Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

Практическая работа №1
по дисциплине «Структуры и организация данных»
на тему «Линейные структуры данных»

Выполнил:
Студент Альков В.С.
Группа И407Б

Преподаватель:
Полухин А.Л.

Санкт-Петербург
2021 г.

Задача 1.

Уровень сложности – **повышенный**. данные хранятся в бинарном файле записей, а для обработки считываются в двусвязный линейный список (если файл не существует, то создается пустой список). При выходе из программы обработанные данные сохраняются в том же файле. Имя файла с данными должно передаваться программе при ее запуске (через параметры функции main()). Если параметры пользователем при запуске программы не заданы, имя файла вводится с клавиатуры. Элементами данных должны являться записи с вариантами (вариативные поля придумать самостоятельно)**.** Обязательные операции: добавление элемента в упорядоченный список с сохранением упорядоченности (ключевое поле выбрать самостоятельно), просмотр списка в прямом и обратном направлении, удаление произвольного элемента списка, поиск в соответствии с индивидуальным вариантом. Вывод данных осуществлять постранично в табличном виде с графлением визуально подходящими символами, предусмотреть листание в прямом и обратном направлении.

Поля данных: фамилия, пол, вид спорта, год рождения, рост. Найти самого высокого спортсмена, занимающегося плаванием, среди мужчин. Вывести сведения о спортсменках, выступающих в юниорском разряде (14-17 лет).

Созданные типы данных:

1. Назначение типа - тип данных, которые будет хранить список.

   Объявление типа:

```
struct sportperson
{
        char name[79], gender, sport[32];
        int year, height;
};
```

   Назначение полей:

   name – ФИО человека, массив символом типа char;

   gender – пол человека, тип char;

   sport – вид спорта, которым занимается человек, массив символов типа char;

   year – год рождения человека, тип int;

   height – рост человека, тип int;

   Даем имя DataType типу struct sportperson

```
typedef struct sportperson DataType;
```

2. Назначение типа – элемент списка.

   Объявление типа:

```
struct element
{
   DataType data;
   struct element *next, *prev;
};
```

Назначение полей:

data – данные элемента, тип DataType;

next, prev – указатели на следующий и предыдущий элементы списка, тип struct element*.

3. Назначение типа – тип для хранения границ списка, будем называть его списком.

   Объявление типа:

```
struct List
{
   struct element *begin, *end;
};
```

   Назначение полей:

   begin, end – указатели на первый и последний элементы списка, тип struct element*;

Используемые функции:

1. Назначение – обнулить указатели на начало и конец списка, чтобы он трактовался как пустой.

   Заголовок функции:

```
void makenull(struct List* list)
```

   Входные данные:

   list - указатель на список, тип struct List*.

   Выходные данные: отсутствуют.

2. Назначение – добавление элемента в начало списка.

   Заголовок функции:

```
struct List add_begin(struct List list, DataType data)
```

   Входные данные:

   list - список, тип struct List;

   data – данные, которые будет хранить элемент, тип DataType.

   Выходные данные: список.

3. Назначение – добавление элемента в конец списка.

   Заголовок функции:

struct List add_end(struct List list, DataType data)

Входные данные:

list - список, тип struct List;

data – данные, которые будет хранить элемент, тип DataType.

Выходные данные: список.

4. Назначение – удаление последнего элемента списка.

   Заголовок функции:

   struct List delete_end(struct List list)

   Входные данные:

   list - список, тип struct List;

   Выходные данные: список.

5. Назначение – удаление первого элемента списка.

   Заголовок функции:

   struct List delete_begin(struct List list)

   Входные данные:

   list - список, тип struct List;

   Выходные данные: список.

6. Назначение – нахождение размера списка.

   Заголовок функции:

   int sizeOfList(struct List list)

   Входные данные:

   list - список, тип struct List;

   Выходные данные: размер списка.

7. Назначение – печать данных элемента списка в табличном виде.

   Заголовок функции:

   void printLine(DataType data, int i)

   Входные данные:

   list - список, тип struct List;

   i – номер элемента в списке, тип int.

   Выходные данные: отсутствуют.

8. Назначение – печать данных всего списка с пролистыванием в табличном виде.

   Заголовок функции:

   void printList(struct List list)

   Входные данные:

   list - список, тип struct List;

Выходные данные: отсутствуют.

9. Назначение – печать данных всего списка с пролистыванием в табличном виде в обратном порядке.

   Заголовок функции:

   void printListBack(struct List list)

   Входные данные:

   list - список, тип struct List;

   Выходные данные: отсутствуют.

10. Назначение – удаление элемента из списка по его порядковому номеру.

    Заголовок функции:

    struct List deleteElementByNumber(struct List list, int number)

    Входные данные:

    list - список, тип struct List;

    number – порядковый номер элемента в списке, тип int.

    Выходные данные: список.

11. Назначение – удаление всех элементов списка.

    Заголовок функции:

    struct List deleteList(struct List list)

    Входные данные:

    list - список, тип struct List.

    Выходные данные: список.

12. Назначение – добавление элемента в список после переданного элемента в функцию.

    Заголовок функции:

    void insert(struct element* element, DataType data)

    Входные данные:

    element – указатель на элемент списка, после которого нужно добавить, тип struct element*;

    data – данные, которые будет хранить элемент, тип DataType.

    Выходные данные: отсутствуют.

13. Назначение – упорядоченное добавление элемента в список по ФИО человека.

    Заголовок функции:

    struct List addWithSort(struct List list, DataType data)

    Входные данные:

    list - список, тип struct List;

data – данные, которые будет хранить элемент, тип DataType.

Выходные данные: список.

14. Назначение – чтение данных из файла в список.

    Заголовок функции:

    struct List readFileToList(char *filename)

    Входные данные:

    filename – имя файла, указатель на строку, тип char*.

    Выходные данные: список.

15. Назначение – запись данных списка в файл.

    Заголовок функции:

    void writeListToFile(struct List list, char *filename)

    Входные данные:

    list - список, тип struct List;

    filename – имя файла, указатель на строку, тип char*.

    Выходные данные: отсутствуют.

16. Назначение – приведение символов строки к нижнему регистру.

    Заголовок функции:

    void toLowerString(char* str)

    Входные данные:

    str – указатель на строку, тип char*;

    Выходные данные: отсутствуют.

17. Назначение – ввод с клавиатуры данных для элемента списка.

    Заголовок функции:

    DataType readData()

    Входные данные: отсутствуют.

    Выходные данные: считанные данные.

18. Назначение – вывод данных в виде визитницы.

    Заголовок функции:

    void print1(DataType *data)

    Входные данные:

    data – указатель на данные, тип DataType*.

    Выходные данные: отсутствуют.

19. Назначение – поиск самого высокого мужчины, занимающегося плаванием, в списке и печать на экран.

    Заголовок функции:

void findMan(struct List list)

Входные данные:

list - список, тип struct List.

Выходные данные: отсутствуют.

20. Назначение – поиск девушек в возрасте от 14 до 17 лет в списке, формирование из них нового списка и печать его на экран.

    Заголовок функции:

    void findWomen(struct List list)

    Входные данные:

    list - список, тип struct List.

    Выходные данные: отсутствуют.

Текст программы:

```
#include <iostream>
#include <string.h>
//#include <clocale>
#include <fstream>
struct sportperson
{
    char name[79], gender, sport[32];
    int year, height;
};
typedef struct sportperson DataType;
struct element
{
    DataType data;
    struct element *next, *prev;
};
struct List
{
    struct element *begin, *end;
};

void makenull(struct List* list)
{
    list->begin = NULL;
    list->end = NULL;
};

struct List add_begin(struct List list, DataType data)
{
    /*выделяем память под элемент*/
    struct element* temp = (struct element*) malloc (sizeof(struct element));
    /*добавляем в него данные*/
    temp->data = data;
    /*так как мы добавляем элемент в начало списка, предыдущего не может
быть*/
    temp->prev = NULL;
    /*назначаем следующим элементом первый в списке элемент*/
    temp->next = list.begin;
    /*если список не пустой, то первый в списке становится вторым,
следовательно предыдущим второго назначем только что созданный*/
    if (list.begin)
        list.begin->prev = temp;
```

```
    else
        /*если список был пуст, то добавляемый элемент будет первый и
последним в списке*/
        list.end = temp;
    /*назначаем добавляемый элемент началом списка*/
    list.begin = temp;
    /*возращаем список*/
    return list;
}

struct List add_end(struct List list, DataType data)
{
    /*выделяем память под элемент*/
    struct element* temp = (struct element*) malloc(sizeof(struct element));
    /*добавляем в него данные*/
    temp->data = data;
    /*так как мы добавляем элемент в конец списка, следующего не может быть*/
    temp->next = NULL;
    /*назначаем предыдущим элементом последний в списке элемент*/
    temp->prev = list.end;
    /*если список не пустой, то последний в списке становится предпоследним,
следовательно следующим предпоследнего назначем только что созданный*/
    if (list.end)
        list.end->next = temp;
    else
        /*если список был пуст, то добавляемый элемент будет первый и
последним в списке*/
        list.begin = temp;
    /*назначаем добавляемый элемент концом списка*/
    list.end = temp;
    /*возращаем список*/
    return list;
}

struct List delete_end(struct List list)
{
    /*объявляем указатель на удаляемый элемент, чтобы освободить память*/
    struct element *temp;
    /*если в списке есть элементы*/
    if (list.end)
    {
        //сохраняем адрес удаляемого элемента
        temp = list.end;
        /*последним в списке становится предпоследним*/
        list.end = list.end->prev;
        /*если в списке нет элементов, обнуляем указатель на первый элемент
списка*/
        if(!list.end)
            list.begin = NULL;
        else
            /*если список не пуст, то обнуляем указатель на следующий элемент
у последнего, так как сейчас он указавает на удаляемый*/
            list.end->next = NULL;
        /*очищаем память*/
        free(temp);
    }
    /*возращаем список*/
    return list;
}

struct List delete_begin(struct List list)
{
    /*объявляем указатель на удаляемый элемент, чтобы освободить память*/
```

```
    struct element *temp;
    /*если в списке есть элементы*/
    if (list.begin)
    {
        temp = list.begin;
        /*первым в списке становится второй*/
        list.begin = list.begin->next;
        /*если в списке нет элементов, обнуляем указатель на последний
элемент списка*/
        if (!list.begin)
            list.end = NULL;
        else
            /*если список не пуст, то обнуляем указатель на предыдущий
элемент у первого, так как сейчас он указавает на удаляемый*/
            list.begin->prev = NULL;
        /*очищаем память*/
        free(temp);
    };
    /*возращаем список*/
    return list;
}

int sizeOfList(struct List list)
{
    int i=0;
    /*проходим по списку и считаем кол-во элементов*/
    while(list.begin)
    {
        list.begin = list.begin->next;
        i++;
    }
    /*возращаем размер*/
    return i;
}

void printLine(DataType data, int i)
{
    printf("|%3d|%-31.31s|%-5c|%-12.12s|%5d|%6d|\n",i, data.name,
data.gender, data.sport,
    data.year, data.height);
    puts("-------------------------------------------------------------------
--");
}

void printList(struct List list)
{
    /* i - кол-во выведеных элементов списка, j - переменная для вывода
десяти элементов, k - переменная для сдвига на 9+j элеметов назад*/
    int i=0, j, k, num, size = sizeOfList(list);
    if(!list.begin)
    {
        std::cout<<"Список пуст";
        return;
    };
    while (1)
    {
        system ("cls");
        puts("| N |              ФИО              | Пол | Вид спорта | Год |
Рост |");
        puts("-------------------------------------------------------------
------");
        /*выводим первые десять элементов, считаем сколько элементов было
выведено с помощью i*/
```

```
        for (j=0; j<10; j++)
        {
            printLine(list.begin->data, ++i);
            /*если есть следующий элемент, то переходим на него*/
            if(list.begin->next)
                list.begin = list.begin->next;
            else
            {
                /*если следующего нет, то учитываем, что был выведен текущий
элемент, и выходим*/
                j++;
                break;
            };
        };
        puts("1. Вывести предыдущие 10 записей");
        puts("2. Вывести следующие 10 записей");
        puts("3. Прекратить вывод");
        scanf("%d%*c", &num);
        system ("cls");
        switch (num)
        {
            case 1 :
                /*если было выведено меньше 11, то предыдущих элементов нет*/
                if(i<11)
                {
                    puts("Предыдущих записей нет.");
                    return;
                }
                else
                    /*в другом случае, выполняем сдивг списка на 9+j, то есть
нам нужно вернуться на 9 элементов назад и еще на j, кол-во, выведеных
элементов в текущем проходе*/
                    for(k = 1; k<10+j; k++, list.begin = list.begin->prev);
                    /*уменьшаем кол-во выведеных*/
                    i-=10+j;
                break;
            case 2 :
                /*если элментов в списке больше нет*/
                if(i == size)
                {
                    puts("Следующих записей нет.");
                    return;
                };
                break;
            case 3 : return; break;
            default : std::cout << "Неправильный ввод"; return; break;
        };
    };
}

void printListBack(struct List list)
{
    int i=0, j, k, num, size = sizeOfList(list);
    if(!list.end)
    {
        std::cout<<"Список пуст";
        return;
    };
    while (1)
    {
        system ("cls");
        puts("| N |              ФИО              | Пол | Вид спорта | Год |
Рост |");
```

```
        puts("-----------------------------------------------------------------
------");
        for (j=0; j<10; j++)
        {
            printLine(list.end->data, ++i);
            if(list.end->prev)
                list.end = list.end->prev;
            else
            {
                j++;
                break;
            };
        };
        puts("1. Вывести предыдущие 10 записей");
        puts("2. Вывести следующие 10 записей");
        puts("3. Прекратить вывод");
        scanf("%d%*c", &num);
        system ("cls");
        switch (num)
        {
            case 1 :
                if(i<11)
                {
                    puts("Предыдущих записей нет.");
                    return;
                }
                else
                    for(k = 1; k<10+j; k++, list.end = list.end->next);
                    i-=10+j;
                break;
            case 2 :
                if(i == size)
                {
                    puts("Следующих записей нет.");
                    return;
                };
                break;
            case 3 : return; break;
            default : std::cout << "Неправильный ввод"; return; break;
        };
    };
}

struct List deleteElementByNumber(struct List list, int number)
{
    int size = sizeOfList(list);
    if (number>size||number<1)
        return list;
    if (number == 1)
        list = delete_begin(list);
    else
        if (number == size)
            list = delete_end(list);
    else
    {
        struct element *temp = list.begin;
        /*переходим к элементу по номеру*/
        for (int i=1; i<number; i++, temp=temp->next);
        /*предыдущему удаляемого элемента нужно переставить указатель,
следующий за ним стновится следущим за удаляемым*/
        temp->prev->next = temp->next;
        /*следующему за удаляемым элементу нужно переставить указатель,
предыдущий за ним стновится предыдущим за удаляемым*/
```

```
        temp->next->prev = temp->prev;
        free(temp);
    };
    return list;

}

struct List deleteList(struct List list)
{
    while(list.begin)
        list = delete_begin(list);
    return list;
}

void insert(struct element* element, DataType data)
{
    struct element* temp = (struct element*) malloc (sizeof(struct element));
    /*встраиваем элемент между двумя элементами списка*/
    temp->data = data;
    temp->prev = element;
    temp->next = element->next;
    element->next->prev = temp;
    element->next = temp;
}

struct List addWithSort(struct List list, DataType data)
{
    struct element* temp = list.begin;
    /*пока не дошли до конца списка и поле имени добавляемого элемента больше
поля имени элемента списка, переходим к следующему элементу списка*/
    while(temp&&strcmp(data.name, temp->data.name)>0)
        temp = temp->next;
    /*если дошли до конца списка, то есть добавляемый элемент больше всех в
списке, temp = NULL, поэтому нужно добавить в конец*/
    if(!temp)
        list = add_end(list, data);
    else
        /*если нет предыдущего элемента, то не было перехода к следующему
элементу списка, сл-но все элементы списка больше добавляемого, нужно
добавить элемент в начало*/
        if(!temp->prev)
            list = add_begin(list, data);
        else
            /*в другом случае добавляем перед элементом, который больше
добавляемого, в список*/
            insert(temp->prev, data);
    return list;
}

struct List readFileToList(char *filename)
{
    struct List list;
    DataType data;
    makenull(&list);
    /*открываем бинарный файловый поток ввода fin*/
    std::ifstream fin(filename, std::ios_base::binary);
    /*если поток открыся, то читаем данные из файла в список*/
    if (fin.is_open())
    {
        while(fin.read((char*)&data, sizeof(DataType)))
            list = addWithSort(list, data);
        /*закрываем файловый поток ввода*/
        fin.close();
```

```
    };
    return list;
}

void writeListToFile(struct List list, char *filename)
{
    /*открываем бинарный файловый поток вывода fout*/
    std::ofstream fout(filename, std::ios_base::binary);
    if (!fout.is_open())
        printf("Ошибка при открытии файла.\n");
    else
    {
        /*если поток открыся, то записывем список в файл*/
        while(list.begin&&fout.write((char*)&list.begin->data,
sizeof(DataType)))
            list.begin = list.begin->next;
        /*закрываем файловый поток вывода*/
        fout.close();
    };
}

void toLowerString(char* str)
{
    for(; *str; str++)
        *str = std::tolower(*str);
}

DataType readData()
{
    DataType data;
    char temp[10];
    std::cout << "ФИО: ";
    std::cin.getline(data.name, 78);
    data.name[0] = std::toupper(data.name[0]);
    std::cout << "Пол(М/Ж): ";
    std::cin.getline(temp, 9);
    data.gender = std::toupper(temp[0]);
    std::cout << "Вид спорта: ";
    std::cin.getline(data.sport, 32);
    toLowerString(data.sport);
    std::cout << "Год рождения: ";
    std::cin >> data.year;
    std::cout << "Рост: ";
    std::cin >> data.height;
    return data;
}

void print1(DataType *data)
{
    printf("ФИО:%s\nПол:%c\nВид спорта:%s\nГод:%d\nРост:%d\n",
    data->name, data->gender, data->sport, data->year, data->height);
}

void findMan(struct List list)
{
    DataType entry;
    int flag = 0;
    /*проходим по списку*/
    while(list.begin)
    {
        /*если нашли нужный элемент, то запоминаем это и сохраняем его в
entry*/
```

```
        if(list.begin->data.gender =='M'&&strcmp(list.begin-
>data.sport,"плавание")==0)
        {
            flag=1;
            entry = list.begin->data;
            break;
        };
        list.begin = list.begin->next;
    };
    /*если был найден элемент, то находим максимальный*/
    if(flag)
    {
        while(list.begin)
        {
            if(list.begin->data.gender == 'M' && list.begin->data.height >
entry.height && (strcmp(list.begin->data.sport,"плавание")==0))
                entry = list.begin->data;
            list.begin = list.begin->next;
        }
        print1(&entry);
    }
    else
        printf("Не найдено");
}

void findWomen(struct List list)
{
    /*создаем список*/
    struct List tmpList;
    makenull(&tmpList);
    while(list.begin)
    {
        /*находим подходящие элементы и добавляем их в новый список*/
        if((list.begin->data.gender == 'Ж') && (2021 - list.begin-
>data.year>=14) && (2021 - list.begin->data.year<=17))
            tmpList = addWithSort(tmpList, list.begin->data);
        list.begin = list.begin->next;
    }
    /*если были найдены подходящие элементы, то выводим список*/
    if(tmpList.begin)
    {
        printList(tmpList);
        /*удаляем темповый список*/
        tmpList = deleteList(tmpList);
    }
    else
        printf("Не найдено");

}

int main(int argc, char* files[])
{
    /*поддержка русского языка*/
    setlocale(LC_ALL, "rus");
    system("chcp 1251");
    system("cls");
    char filename[50];
    int menu, num;
    struct List list;
    /*если не было передачи имени файла через коммандную строку*/
    if(argc<2)
    {
        /*даем пользователю ввести имя файла*/
```

```
        puts ("Имя файла:");
        gets (filename);
    }
    else
        /*копируем переданное через коммандную строку имя файла в filename*/
        strcpy(filename,files[1]);
    /*читаем в список из файла*/
    list = readFileToList(filename);
    do
    {
        system ("CLS");
        puts ("1. Добавление записи");
        puts ("2. Просмотр записей");
        puts ("3. Просмотр записей в обратном порядке");
        puts ("4. Удаление записи");
        puts ("5. Найти мужчину");
        puts ("6. Найти женщин");
        puts ("7. Сохранить изменения и выйти");
        scanf ("%d%*c", &menu);
        system ("CLS");
        switch (menu)
        {
            case 1 :    list = addWithSort(list, readData()); break;
            case 2 :    printList(list); break;
            case 3 :    printListBack(list); break;
            case 4 :    std::cout << "Введите номер записи: ";
                        std::cin >> num;
                        if(num>0&&num<=sizeOfList(list))
                            list = deleteElementByNumber(list, num);
                        else
                            std::cout << "Неправильный ввод";
                        break;
            case 5 :    findMan(list); break;
            case 6 :    findWomen(list); break;
            case 7 :    break;
            default :   std::cout << "Неправильный ввод"; break;
        };
        getchar();
    } while (menu!=7);
    /*записываем из списка в файл*/
    writeListToFile(list, filename);
    /*удаляем список*/
    list = deleteList(list);
    return 0;
}
```

Результаты работы программы:

```
Имя файла:
1.dat
```

2. Просмотр записей

```
| N |              ФИО              | Пол | Вид спорта | Год | Рост |
-----------------------------------------------------------------------
|  1|Альков                         |М    |плавание    | 2001|   190|
-----------------------------------------------------------------------
|  2|Галкина                        |Ж    |баскетбол   | 2007|   150|
-----------------------------------------------------------------------
|  3|Иванов Иван Иванович           |М    |бокс        | 2002|   180|
-----------------------------------------------------------------------
|  4|Иванова                        |Ж    |бокс        | 2004|   180|
-----------------------------------------------------------------------
|  5|Машкина                        |Ж    |футбол      | 2005|   170|
-----------------------------------------------------------------------
|  6|Минкин                         |М    |бокс        | 2002|   178|
-----------------------------------------------------------------------
|  7|Михайлов                       |М    |плавание    | 2002|   181|
-----------------------------------------------------------------------
|  8|Ненастина Людмила Владимировна |Ж    |плавание    | 2005|   180|
-----------------------------------------------------------------------
|  9|Пискурев                       |М    |плавание    | 2002|   177|
-----------------------------------------------------------------------
| 10|Полежайкин                     |М    |танцы       | 1999|   177|
-----------------------------------------------------------------------
1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод
```

Вывести следующие 10 записей

```
| N |              ФИО              | Пол | Вид спорта | Год | Рост |
-----------------------------------------------------------------------
| 11|Рябцева                        |Ж    |волейбол    | 2006|   180|
-----------------------------------------------------------------------
| 12|Сидорова                       |Ж    |плавание    | 2003|   170|
-----------------------------------------------------------------------
| 13|Уманец                         |М    |волейбол    | 2001|   180|
-----------------------------------------------------------------------
1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод
```

Вывести следующие 10 записей

```
Следующих записей нет.
```

2. Просмотр записей

```
| N |              ФИО              | Пол | Вид спорта | Год | Рост |
-----------------------------------------------------------------------
|  1|Альков                         |М    |плавание    | 2001|   190|
-----------------------------------------------------------------------
|  2|Галкина                        |Ж    |баскетбол   | 2007|   150|
-----------------------------------------------------------------------
|  3|Иванов Иван Иванович           |М    |бокс        | 2002|   180|
-----------------------------------------------------------------------
|  4|Иванова                        |Ж    |бокс        | 2004|   180|
-----------------------------------------------------------------------
|  5|Машкина                        |Ж    |футбол      | 2005|   170|
-----------------------------------------------------------------------
|  6|Минкин                         |М    |бокс        | 2002|   178|
-----------------------------------------------------------------------
|  7|Михайлов                       |М    |плавание    | 2002|   181|
-----------------------------------------------------------------------
|  8|Ненастина Людмила Владимировна |Ж    |плавание    | 2005|   180|
-----------------------------------------------------------------------
|  9|Пискурев                       |М    |плавание    | 2002|   177|
-----------------------------------------------------------------------
| 10|Полежайкин                     |М    |танцы       | 1999|   177|
-----------------------------------------------------------------------
1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод
```

Вывести предыдущие 10 записей

```
Предыдущих записей нет.
```

3. Просмотр записей в обратном порядке

N	ФИО	Пол	Вид спорта	Год	Рост
1	Уманец	М	волейбол	2001	180
2	Сидорова	Ж	плавание	2003	170
3	Рябцева	Ж	волейбол	2006	180
4	Полежайкин	М	танцы	1999	177
5	Пискурев	М	плавание	2002	177
6	Ненастина Людмила Владимировна	Ж	плавание	2005	180
7	Михайлов	М	плавание	2002	181
8	Минкин	М	бокс	2002	178
9	Машкина	Ж	футбол	2005	170
10	Иванова	Ж	бокс	2004	180

```
1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод
```

Вывести следующие 10 записей

N	ФИО	Пол	Вид спорта	Год	Рост
11	Иванов Иван Иванович	М	бокс	2002	180
12	Галкина	Ж	баскетбол	2007	150
13	Альков	М	плавание	2001	190

```
1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод
```

5. Найти мужчину

```
ФИО:Альков
Пол:М
Вид спорта:плавание
Год:2001
Рост:190
```

1. Добавление записи

```
ФИО: Жилин
Пол(М/Ж): м
Вид спорта: плавание
Год рождения: 2001
Рост: 200
```

5. Найти мужчину

```
ФИО:Жилин
Пол:М
Вид спорта:плавание
Год:2001
Рост:200
```

6. Найти женщин

N	ФИО	Пол	Вид спорта	Год	Рост
1	Галкина	Ж	баскетбол	2007	150
2	Иванова	Ж	бокс	2004	180
3	Машкина	Ж	футбол	2005	170
4	Ненастина Людмила Владимировна	Ж	плавание	2005	180
5	Рябцева	Ж	волейбол	2006	180

```
1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод
```

4. Удаление записи

```
Введите номер записи: 1
```

2. Просмотр записей

```
| N |              ФИО              | Пол | Вид спорта | Год | Рост |
-----------------------------------------------------------------------
|  1|Галкина                        |Ж    |баскетбол   | 2007|   150|
-----------------------------------------------------------------------
|  2|Жилин                          |М    |плавание    | 2001|   200|
-----------------------------------------------------------------------
|  3|Иванов Иван Иванович           |М    |бокс        | 2002|   180|
-----------------------------------------------------------------------
|  4|Иванова                        |Ж    |бокс        | 2004|   180|
-----------------------------------------------------------------------
|  5|Машкина                        |Ж    |футбол      | 2005|   170|
-----------------------------------------------------------------------
|  6|Минкин                         |М    |бокс        | 2002|   178|
-----------------------------------------------------------------------
|  7|Михайлов                       |М    |плавание    | 2002|   181|
-----------------------------------------------------------------------
|  8|Ненастина Людмила Владимировна |Ж    |плавание    | 2005|   180|
-----------------------------------------------------------------------
|  9|Пискурев                       |М    |плавание    | 2002|   177|
-----------------------------------------------------------------------
| 10|Полежайкин                     |М    |танцы       | 1999|   177|
-----------------------------------------------------------------------
1. Вывести предыдущие 10 записей
2. Вывести следующие 10 записей
3. Прекратить вывод
```

Задача 2. Структура данных – дек с ограниченным входом.

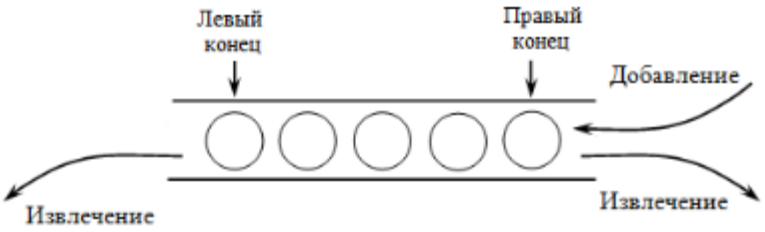
Уровень сложности – **повышенный**. Реализовать необходимую для решения задачи структуру данных с помощью двух структур хранения: векторной и связной, – реализацию оформить в виде иерархии классов. Абстрактный базовый класс должен задавать интерфейс, а производные – реализацию структуры данных. В программе, решающей поставленную задачу, выбор структуры хранения предоставить пользователю.

Напишите программу для моделирования работы конвейера по упаковке молока в бутылки. Бутылки, заполненные не менее номинального объема, закупориваются и отправляются на склад, отбракованные снимаются с конвейера, и молоко возвращается в цех розлива.

Пояснения к заданию. Начальный объем молока в баке, из которого происходит розлив, и номинальный объем бутылки задаются пользователем. В цикле, пока не закончится молоко в баке, производится наполнение бутылок с некоторой погрешностью и отправка их на ленту конвейера (в дек). Как только бутылка поставлена, она проверяется контролирующим автоматом и при недоливе снимается с конвейера (извлекается из дека с того же конца). Когда первая наполненная бутылка доезжает до автомата фасовки (дек полон), она снимается с ленты конвейера (извлекается с другого конца дека). Когда молоко в баке закончится, лента конвейера продолжает движение, пока все бутылки не будут отправлены на склад (пока дек не опустеет). Каждое действие (столько-то молока налито;

недолив – бутылка извлечена, молоко возвращено в бак; бутылка такого-то объема отправлена на склад; молоко в баке закончилось; все бутылки отправлены на склад) должно сопровождаться выводом соответствующего сообщения на экран.

Схематичное изображение структуры данных:



Схематичное изображение структуры хранения, использованной в программе:

Векторная структура хранения

1. Пуста

	data	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
left = 0									
right = 7									
maxsize = 8									

2. заполнена частично

	data	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
left = 0		1	2	3	4	5			
right = 4									
maxsize = 8									

	data	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
left = 4		9	2	3	4	5	6	7	8
right = 0									
maxsize = 8									

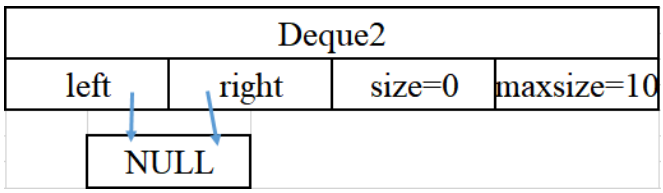
3. заполнена

	data	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
left = 0		1	2	3	4	5	6	7	
right = 6									
maxsize = 8									

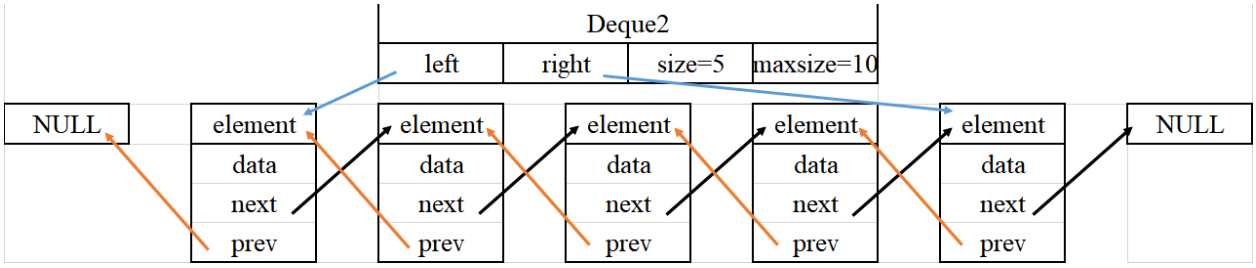
	data	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
left = 2		9	2	3	4	5	6	7	8
right = 0									
maxsize = 8									

Связная структура хранения

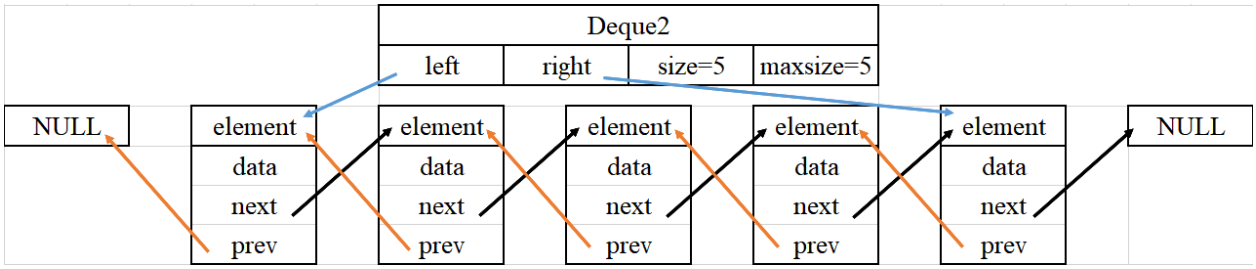
1. Пуста



2. Заполнена частично



3. Заполнена, по заданию требовалось не извлекать из дека, пока он не заполнится, поэтому из–за ограничения размера связная реализация была мной ограничена по возможному кол-ву элементов, а так ее размер ограничен только памятью системы.



Класс Menu

```
#include <iostream>
typedef double DataType;
class Menu
{
public:
    Menu(){};
    ~Menu(){};
    void print();
    /*virtual методы, значит, что эти методы будут определены в наследниках*/
    /*проверка на пустоту*/
    virtual int isEmpty()=0;
    /*проверка на заполненность*/
    virtual int isFull()=0;
    /*поместить в дек справа*/
    virtual int inRight (DataType x)=0;
    /*неразрушающее чтение слева*/
    virtual DataType readLeft()=0;
    /*неразрушающее чтение справа*/
    virtual DataType readRight()=0;
    /*извечь слева*/
    virtual DataType outLeft()=0;
    /*извечь справа*/
    virtual DataType outRight()=0;
};

void Menu::print()
{
    double milk, bottle, rand;
    std::cout << "Введите кол-во молока в баке: ";
```

```
    std::cin >> milk;
    std::cout << "Введите номинальный объем бутылки: ";
    std::cin >> bottle;
    if(bottle<0 || milk<0)
    {
        std::cout << "Неправильный ввод";
        return;
    };
    /*пока в баке достаточно молока для одной бутылки или конвеер не пуст*/
    while(milk>=bottle || !this->isEmpty())
    {
        /*вычисляем погрешность*/
        rand = (double)(std::rand()%21-10)/100+1;
        /*если дек не полон и молока в баке достаточно для бутылки и с
погрешностью*/
        if(!this->isFull() && milk>=bottle*rand)
        {
            /*выливем молоко из бака*/
            milk-=rand*bottle;
            /*ставим бытулку на конвеер*/
            this->inRight(rand*bottle);
            std::cout << "Налито " << this->readRight() << " молока\n";
            /*если в бутылке меньше молока номинального объема бутылки*/
            if(this->readRight() < bottle)
            {
                /*то выливаем молоко в бак, снимаем бутылку с конвеера*/
                milk+=this->outRight();
                std::cout <<"Недолив - бутылка извечена, молоко возвращено в
бак\n";
            }
            else
                /*если в баке недостаточно молока для бытылки*/
                if(milk < bottle)
                    std::cout << "В баке осталось "<<milk<<" молока\n";
        }
        else
            /*если конвеер, стек, пуст*/
            if(!this->isEmpty())
                std::cout << "Бутылка " << this->outLeft() << " объема
отправлена на склад\n";
    };
    std::cout << "Все бутылки отправлены на склад\n";
}
```

Класс Deque (векторная реализация)

```
#include <iostream>
typedef double DataType;
/*объявление класса, наследование от класса Menu*/
class Deque: public Menu
{
    /*индексы левого и правого элементов, размер дека*/
    int left, right, sizeOfDeque;
    /*массив элементов*/
    DataType* data;
public:
    /*конструктор с параметрами*/
    Deque(int size);
    /*деструктор*/
    ~Deque();
    int isEmpty();
    int isFull();
    int inRight (DataType x);
    DataType readLeft();
```

```
    DataType readRight();
    DataType outLeft();
    DataType outRight();
};

/*конструктор с параметрами, принимаем*/
Deque::Deque(int size)
{
    /*чтобы хранить n элементов в деке, нужен массив размером n+1*/
    sizeOfDeque = size+1;
    data = new DataType[sizeOfDeque];
    left = 0;
    right = sizeOfDeque-1;
}

Deque::~Deque()
{
    /*освобождение памяти*/
    delete[] data;
}

int Deque::isEmpty()
{
    return (right+1)%sizeOfDeque == left;
}

int Deque::isFull()
{
    return (right+2)%sizeOfDeque == left;
}

DataType Deque::readLeft()
{
    return data[left];
}

DataType Deque::readRight()
{
    return data[right];
}

int Deque::inRight(DataType x)
{
    if(this->isFull())
        return 0;
    right = (right+1)%sizeOfDeque;
    data[right] = x;
    return 1;
}

DataType Deque::outRight()
{
    int temp = right;
    /*если right != 0, то right = right - 1, если == 0, то right =
sizeOfDeque-1*/
    right = right ? right-1 : sizeOfDeque-1;
    return data[temp];
}

DataType Deque::outLeft()
{
    int temp = left;
    left = (left+1)%sizeOfDeque;
```

```
    return data[temp];
}
```

Класс Deque2 (связная реализация)

```
#include <iostream>
//#include "menu.cpp"
typedef double DataType;
struct element
{
    DataType data;
    struct element *next, *prev;
};
/*объявление класса, наследование от класса Menu*/
class Deque2 : public Menu
{
    /*указатели на левый и правый элементы дека*/
    struct element *left, *right;
    int sizeOfDeque, maxsize;
public:
    /*конструктор с параметрами*/
    Deque2(int size);
    /*деструктор*/
    ~Deque2();
    int isEmpty();
    int isFull();
    int inRight (DataType x);
    DataType readLeft();
    DataType readRight();
    DataType outLeft();
    DataType outRight();
};

Deque2::Deque2(int size)
{
    /*обнуляем указатели на границы дека, дек пуст*/
    left = right = NULL;
    sizeOfDeque = 0;
    maxsize = size;
}

Deque2::~Deque2()
{
    while (left)
        this->outLeft();
}

int Deque2::isEmpty()
{
    if(left)
        return 0;
    return 1;
}

int Deque2::isFull()
{
    return sizeOfDeque == maxsize;
}

int Deque2::inRight(DataType x)
{
    struct element *temp;
    if (!(temp = new struct element) || this->isFull())
        return 0;
```

```
    /*данные, которые хранит элемент*/
    temp->data = x;
    /*следующего элемента нет*/
    temp->next = NULL;
    /*предыдущий элемент правый элемент дека*/
    temp->prev = right;
    /*если дек не пуст, то следующий за правым элементом - добавляемый*/
    if(right)
        right->next = temp;
    else
        /*значит дек пуст, добавляемый элемент будет единственным, он и левый
тогда тоже*/
        left = temp;
    /*добавляемый элемент - новый правый элемент*/
    right = temp;
    /*увеличываем счетчик элементов дека*/
    sizeOfDeque++;
    return 1;
}

DataType Deque2::outRight()
{
    /*указатель на удаляемый элемент*/
    struct element *temp = right;
    /*сохраняем выходные данные*/
    DataType data = temp->data;
    /*правый элемент становится предыдущим правого*/
    right = right->prev;
    /*если в деке не осталось элементов справа, то дек пуст, обнуляем левый
указатль*/
    if (!right)
        left = NULL;
    /*очищаем память*/
    delete temp;
    /*уменьшаем счетчик элементов дека*/
    sizeOfDeque--;
    /*возвращаем данные*/
    return data;
}

DataType Deque2::outLeft()
{
    /*указатель на удаляемый элемент*/
    struct element *temp = left;
    /*сохраняем выходные данные*/
    DataType data = temp->data;
    /*левый элемент становится следующим  за левым*/
    left = left->next;
    /*если в деке не осталось элементов слева, то дек пуст, обнуляем правый
указатль*/
    if (!left)
        right = NULL;
    /*очищаем память*/
    delete temp;
    /*уменьшаем счетчик элементов дека*/
    sizeOfDeque--;
    /*возвращаем данные*/
    return data;
}

DataType Deque2::readRight()
{
    return right->data;
```

```
}

DataType Deque2::readLeft()
{
    return left->data;
}
```

Основная программа

```
#include "menu.cpp"
#include "deque1.cpp"
#include "deque2.cpp"
int main()
{
    system("chcp 1251");
    int choice;
    std::cout << "1. Векторный дек\n";
    std::cout << "2. Связной дек\n";
    std::cin >> choice;
    if(choice != 1 && choice != 2)
    {
        std::cout << "Неправильный ввод";
        return 0;
    };
    if (choice == 1)
    {
        Deque a(10);
        a.print();
    }
    else
    {
        Deque2 a(10);
        a.print();
    };
    return 0;
}
```

Результаты работы программы:

```
1. Векторный дек
2. Связной дек
1
Введите кол-во молока в баке: 10
Введите номинальный объем бутылки: 0.9
Налито 0.99 молока
Налито 0.882 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.927 молока
Налито 0.981 молока
Налито 0.963 молока
Налито 0.954 молока
Налито 0.918 молока
Налито 0.81 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.981 молока
Налито 0.99 молока
Налито 0.936 молока
Налито 0.855 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.927 молока
В баке осталось 0.433 молока
Бутылка 0.99 объема отправлена на склад
Бутылка 0.927 объема отправлена на склад
Бутылка 0.981 объема отправлена на склад
Бутылка 0.963 объема отправлена на склад
Бутылка 0.954 объема отправлена на склад
Бутылка 0.918 объема отправлена на склад
Бутылка 0.981 объема отправлена на склад
Бутылка 0.99 объема отправлена на склад
Бутылка 0.936 объема отправлена на склад
Бутылка 0.927 объема отправлена на склад
Все бутылки отправлены на склад
```

```
1. Векторный дек
2. Связной дек
2
Введите кол-во молока в баке: 10
Введите номинальный объем бутылки: 0.9
Налито 0.99 молока
Налито 0.882 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.927 молока
Налито 0.981 молока
Налито 0.963 молока
Налито 0.954 молока
Налито 0.918 молока
Налито 0.81 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.981 молока
Налито 0.99 молока
Налито 0.936 молока
Налито 0.855 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.927 молока
В баке осталось 0.433 молока
Бутылка 0.99 объема отправлена на склад
Бутылка 0.927 объема отправлена на склад
Бутылка 0.981 объема отправлена на склад
Бутылка 0.963 объема отправлена на склад
Бутылка 0.954 объема отправлена на склад
Бутылка 0.918 объема отправлена на склад
Бутылка 0.981 объема отправлена на склад
Бутылка 0.99 объема отправлена на склад
Бутылка 0.936 объема отправлена на склад
Бутылка 0.927 объема отправлена на склад
Все бутылки отправлены на склад
```

```
1. Векторный дек
2. Связной дек
1
Введите кол-во молока в баке: 2
Введите номинальный объем бутылки: 0.5
Налито 0.55 молока
Налито 0.49 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.515 молока
Налито 0.545 молока
В баке осталось 0.39 молока
Бутылка 0.55 объема отправлена на склад
Бутылка 0.515 объема отправлена на склад
Бутылка 0.545 объема отправлена на склад
Все бутылки отправлены на склад
```

```
1. Векторный дек
2. Связной дек
2
Введите кол-во молока в баке: 2
Введите номинальный объем бутылки: 0.5
Налито 0.55 молока
Налито 0.49 молока
Недолив - бутылка извечена, молоко возвращено в бак
Налито 0.515 молока
Налито 0.545 молока
В баке осталось 0.39 молока
Бутылка 0.55 объема отправлена на склад
Бутылка 0.515 объема отправлена на склад
Бутылка 0.545 объема отправлена на склад
Все бутылки отправлены на склад
```