Отчет по лабораторной работе №VIII по курсу

Языки и методы программирования _____

Студент группы М8О-101Б-21 Постнов Александр Вячеславович, № по списку 17

Контакты www, e-mail: 61pav03@mail.ru

Работа выполнена: «» 202 2г.

Преподаватель: ___ каф. 806 _____Титов В.К._____

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 2022_ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Линейные списки _____
   _____
2. **Цель работы:** Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры с элементами целого типа(по согласованию с преподавателем) _____
   _____
   _____
3. **Задание** (*Вид списка: (17 div 2) % 6 + 1 = 3; Нестандартное действие: 17 % 15 + 1 = 3*)**:** Реализовать линейный однонаправленный список. Удалить из списка все элементы, предшествующие и последующие заданному значению.

4. **Оборудование(лабораторное)**:
   ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,
   НМД - ГБ, терминал- адрес -, принтер -
   Другие устройства -

   Оборудование ПЭВМ студента, если использовалось:
   Процессор AMD Ryzen 5 4500U, с ОП 8 ГБ
   Другие устройства -

5. **Программное обеспечение:**
   Операционная система семейства -, наименование - версия -
   интерпретатор команд - версия
   Система программирования - версия -
   Редактор текстов - версия -
   Утилиты операционной системы -
   Прикладные системы и программы -
   Местонахождение и имена файлов программ и данных -

   Программное обеспечение ЭВМ студента, если использовалось:
   Операционная система семейства GNU/Linux, наименование Manjaro версия 5-13-12-1
   интерпретатор команд GNOME Terminal версия 3.38.2.
   Система программирования _____ версия _____
   Редактор текстов emacs версия 3.27.20
   Утилиты операционной системы
   Прикладные системы и программы -
   Местонахождение и имена файлов программ и данных -

6. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Структура линейного однонаправленного списка:

1) поле body типа item(в моем случае int)
2) поле next типа link(указатель на след элемент списка)

Последний элемент не указывает на первый, так как он линейный.

Для удобства работы со списком создам два указателя этой структуры tail(указатель на первый элемент списка) и head(указатель на последний элемент списка).

Рассмотрим ***стандартные*** действия со списком:

1) add(item m) - Добавление элемента в конец списка. Если список пустой, то создаются 2 указателя tail и head, с пустым указателем на след элемент и с элементом m. Если не пустой, создаем временный указатель структуры списка t(next = 0, body = m), указатель на след элемент head будет равен t(head->next=t), двигаем head вперед(head = head->next). Если указатель на след элемент tail = 0(это может быть только если мы добавляем 2 элемент), то tail->next = t.
2) addFirst(item m) - Добавление элемента в начало списка. Если список пустой, инициализируем также 2 указателя tail и head. Если не пустой, создаем указатель структуры t(next = tail, body = m). tail меняем на t.
3) genlist(int n) - Генерация списка с кол-вом n случайных элементов. С помощью генератора рандома получаем число и записываем в список с помощью функции add(описана выше).
4) printList() - Печать списка. Записываем в переменную t указатель на первый элемент(t = tail). Выводим body t на экран, двигаем t вперед до тех пор, пока t != 0(t=t->next), пока не достигнет пустого указателя
5) sizeList() - Возвращает кол-во элементов списка. Используется такая же техника при печати списка(printList())
6) deleting(item m) - Удаляет выбранный элемент. Рассмотрим случай, когда нужно будет удалить первый элемент. Проверим указатель на след элемент, если он пустой, то это случай, когда список становится пустым. Поэтому удаляем этот элемент, tail = 0, head = 0. Если он не пустой, то двигаем tail вперед и удаляем этот элемент. Теперь рассмотрим случай, когда нужно удалить не первый элемент. Проходим по списку и проверяем, если значение след элемента = m, то записываем след элемент в переменную f, если f - это последний элемент, то двигаем head назад и удаляем f, если это не последний элемент, то указатель текущего элемента next будет равен f next, f удаляем. Если элемент, который нужно удалить, не будет найден, то сообщим об этом пользователю.
7) erase() - Удалить все элемента списка(сделать его пустым). Проходим по списку как в методе printList(), при этом удаляя элементы списка по очереди.
8) insert(item m1, item m2) - Вставить элемент m1 в список после элемента со значением m2. Проходим по списку, если текущий элемент будет равен m2, то проверяем, если след элемента не существует, то это значит, что нужно просто добавить в конец списка элемент m1, сделаем это с помощью метода add(m2). Если след элемент существует. То создаем указатель на переменную структуры f, f->next = t->next(будет на след элемент). Текущий элемент будет указывать на f. Если элемент в списке с значением m2 не будет найден, сообщим об этом пользователю.

Рассмотрим ***нестандартное*** действие: удалить из списка все элементы, предшествующие и последующие заданному значению.

Если переформулировать задание, то нужно удалить все элементы, кроме заданного(заданных, если элементов с таким значением несколько). Алгоритм:

1) Пройтись по списку и узнать кол-во элементов с заданным значением, если их 0, то говорим об этом пользователю и выходим из метода
2) Очищаем список с помощью метода erase(описан выше)
3) Добавляем в список заданный элемент(ы) с помощью метода add(описан выше).

В функции main работа со списком будет реализована с помощью menu.

7. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

```
#include <stdio.h>

#include <time.h>

#include <stdlib.h>
```

```
struct ls;

typedef ls *link;
typedef int item;

struct ls {
   item body;
   link next;
}*tail, *head, *t;

void randomize() {
   long a = time(0);
   srand(a);
}

void add(item m) {
   if (!tail) { //если элементов в списке 0
        tail = new ls;
        tail->body = m;
        tail->next = 0;
        head = new ls;
        head->body = m;
        head->next = 0;
   }
   else {
        t = new ls;
        t->body = m;
        t->next = 0;
        if (tail->next == 0) {
            tail->next = t;
        }
        head->next = t;
        head = head->next; //голова передвигается
   }
}

void addFirst(item m) { //добавление элемента в начало
   if (!head) { //если элементов в списке 0
```

```
        head = new ls;
        head->body = m;
        head->next = 0;
        tail = new ls;
        tail->body = m;
        tail->next = 0;
    }
    else {
        t = new ls;
        t->body = m;
        t->next = tail;
        tail = t;
    }
}

void genList(int n) { //генерация списка с кол-вом случайных элементов n
    for (int i = 0; i < n; i++) {
        item m = rand() % 155;
        add(m);
    }
}

int sizeList() {
    t = tail;
    int n = 0;
    do {
        n++;
        t = t->next;
    } while (t != 0);
    return n;
}

void printList() {
    if (!tail) { //список пустой
        printf("\nList is empty\n");
    }
    else {
        printf("\nList:\n[ ");
```

```
        t = tail;
        do {
            printf("%d ", t->body);
            t = t->next;
        } while (t != 0);
        //printf("%d ]\n", t->body);
        printf("]\n");
    }
}


void deleting(item m) { //удаление элемента из списка
    t = tail;
    int flag = 0;
    ls *f;
    if (t->body == m) { //случай если первый элемент нужно удалить
        if (t->next == 0) { //список станет пустым
            delete t;
            tail = 0;
            head = 0;
            return;
        }
        else { //удаляем текущий элемент и двигаем tail
            f = t;
            t = t->next;
            tail = t;
            delete f;
            return;
        }
    }
    do {
        if (t->next != 0) {
            if (t->next->body == m) {
                flag = 1;
                f = t->next;
                if (f->next != 0){
                    t->next = f->next;
                }
```

```
                else {
                    t->next = 0; //конец списка изменяется!!!
                    head = t;
                    break;
                }
                delete f;
            }
        }
        t = t->next;
    } while (t != 0);
    if (!flag) {
        printf("\nItem with this value not exists!\n");
    }
}

void erase() {
    t = tail;
    ls *f;
    if (t) {
        do {
            f = t;
            t = t->next;
            delete f;
        } while (t != 0);
    }
    tail = 0;
    head = 0;
}

void insert(item m1, item m2) {
    t = tail;
    int flag = 0;
    ls *f;
    do {
        if (t->body == m2) {
            flag = 1;
            if (t->next == 0) { //нужно просто добавить в конец списка число
                add(m1);
```

```
                return;
            }
            else {
                f = new ls;
                f->body = m1;
                f->next = t->next;
                t->next = f;
                return;
            }
        }
        t = t->next;
    } while (t != 0);
    if (!flag) {
        printf("\nItem with this value not exists!\n");
    }
}


void action(item m) {
    if (!tail) {
        return;
    }
    int counter = 0; //кол-во элементов с таким значением в списке
    t = tail;
    do {
        if (t->body == m) {
            counter++;
        }
        t = t->next;
    } while (t != 0);
    if (counter == 0) {
        printf("\nItem with this value not exists!\n");
        return;
    }
    erase(); //очищаем список
    for (int i = 0; i < counter; i++) {
        add(m); //добавляем выбранный элемент counter раз
    }
```

```
}


int main() {
   int k = 10;
   randomize();
   for (;;) {
       if (k == 10) {
           printf("\nInput from 1 to 10 or 0 for actions:\n"
                "0. Exit.\n"
                "1. Generation list.\n"
                "2. Print list.\n"
                "3. Addition new item in begin of list.\n"
                "4. Addition new item in end of list.\n"
                "5. Erase list.\n"
                "6. Deleting from list.\n"
                "7. Lenght of list.\n"
                "8. Inserting in list.\n"
                "9. Main Action. \n"
                "10. Menu.\n");
       }
       else if (!k) {
           break;
       }
       else if (k == 1) {
           int n;
           printf("\nInput number of new items of list: n=");
           scanf("%d", &n);
           genList(n);
       }
       else if (k == 2) {
           printList();
       }
       else if (k == 3) {
           item m;
           printf("\nInput value of adding item: m=");
           scanf("%d", &m);
           addFirst(m);
```

```
}
else if (k == 4) {
    item m;
    printf("\nInput value of adding item: m=");
    scanf("%d", &m);
    add(m);
}
else if (k == 5) {
    erase();
}
else if (k == 6) {
    if (!tail) {
        printf("\nList is empty!\n");
    }
    else {
        printf("\nInput value of deleting item: m=");
        item m;
        scanf("%d", &m);
        deleting(m);
    }
}
else if (k == 7) {
    int n = 0;
    if (head) {
        n = sizeList();
    }
    printf("\nLenght of list=%d\n", n);
}
else if (k == 8) {
    if (!tail) {
        printf("\nList is empty!\n");
    }
    else {
        int m1, m2;
        printf("\nInput value inserting item: m1=");
        scanf("%d", &m1);
        printf("\nInput value item after which to insert: m2=");
        scanf("%d", &m2);
```

```
                insert(m1, m2);
            }
        }
        else if (k == 9) {
            if (!tail) {
                printf("\nList is empty!\n");
            }
            else {
                item m;
                printf("\nSelect the item that will remain in the list: m=");
                scanf("%d", &m);
                action(m);
            }
        }
        else {
            printf("Have no such number of MENU\n");
        }
        printf("\nInput number of MENU: k=");
        scanf("%d", &k);
    }
    return 0;
}
```

Тестирование:

1) Создам случайный список
2) Продемонстрирую стандартные операции со списком
3) Продемонстрирую нестандартное действие по варианту со списком
4) Ответы буду проверять ручным просчётом

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

Допущен к выполнению работы. **Подпись преподавателя _____**

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
[alex@fedora 8()]$ cat head.txt
-------------------------------------------------------
|          Лабораторная работа № VIII                  |
|          Линейные списки                             |
|          Выполнил:  студент группы М8О-101Б-21       |
|          Постнов Александр Вячеславович              |
-------------------------------------------------------
[alex@fedora 8()]$ cat main.cpp
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

struct ls;

typedef ls *link;
typedef int item;

struct ls {
   item body;
   link next;
}*tail, *head, *t;

void randomize() {
   long a = time(0);
   srand(a);
}

void add(item m) {
   if (!tail) { //если элементов в списке 0
      tail = new ls;
      tail->body = m;
      tail->next = 0;
      head = new ls;
      head->body = m;
      head->next = 0;
   }
   else {
      t = new ls;
      t->body = m;
      t->next = 0;
      if (tail->next == 0) {
         tail->next = t;
      }
      head->next = t;
      head = head->next; //голова передвигается
   }
}
```

```
void addFirst(item m) { //добавление элемента в начало
   if (!head) { //если элементов в списке 0
      head = new ls;
      head->body = m;
      head->next = 0;
      tail = new ls;
      tail->body = m;
      tail->next = 0;
   }
   else {
      t = new ls;
      t->body = m;
      t->next = tail;
      tail = t;
   }
}

void genList(int n) { //генерация списка с кол-вом случайных элементов n
   for (int i = 0; i < n; i++) {
      item m = rand() % 155;
      add(m);
   }
}

int sizeList() {
   t = tail;
   int n = 0;
   do {
      n++;
      t = t->next;
   } while (t != 0);
   return n;
}

void printList() {
   if (!tail) { //список пустой
      printf("\nList is empty\n");
   }
   else {
      printf("\nList:\n[ ");
      t = tail;
      do {
         printf("%d ", t->body);
         t = t->next;
      } while (t != 0);
      //printf("%d ]\n", t->body);
      printf("]\n");
   }
}


void deleting(item m) { //удаление элемента из списка
   t = tail;
```

```
  int flag = 0;
  ls *f;
  if (t->body == m) { //случай если первый элемент нужно удалить
    if (t->next == 0) { //список станет пустым
      delete t;
      tail = 0;
      head = 0;
      return;
    }
    else { //удаляем текущий элемент и двигаем tail
      f = t;
      t = t->next;
      tail = t;
      delete f;
      return;
    }
  }
  do {
    if (t->next != 0) {
      if (t->next->body == m) {
        flag = 1;
        f = t->next;
        if (f->next != 0){
          t->next = f->next;
        }
        else {
          t->next = 0; //конец списка изменяется!!!
          head = t;
          break;
        }
        delete f;
      }
    }
    t = t->next;
  } while (t != 0);
  if (!flag) {
    printf("\nItem with this value not exists!\n");
  }
}

void erase() {
  t = tail;
  ls *f;
  if (t) {
    do {
      f = t;
      t = t->next;
      delete f;
    } while (t != 0);
  }
  tail = 0;
  head = 0;
}
```

```
void insert(item m1, item m2) {
   t = tail;
   int flag = 0;
   ls *f;
   do {
      if (t->body == m2) {
         flag = 1;
         if (t->next == 0) { //нужно просто добавить в конец списка число
            add(m1);
            return;
         }
         else {
            f = new ls;
            f->body = m1;
            f->next = t->next;
            t->next = f;
            return;
         }
      }
      t = t->next;
   } while (t != 0);
   if (!flag) {
      printf("\nItem with this value not exists!\n");
   }
}


void action(item m) {
   if (!tail) {
      return;
   }
   int counter = 0; //кол-во элементов с таким значением в списке
   t = tail;
   do {
      if (t->body == m) {
         counter++;
      }
      t = t->next;
   } while (t != 0);
   if (counter == 0) {
      printf("\nItem with this value not exists!\n");
      return;
   }
   erase(); //очищаем список
   for (int i = 0; i < counter; i++) {
      add(m); //добавляем выбранный элемент counter раз
   }
}


int main() {
   int k = 10;
   randomize();
   for (;;) {
```

```
if (k == 10) {
   printf("\nInput from 1 to 10 or 0 for actions:\n"
      "0. Exit.\n"
      "1. Generation list.\n"
      "2. Print list.\n"
      "3. Addition new item in begin of list.\n"
      "4. Addition new item in end of list.\n"
      "5. Erase list.\n"
      "6. Deleting from list.\n"
      "7. Lenght of list.\n"
      "8. Inserting in list.\n"
      "9. Main Action. \n"
      "10. Menu.\n");
}
else if (!k) {
   break;
}
else if (k == 1) {
   int n;
   printf("\nInput number of new items of list: n=");
   scanf("%d", &n);
   genList(n);
}
else if (k == 2) {
   printList();
}
else if (k == 3) {
   item m;
   printf("\nInput value of adding item: m=");
   scanf("%d", &m);
   addFirst(m);
}
else if (k == 4) {
   item m;
   printf("\nInput value of adding item: m=");
   scanf("%d", &m);
   add(m);
}
else if (k == 5) {
   erase();
}
else if (k == 6) {
   if (!tail) {
      printf("\nList is empty!\n");
   }
   else {
      printf("\nInput value of deleting item: m=");
      item m;
      scanf("%d", &m);
      deleting(m);
   }
}
else if (k == 7) {
   int n = 0;
```

```
        if (head) {
            n = sizeList();
        }
        printf("\nLenght of list=%d\n", n);
    }
    else if (k == 8) {
        if (!tail) {
            printf("\nList is empty!\n");
        }
        else {
            int m1, m2;
            printf("\nInput value inserting item: m1=");
            scanf("%d", &m1);
            printf("\nInput value item after which to insert: m2=");
            scanf("%d", &m2);
            insert(m1, m2);
        }
    }
    else if (k == 9) {
        if (!tail) {
            printf("\nList is empty!\n");
        }
        else {
            item m;
            printf("\nSelect the item that will remain in the list: m=");
            scanf("%d", &m);
            action(m);
        }
    }
    else {
        printf("Have no such number of MENU\n");
    }
    printf("\nInput number of MENU: k=");
    scanf("%d", &k);
  }
  return 0;
}[alex@fedora 8()]$ g++ main.cpp
[alex@fedora 8()]$ ./a.out

Input from 1 to 10 or 0 for actions:
0. Exit.
1. Generation list.
2. Print list.
3. Addition new item in begin of list.
4. Addition new item in end of list.
5. Erase list.
6. Deleting from list.
7. Lenght of list.
8. Inserting in list.
9. Main Action.
10. Menu.

Input number of MENU: k=1
```

Input number of new items of list: n=5

Input number of MENU: k=2

List:
[148 29 153 130 104]

Input number of MENU: k=3

Input value of adding item: m=45

Input number of MENU: k=3

Input value of adding item: m=44

Input number of MENU: k=2

List:
[44 45 148 29 153 130 104]

Input number of MENU: k=4

Input value of adding item: m=1

Input number of MENU: k=4

Input value of adding item: m=2

Input number of MENU: k=4

Input value of adding item: m=3

Input number of MENU: k=2

List:
[44 45 148 29 153 130 104 1 2 3]

Input number of MENU: k=6

Input value of deleting item: m=44

Input number of MENU: k=6

Input value of deleting item: m=2

Input number of MENU: k=2

List:
[45 148 29 153 130 104 1 3]

Input number of MENU: k=6

Input value of deleting item: m=130

Input number of MENU: k=2

List:
[45 148 29 153 104 1 3]

Input number of MENU: k=7

Lenght of list=7

Input number of MENU: k=8

Input value inserting item: m1=2

Input value item after which to insert: m2=1

Input number of MENU: k=2

List:
[45 148 29 153 104 1 2 3]

Input number of MENU: k=5

Input number of MENU: k=2

List is empty

Input number of MENU: k=10

Input from 1 to 10 or 0 for actions:
0. Exit.
1. Generation list.
2. Print list.
3. Addition new item in begin of list.
4. Addition new item in end of list.
5. Erase list.
6. Deleting from list.
7. Lenght of list.
8. Inserting in list.
9. Main Action.
10. Menu.

Input number of MENU: k=1

Input number of new items of list: n=5

Input number of MENU: k=2

List:
[73 30 41 76 17]

Input number of MENU: k=4

Input value of adding item: m=30

Input number of MENU: k=2

List:
[73 30 41 76 17 30]

Input number of MENU: k=9

Select the item that will remain in the list: m=30

Input number of MENU: k=2

List:
[30 30]

Input number of MENU: k=9

Select the item that will remain in the list: m=1

Item with this value not exists!

Input number of MENU: k=4

Input value of adding item: m=1

Input number of MENU: k=2

List:
[30 30 1]

Input number of MENU: k=9

Select the item that will remain in the list: m=1

Input number of MENU: k=2

List:
[1]

Input number of MENU: k=4

Input value of adding item: m=25

Input number of MENU: k=4

Input value of adding item: m=25

Input number of MENU: k=2

List:
[1 25 25]

Input number of MENU: k=9

Select the item that will remain in the list: m=25

Input number of MENU: k=2

List:
[25 25]

Input number of MENU: k=9

Select the item that will remain in the list: m=25

Input number of MENU: k=2

List:
[25 25]

Input number of MENU: k=0

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание
1	дом	28.04.2	13:00	В некоторых методах не проверял, есть ли в списке какие-то элементы, что приводило к ошибкам.	Проверял, есть ли в списке элементы	

10. Замечания автора

11.**Выводы**

В ходе лабораторной работы я изучил такую структуру данных как списки, понял недостатки и преимущества перед
массивами._____

_____
_____
_____
_____
_____
_____
_____

Недочёты при выполнении задания могут быть устранены следующим образом:
_____
_____
_____
_____
_____

Подпись студента ____Постнов_____