**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: *Линейные списки*

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Погодаев Н.М. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**1. функции для работы с двунаправленным линейным списком**

* 1. **Создание элемента**

1. MusicalComposition \*createMusicalComposition(char \*name, char \*author, int year)
2. {
3. MusicalComposition \*composition = (MusicalComposition\*)malloc(sizeof(MusicalComposition));
4. composition->name = name;
5. composition->author = author;
6. composition->year = year;
7. composition->next = NULL;
8. composition->previous = NULL;
9. return composition;

Функция принимает в качестве аргументов указатели на название композиции (**char**\* name) и ее автора (**char**\* author), а также год написания (**int** year). Происходит выделение памяти для структура типа MusicalComposition и заполняются ее переменные.

* 1. **Создание двунаправленного списка связанных элементов**

1. MusicalComposition \*createMusicalCompositionList(char \*\*array\_names, char \*\*array\_authors, int \*array\_years, int n)
2. {
3. head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);
4. MusicalComposition \*tmp = head;
5. for (int i = 1; i < n; i++) {
6. tmp->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);
7. tmp->next->previous = tmp;
8. tmp = tmp->next;
9. }
10. return head;
11. }

Функция принимает в качестве аргументов указатели на указатель на массив названий композиций (**char**\*\* array\_names) и их авторов (**char**\*\* array\_authors), указатель на массив лет написания (**int\*** array\_years), размер массива (**int** n). Сначала создается “голова” списка и происходит заполнение ее переменных данными, затем аналогичные операции происходят для всех последующих n-1 элементов списка

* 1. **Добавление элемента в начало, середину и конец списка**

1. void push\_course(MusicalComposition \*element)
2. {
3. MusicalComposition \*tmp = head;
4. while (tmp->next != NULL)
5. {
6. tmp = tmp->next;
7. }
8. tmp->next = element;
9. element->next = NULL;
10. element ->previous = tmp;
11. }

Функция принимает в качестве аргументов указатель на “голову” списка (MusicalComposition\* head) и элемент (MusicalComposition \*element\_end), который нужно добавить в конец списка. С помощью цикла while доходим до конца списка и вписываем элемент(element\_end).

* 1. **Удаление элементов из списка**

1. void removeEl(char\* name\_for\_remove)
2. {
3. MusicalComposition \*tmp = head;
4. while(tmp != NULL)
5. {
6. if (strcmp(tmp->name, name\_for\_remove) == 0)
7. {
8. if ( (tmp->previous == NULL) && (tmp->next != NULL) )
9. {
10. tmp = tmp->next;
11. tmp->previous = NULL;
12. head = tmp;
13. }
14. else
15. {
16. if ( (tmp->previous != NULL) && (tmp->next == NULL) )
17. {
18. tmp = tmp->previous;
19. tmp->next = NULL;
20. tmp = tmp->next;
21. }
22. else
23. {
24. if ( (tmp->previous != NULL) && (tmp->next != NULL) )
25. {
26. tmp->next->previous = tmp->previous;
27. tmp->previous->next = tmp->next;
28. tmp = tmp->next;
29. }
30. else
31. {
32. if ( (tmp->next == NULL) && (tmp->previous == NULL) )
33. {
34. tmp = NULL;
35. head = tmp;
36. }
37. }
38. }
39. }
40. }
41. else
42. {
43. tmp = tmp->next;
44. }
45. }
46. }

Функция принимает в качестве аргументов указатель на “голову” списка (MusicalComposition\* head) и название композиций (**char**\* name\_for\_remove), которые нужно удалить из списка. Для каждого его элемента происходит сравнение названия композиции с именем произведений, которые нужно удалить.

* 1. **Подсчет количества элементов в списке**

1. int Count()
2. {
3. MusicalComposition \*tmp = head;
4. int counter = 0;
5. while (tmp != NULL)
6. {
7. counter++;
8. tmp = tmp->next;
9. }
10. return counter;
11. }

Функция принимает в качестве аргументов указатель на “голову” списка (MusicalComposition\* head). Далее происходит подсчет элементов в списке, пока следующий элемент существует

* 1. **Вывод элементов списка в консоль**

1. void print\_names()
2. {
3. MusicalComposition \*tmp = head;
4. while (tmp != NULL)
5. {
6. printf("%s\n", tmp->name);
7. tmp = tmp->next;
8. }
9. }

Функция принимает в качестве аргументов указатель на “голову” списка (MusicalComposition\* head). Далее происходит вывод названия и даты написания композиции в списке, пока следующий элемент существует.

**Вывод:** В ходе работы был создан и описан двунаправленный список музыкальных композиций MusicalComposition и api для работы со списком. Также были созданы и описаны все необходимые функции и структуры для работы программы.