Отчет по курсовой работе VIII

по курсу: 1 фундаментальная информатика

студента группы М80-101Б-21 Тулина Ивана, № по списку: 22

Контакты www, e-mail, icq, skype: i.tulin0107@gmail.com

Работа выполнена: «20» июня 2022г.

Преподаватель: Титов В. К. каф. 806

Входной контроль знаний с оценкой _____

Отчет сдан « » _____201 __ г., итоговая оценка _____

Подпись преподавателя _____

1.1 **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. Навигацию по списку следует реализовывать с применением итераторов. Предусмотреть выполнение одного нестандартного и четырех стандартных действий: печать списка, вставка нового элемента в список, удаление из списка, подсчет длины списка.

3 **Задание** (*вариант №* **22**)**:** *тип элемента списка:* вещественный, *вид списка:* линейный двунаправленный с барьерным элементом, *нестандартное действие:* дополнить список копиями заданного значения до указанной длины *k*. Если в списке уже имеется не менее *k* элементов, то не менять его.

4 **Оборудование** (лабораторное):
ЭВМ_____-_____, процессор_____-_____, имя узла сети _____-_____ с ОП _____-_____ Мб,
НМД _____ Мб. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
_____

Оборудование ПЭВМ студента, если использовалось:
Процессор Intel Core i5-7300HQ с ОП 7,87 Мб, НМД 15360 Мб. Монитор: встроенный
Другие устройства _____
_____

5 Программное обеспечение (лабораторное):
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 UNIX, наименование Ubuntu версия 20.04.3 LTS
интерпретатор команд bash версия _____.
Система программирования _____версия _____
Редактор текстов Emacs версия 3.22.30
Утилиты операционной системы -
Прикладные системы и программы _____-_____
Местонахождение и имена файлов программ и данных на домашнем компьютере _____-_____

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

Двунаправленный список с барьерным элементом представляет из себя замкнутую цепочку узлов с дополнительным барьерным элементом, который связан указателями с первым и последним элементом списка. Этот элемент служит для того, чтобы обозначать границы списка (принадлежащее ему значение не важно).

Идея:

Построение списка всегда будем начинать с барьера и каждый элемент добавленный в конец списка будет помещаться непосредственно перед барьером (соответственно каждый элемент, помещенный в начало списка, будет помещен сразу после барьера). Таким образом, существование списка можно будет проверять существованием барьера, который станет глобальной переменной программы.

Поскольку в работе поставлена задача отображения списка при помощи динамических структур, память под каждый узел будем выделять на куче при помощи оператора **new**. Сам по себе узел (он же элемент списка) представляет из себя структуру из трех полей: значения элемента, ссылку на предыдущий элемент и на следующий элемент. В значение элемента будем помещать вещественное число, а в поля с ссылками, как можно догадаться, ссылки на соседние 2 элемента в списке. Чтобы реализовать ссылочные поля необходимо описать тип, представляющий из себя множество указателей на структуру, описывающую узел. Для этого сначала объявим структуру, затем опишем ссылочный тип при помощи **typedef** и лишь затем опишем саму структуру.

За добавление элементов в список будет отвечать несколько процедур.

Процедура **add** будет добавлять элемент в конец списка.
При вызове она проверит существование списка (существование барьерного элемента).
Если барьера нет, она создаст новый барьерный элемент. В оба поля ссылок заранее созданного первого узла будут помещены указатели на барьерный элемент, а в поля ссылок барьерного элемента будут помещены ссылки на созданный узел.
Если барьер уже будет существовать, в поле ссылок на последующий элемент нового узла будет помещен указатель на барьер, а в поле ссылок на предыдущий узел барьерного элемента будет помещена ссылка на созданный узел. При этом в поле ссылок нового узла на предыдущий элемент будет помещен указатель на элемент, шедший перед барьером. Соответственно в поле элемента, шедшего перед барьером, будет помещен указатель на созданный узел.
В поле значений созданного узла в обоих случаях будет помещено вещественное число, переданное функции в качестве аргумента.

Процедура **add_first** будет добавлять элемент в начало списка.
Аналогично предыдущей процедуре здесь будет происходить проверка существования барьерного элемента, а также заполнение полей барьера и первого узла в случае его отсутствия.
В случае существования барьера, в поле ссылок на предыдущий элемент созданного узла будет помещен указатель на барьер, а в поле ссылок на последующий элемент барьера – указатель на созданный узел. При этом в поле ссылок на последующий элемент нового узла будет помещена ссылка на элемент, который шел после барьера. А в поле ссылок на предшествующий элемент элемента, шедшего после барьера, будет помещен указатель на созданный узел.
В поле значений, аналогично предыдущей процедуре, будет помещено вещественное число, переданное процедуре в качестве аргумента.

Процедура **insert** будет вставлять новый элемент в список, сразу за первым найденным элементом с заданным значением.
В этой процедуре тоже будет проверяться существование списка, однако создание нового списка производиться не будет, так как в пустом списке нет элементов, которые удовлетворили бы условиям поиска. Значение элемента, которое мы будем искать в списке, будет передаваться функции в качестве аргумента, как и значение элемента который мы собираемся в этот список вставлять. Начало поиска по списку будет начинаться с элемента, идущего сразу после барьера. Если значение этого узла не будет соответствовать искомому значению, то процедура будет переходить к следующему узлу. Если значение узла совпадет с искомым значением, то процедура создаст новый узел, передаст ему требуемое значение и в его ссылочные поля занесет указатели на найденный элемент и элемент, шедший за найденным. При этом она положит в ссылочное поле найденного элемента (указывающее на следующий элемент) и элемента, следовавшего за найденным (указывающее на предыдущий элемент), указатель на созданный узел.

Процедура **gen_list** будет создавать новый список и заполнять его элементами, со случайно сгенерированными значениями.
Процедура использует функцию **rand**, определенную в заголовочном файле **stdlib.h**, для генерации значений новых элементов в списке. Чтобы сгенерированные значения из запуска в запуск программы отличались друг от друга, точку начала генерации чисел необходимо при каждом запуске программы переопределять при помощи функции **srand**, в аргументы которой мы будем передавать целочисленное значение времени. Для удобства вынесем это действие в отдельную функцию **randomize**.
Итак, процедура **gen_list** состоит из цикла **for**, в котором генерируется случайное целое число и делится на 10 в случайно сгенерированной целочисленной степени (возведение в степень происходит во вложенном цикле **for**), а затем

при помощи процедуры **add** это полученное число добавляет в список. В случае, если список уже существует, он просто будет пополняться сгенерированными значениями.
Для работы следующей процедуры, добавляющий элементы в список потребуется функция, подсчитывающая количество элементов в списке **lenght_list**.
Эта функция создает и инициализирует нулем счетчик, а затем в цикле проходит от первого до последнего элемента списка, увеличивая его и останавливаясь при достижении барьерного элемента. Функция возвращает значение счетчика. Если список не существует, функция возвратит ноль.

Процедура **action** будет выполнять заданное нестандартное действие.
В качестве аргументов процедура получает значение элемента, который надо добавить в список, и целое число, которое означает количество повторений этого действия.
При помощи функции **lenght_list** процедура проверяет количество элементов в списке. Если их меньше, чем заданная граница, то в цикле **for** при помощи процедуры **add** список дополняется элементам с заданным значением. Так как реализация процедуры будет происходить при помощи ранее описанного **add**, программа будет выполняться даже в случае отсутствия списка.

За удаление элементов из списка будет отвечать две процедуры.

Процедура **delete_elem** будет удалять элемент с заданным значением.
В качестве аргумента процедуре будет передаваться значение элемента, который необходимо удалить. Первоначально процедура будет проверять существование списка. Затем (только если список существует) в цикле она будет искать заданное значение, пока не встретит первое совпавшее, либо барьерный элемент.
Найдя элемент с заданным значением, процедура обратится к предшествовавшему ему элементу и запишет в его ссылочное поле, указывавшее на найденный элемент, указатель на элемент следовавший за найденным. В свою очередь для элемента следовавшего за найденным, в поле, указывавшее на найденный элемент, она запишет указатель на элемент предшествовавший найденному. И в конце произведет удаление найденного узла при помощи встроенной функции языка **delete**.

Процедура **erase_list** будет стирать весь список целиком.
Первоначально процедура будет проверять существование списка. Затем (только если список существует) в цикле, начиная с элемента следующего за барьерным, при помощи **delete** будут стираться узлы списка. Предварительно в каждой итерации цикла будет сохраняться указатель на элемент, последующий за удаляемым, чтобы перейти к нему в следующей итерации цикла. При достижении барьера цикл будет останавливаться. В указатель на барьерный элемент процедура просто запишет ноль.

За вывод элементов списка будет отвечать процедура **print_list**.
Она, как и другие процедуры, будет проверять существование списка. В случае его присутствия в цикле будут выводиться значения узлов и происходить переходы по ссылкам от элемента к элементу (от первого к последнему).

Функция **main** в программе будет отвечать за вызов вышеописанных процедур и функций.

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<stdlib.h>
#include<time.h>

struct ls;
typedef ls *link;


struct ls {
   float body;
   link next, prev;
} *barrier, *list, *elem;


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


void add(float m){
   list=new ls; list->body=m;
   if(!barrier)
      {
```

```
        barrier=new ls;
        list->prev=barrier;
        barrier->next=list;
        }
    else
        {
        list->prev=barrier->prev;
        barrier->prev->next=list;
        }
    list->next=barrier;
    barrier->prev=list;
}


void add_first(float m){
    list=new ls; list->body=m;
    if(!barrier)
        {
        barrier=new ls;
        list->next=barrier;
        barrier->prev=list;
        }
    else
        {
        list->next=barrier->next;
        barrier->next->prev=list;
        }
    list->prev=barrier;
    barrier->next=list;
}


void print_list(){
    if(!barrier) printf("List is empty\n\n");
    else
   {
        list=barrier->next; printf("[");
        do  {
            printf(" %1.2f ", list->body);
            list=list->next;
            }
        while(list!=barrier); printf("]\n\n");
   }
}


void delete_elem(float m){
    if(!barrier) printf("List is aldeady empty\n\n");
    else
   {
    list=barrier->next; int c=1;
    while(list!=barrier)
        {
        if(list->body==m)
            {
            list->prev->next=list->next;
            list->next->prev=list->prev;
            delete list; c=0; break;
            }
        list=list->next;
        }
    if(c) printf("There is no element"
                 "equal m in list\n\n");
   }
}


void insert(float m, float m1){
    if(!barrier) printf("List is empty\n\n");
    else
        {
        list=barrier->next; int c=1;
        while(list!=barrier)
            {
            if(list->body==m)
```

```
            {
             elem=new ls; elem->body=m1;
             elem->prev=list;
             elem->next=list->next;
             list->next->prev=elem;
             list->next=elem; c=0; break;
            }
          list=list->next;
         }
       if(c) printf("There is no element"
                    "equal %f in list\n\n", m);
      }

}


void gen_list(int k){
   for(int i=0; i<k; i++)
      {
       int div=1, p=rand()%3+1;
       for(int j=1; j<p; j++) div*=10;
       float m=(rand()%100)/((float)div);
       add(m);
      }
}


void erase_list(){
   if(!barrier) printf("List is already empty\n\n");
   else
      {
       list=barrier->next;
       while(list!=barrier)
          {
           elem=list;
           list=list->next;
           delete elem;
          }
       barrier=0;
      }
}


int lenght_list(){
   if(!barrier) return 0;
   int counter=0;
   list=barrier->next;
   while(list!=barrier)
      {
       counter++;
       list=list->next;
      }
   return counter;
}


void action(int k, float m){
   int l=lenght_list();
   if(l>=k) printf("There are already %d "
                   "elements in list\n\n", k);
   else for(int i=l; i<k; i++) add(m);
}


void menu(){
   printf( "Choose a number\n"
           "1-print list\n"
           "2-add element to the end of list\n"
           "3-add element to the begin of list\n"
           "4-insert element to the list\n"
           "5-delete element from list\n"
           "6-print lenght of list\n"
           "7-generate list\n"
           "8-erase list\n"
           "9-main action\n"
```

```
          "10-print this menu\n"
          "0-exit program\n\n");
}


int main(){
   float m, m1; int k, n=10;
   randomize();
   for(;;)
     {
      if(n==1)
        {
         printf("Now list is looking like:\n");
         print_list();
        }
      else if(n==2)
        {
         printf("Enter new element\n");
         scanf("%f", &m); add(m);
        }
      else if(n==3)
        {
         printf("Enter new element\n");
         scanf("%f", &m); add_first(m);
        }
      else if(n==4)
        {
         printf("Where do you want to insert new element?\n"
                "Enter value of neighboring element\n");
         scanf("%f", &m);
         printf("Enter value of inserting element\n");
         scanf("%f", &m1); insert(m, m1);
        }
      else if(n==5)
        {
         printf("What element you want to delete?\n"
                 "Enter the value\n");
         scanf("%f", &m); delete_elem(m);
        }
      else if(n==6)
         printf("Lenght of list l=%d\n\n", lenght_list());
      else if(n==7)
        {
         printf("Enter number of elements in new list\n");
         scanf("%d", &k); gen_list(k);
        }
      else if(n==8)
        {
         printf("Erasing list...\n\n");
         erase_list();
        }
      else if(n==9)
        {
         printf("Enter lower bound of number of elements in list\n");
         scanf("%d", &k);
         printf("Enter value of elements you want to add\n");
         scanf("%f", &m); action(k ,m);
        }
      else if(n==10)
         menu();
      else if(n==0)
         break;
      printf("Select the instruction number\n");
      scanf("%d", &n);
     }
return 0;}
```

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

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

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

```
yusayu@YS:~/Рабочий стол/cppProjects$ cat head
**********************************************************************
*                    Курсовая работа VIII                            *
*                     Линейные списки                                *
*              Выполнил: Тулин Иван Денисович                        *
*                  (номер по списку: 22)                             *
*                    Группа: М8О-101Б-21                             *
**********************************************************************
yusayu@YS:~/Рабочий стол/cppProjects$ cat kr8.cpp
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

struct ls;
typedef ls *link;


struct ls {
   float body;
   link next, prev;
} *barrier, *list, *elem;


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


void add(float m){
   list=new ls; list->body=m;
   if(!barrier)
      {
       barrier=new ls;
       list->prev=barrier;
       barrier->next=list;
      }
   else
      {
       list->prev=barrier->prev;
       barrier->prev->next=list;
      }
   list->next=barrier;
   barrier->prev=list;
}


void add_first(float m){
   list=new ls; list->body=m;
   if(!barrier)
      {
       barrier=new ls;
       list->next=barrier;
       barrier->prev=list;
      }
   else
      {
       list->next=barrier->next;
```

```
        barrier->next->prev=list;
       }
    list->prev=barrier;
    barrier->next=list;
}


void print_list(){
    if(!barrier) printf("List is empty\n\n");
    else
   {
        list=barrier->next; printf("[");
        do  {
            printf(" %1.2f ", list->body);
            list=list->next;
            }
        while(list!=barrier); printf("]\n\n");
   }
}


void delete_elem(float m){
    if(!barrier) printf("List is aldeady empty\n\n");
    else
   {
    list=barrier->next; int c=1;
    while(list!=barrier)
       {
        if(list->body==m)
           {
            list->prev->next=list->next;
            list->next->prev=list->prev;
            delete list; c=0; break;
           }
        list=list->next;
       }
    if(c) printf("There is no element"
                 "equal m in list\n\n");
   }
}


void insert(float m, float m1){
    if(!barrier) printf("List is empty\n\n");
    else
       {
        list=barrier->next; int c=1;
        while(list!=barrier)
           {
            if(list->body==m)
               {
                elem=new ls; elem->body=m1;
                elem->prev=list;
                elem->next=list->next;
                list->next->prev=elem;
                list->next=elem; c=0; break;
               }
            list=list->next;
           }
        if(c) printf("There is no element"
                     "equal %f in list\n\n", m);
```

```
      }

}


void gen_list(int k){
   for(int i=0; i<k; i++)
      {
       int div=1, p=rand()%3+1;
       for(int j=1; j<p; j++) div*=10;
       float m=(rand()%100)/((float)div);
       add(m);
      }
}


void erase_list(){
   if(!barrier) printf("List is already empty\n\n");
   else
      {
       list=barrier->next;
       while(list!=barrier)
          {
           elem=list;
           list=list->next;
           delete elem;
          }
       barrier=0;
      }
}


int lenght_list(){
   if(!barrier) return 0;
   int counter=0;
   list=barrier->next;
   while(list!=barrier)
      {
       counter++;
       list=list->next;
      }
   return counter;
}


void action(int k, float m){
   int l=lenght_list();
   if(l>=k) printf("There are already %d "
                   "elements in list\n\n", k);
   else for(int i=l; i<k; i++) add(m);
}


void menu(){
   printf( "Choose a number\n"
           "1-print list\n"
           "2-add element to the end of list\n"
           "3-add element to the begin of list\n"
           "4-insert element to the list\n"
           "5-delete element from list\n"
           "6-print lenght of list\n"
```

```
            "7-generate list\n"
            "8-erase list\n"
            "9-main action\n"
            "10-print this menu\n"
            "0-exit program\n\n");
}


int main(){
    float m, m1; int k, n=10;
    randomize();
    for(;;)
       {
        if(n==1)
           {
            printf("Now list is looking like:\n");
            print_list();
           }
        else if(n==2)
           {
            printf("Enter new element\n");
            scanf("%f", &m); add(m);
           }
        else if(n==3)
           {
            printf("Enter new element\n");
            scanf("%f", &m); add_first(m);
           }
        else if(n==4)
           {
            printf("Where do you want to insert new element?\n"
                   "Enter value of neighboring element\n");
            scanf("%f", &m);
            printf("Enter value of inserting element\n");
            scanf("%f", &m1); insert(m, m1);
           }
        else if(n==5)
           {
            printf("What element you want to delete?\n"
                    "Enter the value\n");
            scanf("%f", &m); delete_elem(m);
           }
        else if(n==6)
            printf("Lenght of list l=%d\n\n", lenght_list());
        else if(n==7)
           {
            printf("Enter number of elements in new list\n");
            scanf("%d", &k); gen_list(k);
           }
        else if(n==8)
           {
            printf("Erasing list...\n\n");
            erase_list();
           }
        else if(n==9)
           {
            printf("Enter lower bound of number of elements in list\n");
            scanf("%d", &k);
            printf("Enter value of elements you want to add\n");
            scanf("%f", &m); action(k ,m);
           }
        else if(n==10)
```

```
            menu();
        else if(n==0)
            break;
        printf("Select the instruction number\n");
        scanf("%d", &n);
       }
return 0;}
yusayu@YS:~/Рабочий стол/cppProjects$ c++ kr8.cpp
yusayu@YS:~/Рабочий стол/cppProjects$ ./a.out
Choose a number
1-print list
2-add element to the end of list
3-add element to the begin of list
4-insert element to the list
5-delete element from list
6-print lenght of list
7-generate list
8-erase list
9-main action
10-print this menu
0-exit program

Select the instruction number
7
Enter number of elements in new list
4
Select the instruction number
1
Now list is looking like:
[ 7.80  96.00  8.20  7.90 ]

Select the instruction number
9
Enter lower bound of number of elements in list
8
Enter value of elements you want to add
6.67
Select the instruction number
1
Now list is looking like:
[ 7.80  96.00  8.20  7.90  6.67  6.67  6.67  6.67 ]

Select the instruction number
9
Enter lower bound of number of elements in list
5
Enter value of elements you want to add
8.43
There are already 5 elements in list

Select the instruction number
2
Enter new element
4.32
Select the instruction number
1
Now list is looking like:
[ 7.80  96.00  8.20  7.90  6.67  6.67  6.67  6.67  4.32 ]

Select the instruction number
3
```

```
Enter new element
8.77
Select the instruction number
1
Now list is looking like:
[ 8.77  7.80  96.00  8.20  7.90  6.67  6.67  6.67  6.67  4.32 ]

Select the instruction number
4
Where do you want to insert new element?
Enter value of neighboring element
6.67
Enter value of inserting element
2.1
Select the instruction number
1
Now list is looking like:
[ 8.77  7.80  96.00  8.20  7.90  6.67  2.10  6.67  6.67  6.67  4.32 ]

Select the instruction number
5
What element you want to delete?
Enter the value
96.00
Select the instruction number
1
Now list is looking like:
[ 8.77  7.80  8.20  7.90  6.67  2.10  6.67  6.67  6.67  4.32 ]

Select the instruction number
6
Lenght of list l=10

Select the instruction number
8
Erasing list...

Select the instruction number
1
Now list is looking like:
List is empty

Select the instruction number
8
Erasing list...

List is already empty

Select the instruction number
5
What element you want to delete?
Enter the value
3.3
List is aldeady empty

Select the instruction number
4
Where do you want to insert new element?
Enter value of neighboring element
2.1
Enter value of inserting element
```

```
3.4
List is empty

Select the instruction number
6
Lenght of list l=0

Select the instruction number
3
Enter new element
7.65
Select the instruction number
1
Now list is looking like:
[ 7.65 ]

Select the instruction number
10
Choose a number
1-print list
2-add element to the end of list
3-add element to the begin of list
4-insert element to the list
5-delete element from list
6-print lenght of list
7-generate list
8-erase list
9-main action
10-print this menu
0-exit program

Select the instruction number
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

10 **Замечания автора** по существу работы _____

11 Выводы

В ходе работы я научился составлять и отлаживать программы на языке Си для обработки линейных списков с отображением их на динамические структуры.

Недочёты при выполнении задания могут быть устранены следующим образом: _____

Подпись студента _____