
p_node = list_init(p_head, tmp_lt);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	hd	l_head*	Указатель на голову
Формальный аргумент	lt	transaction*	Указатель на структуру

Локальная	lst	l_elem*	Указатель на новый (первый) элемент списка.
-----------	-----	---------	---

Возвращаемое значение:

Указатель на новый (первый) элемент списка.

2.3.8. Функция **list_add**

Описание:

Функция добавляет в конец новый элемент списка, принимая указатель на голову, указатель на текущий элемент и указатель на структуру. В новый элемент записывается указатель на структуру, указателю на следующий элемент присваивается NULL. В текущем (т.е. прошлом) элементе указатель на следующий элемент устанавливается в новый элемент. В голове обновляется указатель на последний элемент, счетчик увеличивается на 1.

Прототип:

```
l_elem *list_add(l_head *hd, l_elem *c_lst, transaction *lt)
```

Пример вызова:

```
p_node = list_add(p_head, p_node, tmp_lt);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	hd	l_head*	Указатель на голову
Формальный аргумент	c_lst	l_elem*	Указатель на текущий элемент
Формальный аргумент	lt	transaction*	Указатель на структуру
Локальная	n_lst	l_elem*	Указатель на добавленный элемент

Возвращаемое значение:

Указатель на добавленный элемент.

2.3.9. Функция **list_print**

Описание:

Функция принимает указатель на голову и печатает все элементы списка, обращаясь к функции struct_print. Работает только с линейными списками, т.к. печать выполняется в цикле ПОКА указатель на следующий элемент не будет NULL.

Прототип:

```
void list_print(l_head *hd)
```

Пример вызова:

```
list_print(p_head);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	hd	l_head*	Указатель на голову
Локальная	tmp	l_elem*	Указатель на элемент для прохода по списку

2.3.10. Функция **list_swap**

Описание:

Функция обмена элементов местами по двум принятым указателям. Фактически обменивает не положение элементов в списке, а указатели на структуру с информационными полями, *поэтому универсальна и работает с любым видом списка*. Используется при сортировке.

Прототип:

```
void list_swap(l_elem *n1, list_elem *n2)
```

Пример вызова:

```
list_swap(node_1, node_2);
```

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	n1	l_elem*	Указатель на элемент для обмена
Формальный	n2	l_elem*	Указатель на элемент для

аргумент			обмена
Локальная	tmp	transaction*	Временный указатель на структуру, буфер обмена

2.4. Пример работы программы

```
This program is usable for budget calculations.
About work: programming course work of second semester
About author: Arkhipov Nikita, group 9305
MENU:
0 -- info
1 -- add a card
2 -- redact a card
3 -- delete a card
4 -- print table
5 -- search a card
6 -- sort by param
7 -- exit
Your num: 6
0 -- sort by date
1 -- sort by price
Your num: 0
MENU:
0 -- info
1 -- add a card
2 -- redact a card
3 -- delete a card
4 -- print table
5 -- search a card
6 -- sort by param
7 -- exit
Your num: 4
+-----+
| number |      date |      sum |   type | card name |      category |
+-----+
|    1 | 1. 1.2020 | 35.00 R | outcome | Sberbank | mobile connection |
|    2 | 1. 1.2020 | 2000.00 R | income | Sberbank | main job |
|    3 | 2. 1.2020 | 159.33 R | outcome | Sberbank | girlfriend |
|    4 | 2. 1.2020 | 4545.00 R | income | Qiwi Bank | side job |
|    5 | 2. 1.2020 | 1100.00 R | outcome | Qiwi Bank | debt payment |
|    6 | 2. 1.2020 | 1074.00 R | outcome | Qiwi Bank | debt payment |
|    7 | 2. 1.2020 | 318.67 R | outcome | Sberbank | smoking |
|    8 | 2. 1.2020 | 250.00 R | income | Sberbank | side job |
|    9 | 3. 1.2020 | 135.00 R | outcome | Qiwi Bank | present |
|   10 | 3. 1.2020 | 59.94 R | outcome | Qiwi Bank | girlfriend |
|   11 | 3. 1.2020 | 1010.00 R | outcome | Qiwi Bank | debt payment |
|   12 | 3. 1.2020 | 155.70 R | outcome | Sberbank | food |
|   13 | 3. 1.2020 | 39.96 R | outcome | Qiwi Bank | food |
|   14 | 4. 1.2020 | 199.99 R | outcome | Qiwi Bank | present |
|   15 | 4. 1.2020 | 134.10 R | outcome | Qiwi Bank | girlfriend |
|   16 | 4. 1.2020 | 312.20 R | outcome | Qiwi Bank | smoking |
|   17 | 4. 1.2020 | 60.00 R | outcome | Qiwi Bank | girlfriend |
|   18 | 4. 1.2020 | 83.40 R | outcome | Qiwi Bank | food |
|   19 | 4. 1.2020 | 25.00 R | outcome | Qiwi Bank | mobile connection |
```

```

5 -- search a card
6 -- sort by param
7 -- exit
Your num: 5
This is searching between two dates, format: dd.mm.yyyy
Print min (older) date: 10.01.2020
Print max (newest) date: 12.01.2020
| 45|10. 1.2020|    206.54 R| outcome| Sberbank|      smoking|
| 46|12. 1.2020|    211.25 R| outcome| Qiwi Bank|      smoking|
| 47|12. 1.2020|     62.00 R| outcome| Sberbank|      food|
| 48|12. 1.2020|     38.00 R| income| Qiwi Bank|      main job|
| 49|12. 1.2020|    362.00 R| income| Qiwi Bank|      loan|
| 50|12. 1.2020|    113.75 R| outcome| Qiwi Bank|      girlfriend|
MENU:

```

```

2 -- redact a card
3 -- delete a card
4 -- print table
5 -- search a card
6 -- sort by param
7 -- exit
Your num: 3
Write a number of transaction to delete: 45
MENU:
0 -- info
1 -- add a card
2 -- redact a card
3 -- delete a card
4 -- print table
5 -- search a card
6 -- sort by param
7 -- exit
Your num: 5
This is searching between two dates, format: dd.mm.yyyy
Print min (older) date: 10.01.2020
Print max (newest) date: 12.01.2020
| 45|12. 1.2020|    211.25 R| outcome| Qiwi Bank|      smoking|
| 46|12. 1.2020|     62.00 R| outcome| Sberbank|      food|
| 47|12. 1.2020|     38.00 R| income| Qiwi Bank|      main job|
| 48|12. 1.2020|    362.00 R| income| Qiwi Bank|      loan|
| 49|12. 1.2020|    113.75 R| outcome| Qiwi Bank|      girlfriend|
MENU:
0 -- info
1 -- add a card
2 -- redact a card
3 -- delete a card
4 -- print table
5 -- search a card
6 -- sort by param
7 -- exit
Your num: 7
Data saving...
Process returned 0 (0x0)  execution time : 306.146 s
Press any key to continue.

```

ЗАКЛЮЧЕНИЕ

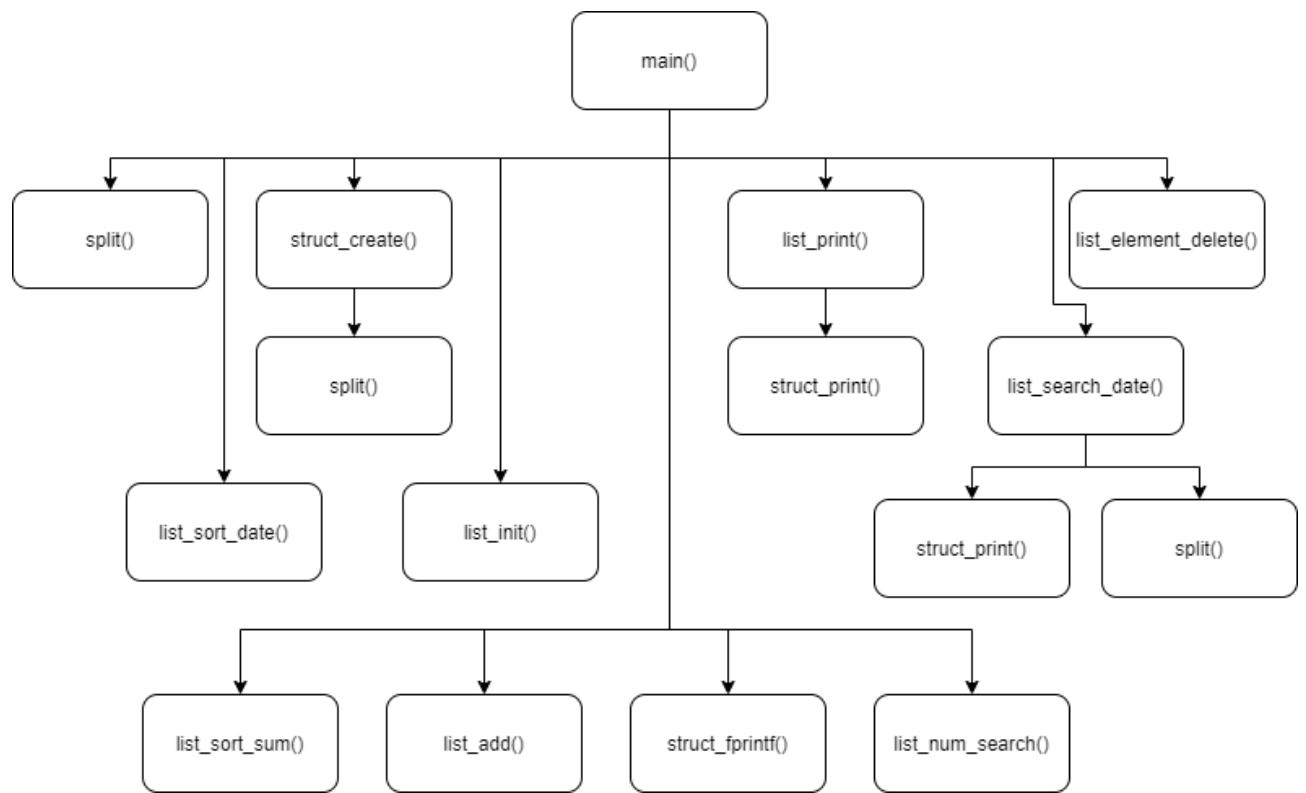
В ходе выполнения лабораторной работы была реализована электронная картотека транзакций по банковским картам одного человека, которая при дальнейшем совершенствовании может стать полноценным инструментом ведения личного бюджета. При этом были решены задачи планирования решения, построения алгоритма и его последующей реализации на языке Си, а также практические навыки отладки написанного программного кода и анализа произведенной работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Примеры программного кода / сост.: И.А. Хахаев, СПбГЭТУ «ЛЭТИ».
2. Информационный ресурс с описанием функций языка Си // prog-cpp. URL: <https://prog-cpp.ru> (дата обращения: 20.05.2020).
3. Информационный ресурс с описанием функций языка Си (в том числе) // all-ht. URL: <http://all-ht.ru/inf/prog/c> (дата обращения: 11.05.2020).

ПРИЛОЖЕНИЕ А

СХЕМЫ ВЫЗОВА ФУНКЦИЙ



ПРИЛОЖЕНИЕ В
ССЫЛКА НА GITHUB

https://github.com/NiArhETU9305/9305_ARKHIPOV_course