

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по курсовой работе
по дисциплине «Программирование»
Тема: Генерация отчетов

Студент гр. 7382

Глазунов С.А.

Преподаватель

Кринкин К.В.

Санкт-Петербург

2018

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Глазунов С.А.

Группа 7382

Тема работы : Генерация отчетов

Исходные данные: В качестве основы для курсовой работы используется код лабораторной работы No4.

Содержание пояснительной записки: «Введение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 5 страниц.

Дата выдачи задания: 28.11.2017

Дата сдачи реферата: 23.12.2017

Дата защиты реферата: 23.12.2017

Студент

Глазунов С.А.

Преподаватель

Кринкин К.В.

АННОТАЦИЯ

Необходимо, имея код лабораторной работы No4, реализовать алгоритм, который делает следующие преобразования со списком:

- Отсортировать список по невозрастанию по полю year в этом списке;
- Менять местами элементы не затрагивая поля, кроме тех, что указывают наследующие и предыдущие элементы;

Пишется две функции, которые производят все эти преобразования и возвращают головной элемент списка.

ВВЕДЕНИЕ

Необходимо, имея код лабораторной работы No4, реализовать алгоритм, который делает следующие преобразования со списком:

- Отсортировать список по невозрастанию по полю year в этом списке;
- Менять элементы не трогая поля, кроме тех, которые указывают на следующий и на предыдущий элемент;

РЕАЛИЗАЦИЯ ФУНКЦИЙ

На вход функции, которая названа `l1st_bubble_sort ()` подается адрес первого элемента списка, который условно назван “head”. Внутри `l1st_bubble_sort ()` была использована функция `swap()`, на вход которой подается 2 элемента (подразумевается, что первый аргумент является левым элементом списка второго аргумента) и адрес “head”, а также функция `count()`, написанная ранее для лабораторной работы No4, чтобы подсчитать количество элементов списка.

Сортировка списка

Изначально нужно определить, сколько элементов будет в списке. Для этого вызовем функцию `count()`. Также потребуется 2 новых указателя, которые изначально будут указывать на “head” и “head->next” соответственно.

Реализация:

Создаем переменную `len`, которая будет хранить количество элементов списка, и присваиваем ей значение: `int k=count(*head);` Далее инициализируем другие переменные, которые будут являться указателями на “head” и “head->next”: `Struct MusicalComposition* cur1*cur2;` Далее начинается первый цикл: `for (int j=0; j<len-1; j++);` Переменная `j<len-1`, потому что указатель `cur2` из пункта указывает на следующий элемент первого. Когда бы наступила последняя итерация цикла то `cur2` указывал уже на несуществующий элемент списка, что могло бы повлечь за собой ошибку сегментации. Далее во внутри 1-ого цикла есть еще и второй цикл, который при каждой итерации первого цикла проходит по `len-1-j` элементов списка. Уже во 2-ом цикле происходит сравнение поля `year` двух элементов списка. Если `cur1->year` меньше `cur2->year`, то вызывается функция `swap`, которая меняет элементы местами (См. Рис. 1).

Теперь рассмотрим функцию `swar`. В функции `swar` надо рассмотреть 3 случая: 1. Меняется “head” со следующим элементом; 2. Меняется два элемента списка, ни один из которых не является головой или хвостом списка; 3. Меняется хвост и предыдущий элемент. Для каждого случая нужен свой блок кода, состоящий из двух условий, поэтому в каждом блоке есть команда `return` – это сделано для того, чтобы Функция не проверяла лишние условия, когда заведомо следующие условия ложны (См. Рис. 2).

Принцип работы функции `swar`:

Так как мы работаем в линейном двусвязном списке, то для того, чтобы поменять два элемента местами в общем случае надо поменять 6 связей в сумме. По две связи у элементов, которые должны поменяться местами и по одной связи у элементов, которые до `cur1` и после `cur2`. В `cur1->prev` надо поменять указатель `next` с `cur1` на `cur2`. В `cur2->next` надо поменять указатель `prev` с `cur2` на `cur1`. В `cur1` надо поменять `prev` с `cur1->prev` на `cur2`, `next` поменять с `cur2` на `cur2->next` (аналогично `cur1->next->next`). В `cur2` надо поменять `prev` с `cur1` на `cur1->prev` и `next` поменять с `cur2->next` на `cur1`. Следует отметить, что менять адреса самих указателей не рекомендуется, потому что возможна потеря адреса какого элемента и в последствии приведет к неправильной работе программы. В случаях когда меняется голова или хвост следует лишь отметить, что один из указателей будет указывать на `NULL`, и поэтому код программы для этих случаев будет лишь слегка отличаться от общего случая.

Как запускается программа.

Вся программа состоит из нескольких файлов. Один файл-main.c, который и содержит код для выполнения основной задачи. Главные функции этого файла уже были описаны ранее в 4 лабораторной и в пункте 1 В файле “course.sh” содержится скрипт на языке bash. Благодаря этому скрипту можно передать основной программе (main.c) любой файл, который содержит текст и лежит в одной директории. Также в этом файле реализована утилита beep(См.Рис.3). Она нужна для того, чтобы компьютер подал звуковой сигнал,когда программа завершится. Это сделано для удобства программиста, так как если список будет состоять из тысячи и больше элементов, то сортировка может занять некоторое время.

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

Для того, чтобы показать работу функции llist_bubble_sort () используем код функции main из лабораторной работы No4 и исходные данные из нее же.Внутри функции main() происходит считывание значений и создание списка,состоящего из структур MusicalComposition, которые содержат: название группы,альбома, год выхода альбома и указатели на предыдущий и следующий элементы списка.

Сначала проверяется возможен ли push в списке и подсчет элементов(См.Рис.4).

Дальше идет сортировка списка функцией llist_bubble_sort () и после удаления одного элемента(См.Рис.5).

ЗАКЛЮЧЕНИЕ

Была поставлена задача – отсортировать список по убыванию поля year.Написанные функции llist_bubble_sort и swap успешно справляются с поставленной задачей.

Приложение

test.c

```
#include<stdio.h>
int main(){
printf("Hello Wolrd!!!\n");
return 0;
}
```

test2.c

//Следующая программа сортирует список чисел и выводит результат:

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

```
int num[10] = {
1,3,6,5,8,7,9,6,2,0
};
```

```
int comp (const int *, const int *);
```

```
int main(void)
{
int i;
printf("Original array: ");
for (i=0; i<10; i + +) printf("%d ",num[i]);
printf ("\n");
qsort(num, 10, sizeof (int), (int(*) (const void *, const void *)) comp);
printf("Sorted array: ");
for(i = 0; i <10; i + + ) printf("%d ", num[i]);
return 0;
}
```

```
/* сравнение двух целых */
int comp (const int *i, const int *j)
{
return *i - *j;
}
```