Отчет по лабораторным работам 25-26

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: «28» августа 2022г.

Преподаватель: Титов В. К. каф. 806

Входной контроль знаний с оценкой _____

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

Подпись преподавателя _____

1 **Тема:** Автоматизация сборки программ модульной структуры на языке Си с использованием утилиты *make*. Абстрактные типы данных. Рекурсия. Модульное программирование на языке Си.

2 **Цель работы:** Изучить принципы работы утилиты *make*, отладить программу с помощью этой утилиты. Составить модуль определений и модуль реализации по заданной схеме модуля определений для абстрактного (пользовательского) типа данных (стека, очереди, списка или дека, в зависимости от варианта задания). Составить программный модуль, сортирующий экземпляр указанного абстрактного абстрактного типа данных заданным методом, используя только операции, импортированные из модуля UUDT.

3 **Задание** (*вариант №* 22)**:** *абстрактный тип данных:* дек, *метод сортировки:* вариант метода вставки, *вспомогательная процедура:* поиск в очереди, стеке или деке первого от начала элемента, который меньше своего предшественника. Если такой элемент найден, смещение его к началу до тех пор, пока он не станет первым или больше своего предшественника.

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)

Идея:

Согласно измененному заданию реализацию дека необходимо отобразить как на массив, так и на динамическую структуру.

Структура дека отображенного на массив состоит из целочисленных индексов первого и последнего элемента дека, целочисленного размера дека и указателя на, собственно, само тело дека (массив фиксированной длины).

Структура дека отображенного на динамическую состоит из целочисленного размера дека, а также ссылок на первый и последний его элементы. Элемент дека представляет из себя структуру, состоящую из значения элемента и ссылок на следующий и предыдущий элементы.

Заголовочный файл обязан содержать описание структур данных, используемых в файле реализации. Поэтому для разных реализаций дека придется использовать разные хедер-файлы. Причем эти файлы будут отличаться лишь в части описания вышеупомянутых структур, используемых для отображения дека. Объявление же процедур и функций будет полностью совпадать. Итак, в хедер-файле должны быть объявлены следующие процедуры и функции:

- Процедура создания дека (***Create***), будет принимать на вход указатель на неинициализированный дек;
- Функция проверки пустоты дека (***Empty***), принимает на вход копию дека, возвращает целое число (0 или 1);
- Процедура добавления элемента в начало дека (***PushFront***), принимает на вход указатель на дек и само значение вставляемого элемента;
- Процедура добавления элемента в конец дека (***PushBack***), принимает на вход указатель на дек и само значение вставляемого элемента;
- Функция удаления элемента из начала дека (***PopFront***), принимает на вход указатель на дек и возвращает значение удаленного элемента;
- Функция удаления элемента из конца дека (***PopFront***), принимает на вход указатель на дек и возвращает значение удаленного элемента;
- Функция прочтения верхнего элемента дека (***Top***), принимает на вход копию дека, возвращает значение первого элемента в деке;
- Функция вывода размера дека (***Size***), принимает на вход копию дека, возвращает целочисленное значение размера дека;
- Процедура печати содержимого дека (***Display***), принимает на вход копию дека;
- Процедура уничтожения дека (***Erase***), принимает на вход указатель на дек.

Опишем реализацию этих функций сразу для отображения на массив и динамическую структуру

*Процедура **Create***
Для реализации процедуры на массиве достаточно задать нулевое значение размеру дека и приравнять индексы первого и последнего элемента дека к индексам любых двух соседних элементов в массиве.
Для реализации процедуры на динамической структуре размеру дека также необходимо задать нулевое значение, а затем создать первый и последний элементы дека (оператором ***new***) и передвть им ссылки на друг друга.

*Функция **Empty***
В случае обоих реализаций достаточно вернуть результат сравнения размера дека с нулем.

*Процедура **PushFront***
При добавлении элементов в дек, реализованный на статическом массиве, необходимо проверять его переполнение, сравнивая размер дека с размером созданного массива. Чтобы добавить значение элемента в начало этого дека, достаточно присвоить это значение элементу массива с индексом первого элемента дека, а затем уменьшить значение индекса.
В случае реализации дека на динамической структуре проверять переполнение массива не нужно. Однако можно проверить переполнение кучи. Для этого во время создания нового элемента дека, нужно оператор присваивания засунуть в условный оператор.
После создания нового первого элемента дека старому первому элементу присваивается переданное значение. Затем эти элементы связываются указателями. После этого в переменную хранившую старый первый элемент дека записывается новый первый элемент и размер дека увеличивается на единицу.

*Процедура **PushBack*** работает подобно предыдущей, с той лишь разницей, что элемент добавляется в конец дека.

*Функция **PopFront***
В обеих реализациях первым делом функция проверяет наличие элементов в деке при помощи функции ***Empty***.
Затем для реализации на массиве индекс первого элемента дека увеличивается на единицу, размер дека уменьшается на единицу, и в конце выводится значение первого элемента дека,
В реализации на динамической системе ссылка на первый элемент заменяется ссылкой на элемент, идущий за ним (сам же элемент записывается в другую переменную и удаляется). Затем производится декремент размера дека и вывод значения нового первого элемента.

*Функция **PopBack*** выполняет те же действия, что и предыдущая, только для последнего элемента в деке.

*Функция **Top***
Проверяет пустой ли дек. Если он не пустой, возвращает значение элемента, идущего за первым в деке.

*Функция **Size***
Возвращает размер дека

*Процедура **Display***
Процедура пробегает в цикле по индексам массива или ссылкам элементов, начиная с первого, пока не встретит последний, печатая значения пройденных элементов.

Процедура ***Erase***
В случае реализации на статическом массиве процедура просто приравнивает размер дека к нулю и сдвигает индексы первого и последнего элемента к соседним элементам в массиве.
В случае с динамической структурой придется пробежать элементы массива, поочередно удаляя все, за исключением последнего и предпоследнего элемента, а затем приравнять размер дека к нулю.

Рассмотрим функции и процедуры программного модуля

*Процедура сортировки вставкой (**Procedure)***, принимает на вход указатель на дек.
Как было описано в задании эта процедура вспомогательная. С помощью нее необходимо реализовать сортировку дека.
Реализация процедуры будет основываться на трех циклах с условием. Первый цикл пробегает элементы начиная с первого,изымая их из основного дека функцией ***PopFront*** и складывая их процедурой ***Pushfront*** во вспомогательный дек до тех пор, пока не прервется их неубывающая последовательность. После окончания первого цикла из дека изымается первый элемент. Именно на нем прервалась последовательность, что означает, что он меньше предшественника. Этот элемент мы и будем сортировать. Во втором цикле из второго дека в первый дек перекладываются все элементы начиная с первого, большие чем найденный элемент, (так как мы складывали элементы в начало второго дека, сверху второго дека лежит наибольший элементы в последовательности) при помощи ***PopFront*** и ***PushFront***. После завершения второго цикла в начало дека добавим найденный элемент, а затем все остальные элементы из вспомогательного дека. В результате работы процедуры в начале дека будут собираться отсортированные элементы, что значит, что после n-ого применения процедуры весь дек будет отсортирован.

Опишем функцию проверки отсортированности дека
*Функция **isSorted***
Функция пробегает в цикле дек, складывая все элементы во вспомогательный, и изменяет значение флажка, если встречает элемент, меньший чем его предшественник. После этого во втором цикле перекладывает элементы обратно в основной дек и возвращает значение флажка (1 — если дек отсортирован, 0 — в противном случае)

*Функция сортировки дека **ShellSort**.*
По заданию сортировку дека необходимо провести одним из вариантов сортировки вставкой.
Метод бинарной вставки не имеет смысла для структуры дека. Метод обычной вставки требует проверки, отсортирован ли массив, после размещения каждого отсортировнного элемента. Либо же можно применять процедуру сортировки столько раз, сколько есть в деке элементов. Так или иначе, асимптотика такого алгоритма может быть эквивалентна $O(n^2)$.
Чтобы снизить цену сортировки воспользуемся сортировкой Шелла, которая является модификацией метода сортировки вставкой. Метод подразумевает разбиение сортируемого дека на деки поменьше, цена проверки отсортированности которых меньше. Если для обычной сортировки вставкой нам бы пришлось проверять отсортирован ли основной дек после каждого применения вспомогательной процедуры, то для сортировки Шелла нам необходимо будет проверить отсортированность целого дека только после объединения уже отсортированных его частей. Это должно ускорить работу алгоритма. Однако у такого метода сортировки есть свои минусы. Для обычной сортировки вставкой пространственная сложность является линейной, так как все действия выполняются непосредственно в основном деке и одном вспомогательном. Для сортировки Шелла придется выделить память для $s=n/2$ дополнительных деков, в которых будут происходить сортировки дека по частям. Эта проблема наиболее значительна для реализации дека на статическом массиве, где занимаемая структурой память не зависит от количества находящихся в ней элементов. Таким образом временная сложность алгоритма должна быть равна $O(n*logn)$, но пространственная сложность станет $O(n^2)$.
Опишем саму реализацию сортировки.
Алгоритм сортирует дек в цикле. Для каждой итерации цикла вычисляется свое $s=s/2$. Изначально же $s=n/2$, где n является размером изначального дека. До начала работы цикла при помощи оператора ***new*** конструируется массив из s деков, которые далее будут использованы для сортировки основного дека. Каждая итерация цикла выглядит так: элементы основного дека раскладываются по вспомогательным декам в массиве, используя правило $m=i\%s$, где m — номер вспомогательного дека в массиве, i — номер элемента в основном деке (если считать от первого элемента), затем каждый вспомогателный дек сортируется процедурой ***Procedure*** до тех пор, пока функция ***isSorted*** не скажет что этот дек отсортирован. После этого элементы из вспомогательных деков возвращаются в основной дек, причем вытаскиваются они процедурой ***PopFront*** по тому же правилу, по которому складывались во вспомогательные деки. То есть из каждого вспомогательного дека поочередно вытаскивается первый элемент и добавляется в конец основного дека. В конце работы процедуры оператором ***delete*** уничтожается массив вспомогательных деков.

Опишем также ***Makefile*** использованный для компиляции и отладки проекта, где проект реализован на динамической структуре.
В вершине графа помещен файл ***lab26.exe***, который и должен стать результатом компиляции проекта. Он компилируется на основе файлов ***lab26.o*** и **dynamic.o**, которые в свою очередь получаются в результате применения команд *g++ -c -o lab26.o lab26.cpp* и *g++ -o lab26.exe lab26.o dynamic.o*. Эти файлы являются вершинами подграфов, которые имеют один общий лист ***deque.h***.

Для реализации на массиве файл ***dynamic.o*** заменен файлом ***array.o***.

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

Реализация на массиве

Файл deque.h

```
#ifndef _DEQUE_H_
#define _DEQUE_H_
#define N 100
#define Tvalue int

struct deque
{ int first,last,size;
 Tvalue body[N];
} D;


void Create(deque &D);
int Empty(deque D);
void PushFront(deque &D, Tvalue V);
void PushBack(deque &D, Tvalue V);
Tvalue PopFront(deque &D);
Tvalue PopBack(deque &D);
Tvalue Top(deque D);
int Size(deque D);
void Display(deque D);
void Erase(deque &D);

#endif
```

Файл array.cpp

```
#include<stdio.h>
#define N 100
#define Tvalue int

struct deque
{ int first,last,size;
 Tvalue body[N];
};

void Create(deque &D)
{ D.first=D.size=0; D.last=N-1; }

int Empty(deque D)
{return D.size==0;}

void PushFront(deque &D, Tvalue V)
{ if(D.size==N) printf("Error: deque is overfolw\n");
   else {D.body[(D.first)?--D.first:(D.first=N-1)]=V;
D.size++;}
}
```

Реализация на динамической структуре

Файл deque.h

```
#ifndef _DEQUE_H_
#define _DEQUE_H_
#define N 100
#define Tvalue int

struct Item;
typedef Item* link;
struct Item
{ Tvalue data;
  link next;
  link prev;
};

struct deque
{ link last;
  link first;
  int size;
} D;

void Create(deque &D);
int Empty(deque D);
void PushFront(deque &D, Tvalue V);
void PushBack(deque &D, Tvalue V);
Tvalue PopFront(deque &D);
Tvalue PopBack(deque &D);
Tvalue Top(deque D);
int Size(deque D);
void Display(deque D);
void Erase(deque &D);

#endif
```

Файл dynamic.cpp

```
#include<stdio.h>
#define Tvalue int

struct Item;
typedef Item* link;
struct Item
{ Tvalue data;
  link next;
  link prev;
};

struct deque
{ link last;
  link first;
  int size;
};

void Create(deque &D)
{ D.first=new Item;
  D.last=new Item;
```

```
void PushBack(deque &D, Tvalue V)
{ if(D.size==N) printf("Error: deque is overflow\n");
   else {D.body[++D.last%=N]=V; D.size++;}
}

Tvalue PopFront(deque &D)
{ if(Empty(D)) printf("Error: deque is empty\n");
  else {D.size--; int i=D.first++; D.first%=N;
  return D.body[i];} return 0;
}

Tvalue PopBack(deque &D)
{ if(Empty(D)) printf("Error: deque is empty\n");
  else { D.size--; int i=D.last;
          (D.last)?D.last--:(D.last=N-1);
          return D.body[i];} return 0;
}


Tvalue Top(deque D)
{ if(Empty(D)) printf("Error: deque is empty\n");
  else return D.body[D.first]; return 0;
}

int Size(deque D){return D.size;}

void Display(deque D)
{ printf("[");
for(int i=D.first;i<D.first+D.size;i++)
 printf("%d%s",D.body[i%N],(i<D.first+D.size-
1)?",":"");
  printf("]\n");
}

void Erase(deque &D)
{ D.first=D.size=0; D.last=N-1;}
```

```
  D.first->next=D.last;
  D.last->prev=D.first;
  D.size=0;}


int Empty(deque D)
{ return D.size==0;}


void PushFront(deque &D, Tvalue V)
{ if(!(D.first->prev=new Item))
    printf("Error: deque is overflow\n");
  D.first->prev->next=D.first;
  D.first->data=V;
  D.first=D.first->prev;
  D.size++;
}


void PushBack(deque &D, Tvalue V)
{ if(!(D.last->next=new Item))
    printf("Error: deque is overflow\n");
  D.last->next->prev=D.last;
  D.last->data=V;
  D.last=D.last->next;
  D.size++;
}


Tvalue PopFront(deque &D)
{ if(Empty(D))
    printf("Error: deque is empty\n");
  else
  { link pi=D.first;
    D.first=D.first->next;
    D.size--;
    delete pi;
    return D.first->data;
  }
  return 0;
}


Tvalue PopBack(deque &D)
{ if(Empty(D))
    printf("Error: deque is empty\n");
  else
  { link pi=D.last;
    D.last=D.last->prev;
    D.size--;
    delete pi;
    return D.last->data;
  }
  return 0;
}


Tvalue Top(deque D)
{ if(Empty(D))
```

```
   { printf("Error: deque is empty\n");
     return 0;
   }
  return D.first->next->data;
}


int Size(deque D){return D.size;}


void Display(deque D)
{ printf("[");
  link s=D.first->next;
  while(s!=D.last)
  { printf("%d%s", s->data, (s->next!=D.last)?", ":"");
    s=s->next;
  }
  printf("]\n");
}


void Erase(deque &D)
{ while(D.first->next!=D.last)
  { link pi=D.first;
    D.first=D.first->next;
    delete pi;
  }
  D.size=0;
  printf("Deque is empty\n");
}
```

Файл lab26.cpp

```
#include<stdio.h>
#include"deque.h"


void Procedure(deque &D1)
{ deque D2; Tvalue V, W;
  if(!Empty(D1))
  { Create(D2);
    V=PopFront(D1);
    while(!Empty(D1)&&V<=Top(D1))
    { PushFront(D2,V);
      V=PopFront(D1);
    }
    if(!Empty(D1))
    { W=PopFront(D1);
      PushFront(D1,V);
      while(!Empty(D2)&&W<Top(D2))
      { V=PopFront(D2);
        PushFront(D1,V);
      }
      PushFront(D1,W);
    }
    else PushFront(D1,V);
    while(!Empty(D2))
    { V=PopFront(D2);
      PushFront(D1,V);
    }
  }
}
```

```
int isSorted(deque &D1)
{ deque D2; int flag=1;
  Create(D2);
  while(!Empty(D1))
  { int V=PopFront(D1);
    PushBack(D2, V);
    if(!Empty(D1)&&V>Top(D1))
       flag=0;
  }
  while(!Empty(D2))
  { int V=PopFront(D2);
    PushBack(D1, V);
  }
  return flag;
}


void ShellSort(deque &D)
{ int i, n, s; Tvalue V;
  n=D.size; s=1;
  while(s<n) s*=2;
  deque *m=new deque[s];
  for(i=0; i<s; i++)
    Create(m[i]);
  s=(s-1)/2;
  while(s>0)
  { for(i=0; i<n; i++)
    { V=PopFront(D);
      PushBack(m[i%s],V);
    }
    for(i=0; i<s; i++)
      { while(!isSorted(m[i]))
          Procedure(m[i]);
      }
    for(i=0; i<n; i++)
    { V=PopFront(m[i%s]);
      PushBack(D, V);
    }
    s=(s-1)/2;
  }
  delete m;
}

void menu()
{ printf("\n_____MENU_____\n"
        "1. Print deque\n"
        "2. Push elements in front\n"
        "3. Push elements in back\n"
        "4. Pop first element\n"
        "5. Pop last element\n"
        "6. Read top value\n"
        "7. Size of deque\n"
        "8. Sort deque\n"
        "9. Menu\n"
        "0. Exit\n"
        "================\n");
}

int main(){
  Create(D);
  int c=9;
  while(c)
  { if(c==1)
    { printf("Printing deque\n");
      Display(D);
    }
    else if(c==2)
    { printf("Input values\n");
      Tvalue V; scanf("%d", &V);
      while(V)
      { PushFront(D, V);
        scanf("%d", &V);
      }
      printf("Values was inserted\n");
    }
```

```
    else if(c==3)
    { printf("Input values\n");
      Tvalue V; scanf("%d", &V);
      while(V)
      { PushBack(D, V);
        scanf("%d", &V);
      }
      printf("Values was inserted\n");
    }
    else if(c==4)
    { Tvalue V=PopFront(D);
      if(V)
        printf("Value of first element is %d\n", V);
    }
    else if(c==5)
    { Tvalue V=PopBack(D);
      if(V)
        printf("Value of last element is %d\n", V);
    }
    else if(c==6)
    { Tvalue V=Top(D);
      if(V)
        printf("Value of top element is %d\n", V);
    }
    else if(c==7)
      printf("Size of deque is %d\n", Size(D));
    else if(c==8)
      { ShellSort(D);
        printf("Deque was sorted\n");
      }
    else if(c==9)
      menu();
    printf("Choose action\n");
    scanf("%d", &c);
  }
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/lab26/dynamic$ cat head
*********************************************************************
*                  Лабораторна работа №25-26                        *
*                       Утилита make                                *
*              Абстрактные типы данных. Рекурсия.                   *
*           Модульное программирование на языке Си.                 *
*              Выполнил: Тулин Иван Денисович                       *
*                  (номер по списку: 22)                            *
*                   Группа: М8О-101Б-21                             *
*********************************************************************
yusayu@YS:~/Рабочий стол/cppProjects/lab26/dynamic$ cat deque.h
#ifndef _DEQUE_H_
#define _DEQUE_H_
#define N 100
#define Tvalue int

struct Item;
typedef Item* link;
struct Item
{ Tvalue data;
  link next;
  link prev;
};

struct deque
{ link last;
  link first;
  int size;
} D;

void Create(deque &D);
int Empty(deque D);
void PushFront(deque &D, Tvalue V);
void PushBack(deque &D, Tvalue V);
Tvalue PopFront(deque &D);
Tvalue PopBack(deque &D);
Tvalue Top(deque D);
int Size(deque D);
void Display(deque D);
void Erase(deque &D);

#endif
yusayu@YS:~/Рабочий стол/cppProjects/lab26/dynamic$ cat dynamic.cpp
#include<stdio.h>
#define Tvalue int

struct Item;
typedef Item* link;
struct Item
{ Tvalue data;
  link next;
  link prev;
};

struct deque
{ link last;
  link first;
  int size;
};

void Create(deque &D)
{ D.first=new Item;
  D.last=new Item;
  D.first->next=D.last;
  D.last->prev=D.first;
  D.size=0;}


int Empty(deque D)
{ return D.size==0;}
```

```
void PushFront(deque &D, Tvalue V)
{ if(!(D.first->prev=new Item))
    printf("Error: deque is overflow\n");
  D.first->prev->next=D.first;
  D.first->data=V;
  D.first=D.first->prev;
  D.size++;
}


void PushBack(deque &D, Tvalue V)
{ if(!(D.last->next=new Item))
    printf("Error: deque is overflow\n");
  D.last->next->prev=D.last;
  D.last->data=V;
  D.last=D.last->next;
  D.size++;
}


Tvalue PopFront(deque &D)
{ if(Empty(D))
    printf("Error: deque is empty\n");
  else
  { link pi=D.first;
    D.first=D.first->next;
    D.size--;
    delete pi;
    return D.first->data;
  }
  return 0;
}


Tvalue PopBack(deque &D)
{ if(Empty(D))
    printf("Error: deque is empty\n");
  else
  { link pi=D.last;
    D.last=D.last->prev;
    D.size--;
    delete pi;
    return D.last->data;
  }
  return 0;
}


Tvalue Top(deque D)
{ if(Empty(D))
    { printf("Error: deque is empty\n");
      return 0;
    }
  return D.first->next->data;
}


int Size(deque D){return D.size;}


void Display(deque D)
{ printf("[");
  link s=D.first->next;
  while(s!=D.last)
  { printf("%d%s", s->data, (s->next!=D.last)?", ":"");
    s=s->next;
  }
  printf("]\n");
}
```

```
void Erase(deque &D)
{ while(D.first->next!=D.last)
  { link pi=D.first;
    D.first=D.first->next;
    delete pi;
  }
  D.size=0;
  printf("Deque is empty\n");
}

yusayu@YS:~/Рабочий стол/cppProjects/lab26/dynamic$ cat lab26.cpp
#include<stdio.h>
#include"deque.h"


void Procedure(deque &D1)
{ deque D2; Tvalue V, W;
  if(!Empty(D1))
  { Create(D2);
    V=PopFront(D1);
    while(!Empty(D1)&&V<=Top(D1))
    { PushFront(D2,V);
      V=PopFront(D1);
    }
    if(!Empty(D1))
    { W=PopFront(D1);
      PushFront(D1,V);
      while(!Empty(D2)&&W<Top(D2))
      { V=PopFront(D2);
        PushFront(D1,V);
      }
      PushFront(D1,W);
    }
    else PushFront(D1,V);
    while(!Empty(D2))
    { V=PopFront(D2);
      PushFront(D1,V);
    }
  }
}


int isSorted(deque &D1)
{ deque D2; int flag=1;
  Create(D2);
  while(!Empty(D1))
  { int V=PopFront(D1);
    PushBack(D2, V);
    if(!Empty(D1)&&V>Top(D1))
        flag=0;
  }
  while(!Empty(D2))
  { int V=PopFront(D2);
    PushBack(D1, V);
  }
  return flag;
}


void ShellSort(deque &D)
{ int i, n, s; Tvalue V;
  n=D.size; s=1;
  while(s<n) s*=2;
  deque *m=new deque[s];
  for(i=0; i<s; i++)
    Create(m[i]);
  s=(s-1)/2;
  while(s>0)
  { for(i=0; i<n; i++)
    { V=PopFront(D);
      PushBack(m[i%s],V);
    }
    for(i=0; i<s; i++)
```

```
    { while(!isSorted(m[i]))
        Procedure(m[i]);
    }
   for(i=0; i<n; i++)
   { V=PopFront(m[i%s]);
    PushBack(D, V);
   }
   s=(s-1)/2;
  }
  delete m;
}


void menu()
{ printf("\n_____MENU_____\n"
       "1. Print deque\n"
       "2. Push elements in front\n"
       "3. Push elements in back\n"
       "4. Pop first element\n"
       "5. Pop last element\n"
       "6. Read top value\n"
       "7. Size of deque\n"
       "8. Sort deque\n"
       "9. Menu\n"
       "0. Exit\n"
       "===============\n");
}

int main(){
  Create(D);
  int c=9;
  while(c)
  { if(c==1)
    { printf("Printing deque\n");
     Display(D);
    }
    else if(c==2)
    { printf("Input values\n");
     Tvalue V; scanf("%d", &V);
     while(V)
     { PushFront(D, V);
      scanf("%d", &V);
     }
     printf("Values was inserted\n");
    }
    else if(c==3)
    { printf("Input values\n");
     Tvalue V; scanf("%d", &V);
     while(V)
     { PushBack(D, V);
      scanf("%d", &V);
     }
     printf("Values was inserted\n");
    }
    else if(c==4)
    { Tvalue V=PopFront(D);
     if(V)
       printf("Value of first element is %d\n", V);
    }
    else if(c==5)
    { Tvalue V=PopBack(D);
     if(V)
       printf("Value of last element is %d\n", V);
    }
    else if(c==6)
    { Tvalue V=Top(D);
     if(V)
       printf("Value of top element is %d\n", V);
    }
    else if(c==7)
     printf("Size of deque is %d\n", Size(D));
    else if(c==8)
     { ShellSort(D);
      printf("Deque was sorted\n");
     }
```

```
    else if(c==9)
      menu();
    printf("Choose action\n");
    scanf("%d", &c);
  }
return 0;}
```

```
yusayu@YS:~/Рабочий стол/cppProjects/lab26/dynamic$ make
g++     -c -o dynamic.o dynamic.cpp
g++ -c -o lab26.o lab26.cpp
g++ -o lab26.exe lab26.o dynamic.o
yusayu@YS:~/Рабочий стол/cppProjects/lab26/dynamic$ ./lab26.exe

_____MENU_____
1. Print deque
2. Push elements in front
3. Push elements in back
4. Pop first element
5. Pop last element
6. Read top value
7. Size of deque
8. Sort deque
9. Menu
0. Exit
===============
Choose action
3
Input values
5 3 3 3 5 1 7 3 2 9 4 2 6 8 0
Values was inserted
Choose action
1
Printing deque
[5, 3, 3, 3, 5, 1, 7, 3, 2, 9, 4, 2, 6, 8]
Choose action
4
Value of first element is 5
Choose action
5
Value of last element is 8
Choose action
2
Input values
1
0
Values was inserted
Choose action
6
Value of top element is 1
Choose action
7
Size of deque is 13
Choose action
8
Deque was sorted
Choose action
1
Printing deque
[1, 1, 2, 2, 3, 3, 3, 3, 4, 5, 6, 7, 9]
Choose action
0
yusayu@YS:~/Рабочий стол/cppProjects/lab26/dynamic$ cd ..
yusayu@YS:~/Рабочий стол/cppProjects/lab26$ cd array/
yusayu@YS:~/Рабочий стол/cppProjects/lab26/array$ cat deque.h
#ifndef _DEQUE_H_
#define _DEQUE_H_
#define N 100
#define Tvalue int

struct deque
{ int first,last,size;
 Tvalue body[N];
} D;
```

```
void Create(deque &D);
int Empty(deque D);
void PushFront(deque &D, Tvalue V);
void PushBack(deque &D, Tvalue V);
Tvalue PopFront(deque &D);
Tvalue PopBack(deque &D);
Tvalue Top(deque D);
int Size(deque D);
void Display(deque D);
void Erase(deque &D);

#endif
yusayu@YS:~/Рабочий стол/cppProjects/lab26/array$ cat array.cpp
#include<stdio.h>
#define N 100
#define Tvalue int

struct deque
{ int first,last,size;
 Tvalue body[N];
};

void Create(deque &D)
{ D.first=D.size=0; D.last=N-1; }

int Empty(deque D)
{return D.size==0;}

void PushFront(deque &D, Tvalue V)
{ if(D.size==N) printf("Error: deque is overfolw\n");
   else {D.body[(D.first)?--D.first:(D.first=N-1)]=V; D.size++;}
}

void PushBack(deque &D, Tvalue V)
{ if(D.size==N) printf("Error: deque is overflow\n");
   else {D.body[++D.last%=N]=V; D.size++;}
}

Tvalue PopFront(deque &D)
{ if(Empty(D)) printf("Error: deque is empty\n");
  else {D.size--; int i=D.first++; D.first%=N;
  return D.body[i];} return 0;
}

Tvalue PopBack(deque &D)
{ if(Empty(D)) printf("Error: deque is empty\n");
  else { D.size--; int i=D.last;
 (D.last)?D.last--:(D.last=N-1);
 return D.body[i];} return 0;
}



Tvalue Top(deque D)
{ if(Empty(D)) printf("Error: deque is empty\n");
  else return D.body[D.first]; return 0;
}

int Size(deque D){return D.size;}

void Display(deque D)
{ printf("[");
for(int i=D.first;i<D.first+D.size;i++)
 printf("%d%s",D.body[i%N],(i<D.first+D.size-1)?",":"");
  printf("]\n");
}

void Erase(deque &D)
{ D.first=D.size=0; D.last=N-1;}



yusayu@YS:~/Рабочий стол/cppProjects/lab26/array$ cat lab26.cpp
#include<stdio.h>
```

```
#include"deque.h"


void Procedure(deque &D1)
{ deque D2; Tvalue V, W;
  if(!Empty(D1))
  { Create(D2);
    V=PopFront(D1);
    while(!Empty(D1)&&V<=Top(D1))
    { PushFront(D2,V);
      V=PopFront(D1);
    }
    if(!Empty(D1))
    { W=PopFront(D1);
      PushFront(D1,V);
      while(!Empty(D2)&&W<Top(D2))
      { V=PopFront(D2);
        PushFront(D1,V);
      }
      PushFront(D1,W);
    }
    else PushFront(D1,V);
    while(!Empty(D2))
    { V=PopFront(D2);
      PushFront(D1,V);
    }
  }
}


int isSorted(deque &D1)
{ deque D2; int flag=1;
  Create(D2);
  while(!Empty(D1))
  { int V=PopFront(D1);
    PushBack(D2, V);
    if(!Empty(D1)&&V>Top(D1))
        flag=0;
  }
  while(!Empty(D2))
  { int V=PopFront(D2);
    PushBack(D1, V);
  }
  return flag;
}


void ShellSort(deque &D)
{ int i, n, s; Tvalue V;
  n=D.size; s=1;
  while(s<n) s*=2;
  deque *m=new deque[s];
  for(i=0; i<s; i++)
    Create(m[i]);
  s=(s-1)/2;
  while(s>0)
  { for(i=0; i<n; i++)
    { V=PopFront(D);
      PushBack(m[i%s],V);
    }
    for(i=0; i<s; i++)
      { while(!isSorted(m[i]))
          Procedure(m[i]);
      }
    for(i=0; i<n; i++)
    { V=PopFront(m[i%s]);
      PushBack(D, V);
    }
    s=(s-1)/2;
  }
  delete m;
}
```

```
void menu()
{ printf("\n_____MENU_____\n"
       "1. Print deque\n"
       "2. Push elements in front\n"
       "3. Push elements in back\n"
       "4. Pop first element\n"
       "5. Pop last element\n"
       "6. Read top value\n"
       "7. Size of deque\n"
       "8. Sort deque\n"
       "9. Menu\n"
       "0. Exit\n"
       "===============\n");
}

int main(){
  Create(D);
  int c=9;
  while(c)
  { if(c==1)
    { printf("Printing deque\n");
      Display(D);
    }
    else if(c==2)
    { printf("Input values\n");
      Tvalue V; scanf("%d", &V);
      while(V)
      { PushFront(D, V);
        scanf("%d", &V);
      }
      printf("Values was inserted\n");
    }
    else if(c==3)
    { printf("Input values\n");
      Tvalue V; scanf("%d", &V);
      while(V)
      { PushBack(D, V);
        scanf("%d", &V);
      }
      printf("Values was inserted\n");
    }
    else if(c==4)
    { Tvalue V=PopFront(D);
      if(V)
        printf("Value of first element is %d\n", V);
    }
    else if(c==5)
    { Tvalue V=PopBack(D);
      if(V)
        printf("Value of last element is %d\n", V);
    }
    else if(c==6)
    { Tvalue V=Top(D);
      if(V)
        printf("Value of top element is %d\n", V);
    }
    else if(c==7)
      printf("Size of deque is %d\n", Size(D));
    else if(c==8)
      { ShellSort(D);
        printf("Deque was sorted\n");
      }
    else if(c==9)
      menu();
    printf("Choose action\n");
    scanf("%d", &c);
  }
return 0;}
yusayu@YS:~/Рабочий стол/cppProjects/lab26/array$ make
g++    -c -o lab26.o lab26.cpp
g++    -c -o array.o array.cpp
g++ -c -o lab26.o lab26.cpp
g++ -o lab26.exe lab26.o array.o
yusayu@YS:~/Рабочий стол/cppProjects/lab26/array$ ./lab26.exe
```

_____MENU_____
1. Print deque
2. Push elements in front
3. Push elements in back
4. Pop first element
5. Pop last element
6. Read top value
7. Size of deque
8. Sort deque
9. Menu
0. Exit
==========
Choose action
2
Input values
6 3 6 8 1 4 2 9 0
Values was inserted
Choose action
1
Printing deque
[9,2,4,1,8,6,3,6]
Choose action
6
Value of top element is 9
Choose action
5
Value of last element is 6
Choose action
3
Input values
8 0
Values was inserted
Choose action
4
Value of first element is 9
Choose action
7
Size of deque is 7
Choose action
8
Deque was sorted
Choose action
1
Printing deque
[1,2,3,4,6,8,8]
Choose action
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 использующие АТД и применять утилиту make

Недочёты при выполнении задания могут быть устранены следующим образом: _____

Подпись студента _____