МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

КАФЕДРА ТЕОРЕТИЧЕСКОЙ И ПРИКЛАДНОЙ ИНФОРМАТИКИ

Лабораторная работа №2
по дисциплине «Структуры данных и алгоритмы»

Факультет: ПМИ
Группа: ПМИ-03
Студенты: Сидоров Д.И., Малыгин С. А.
Преподаватель: Еланцева Е.Л.

НОВОСИБИРСК
2021

1)Условие задачи: Написать операции работы с заданной структурой данных, включив их в один модуль (файл). К основным операциям добавить операцию, показывающую содержимое структуры после выполнения какого-либо действия с ней. Эту операцию реализовать на основе базовых операций над динамическим деком. Написать программу, демонстрирующую выполнение операций
над заданной структурой данных. Эту программу надо поместить в свой модуль (файл). Модуль с основными операциями включать в программу, используя директиву include.

2)Анализ данных:

- *Входные данные:* последовательность символов, заданных с клавиатуры.
- *Выходные данные:* последовательность символов, выведенных на консоль.
- *Основные операции над динамическим деком:*

First– ввод первого элемента.

Метод решения: добавить элемент в дек, если дек пуст, иначе вывести “Dec is not empty”

PushBack- добавить элемент в конец списка.

Метод решения: если дек не пуст, добавить элемент в конец списка, иначе вызвать процедуру First

PushFront – добавить элемент в начало списка.

Метод решения: если дек не пуст, добавить элемент в начало списка, иначе вызвать процедуру First

PopBack – взятие последнего элемента в динамическом деке.

Метод решения: если дек не пуст, изменим входящую переменную на последний элемент и вернём true, иначе не будем изменять переменную и вернём false.

PopFront – взятие первого элемента в динамическом деке.

Метод решения: если дек не пуст, изменим входящую переменную на первый элемент и вернём true, иначе не будем изменять переменную и вернём false.

IsEmpty – проверка динамического дека на пустоту.

Метод решения: Если дек пуст, то вернуть True,иначе False

Clear – очистка динамического дека.

Метод решения: Если дек не пуст, очищаем дек.

Show_ALL– вывод динамического дека.

- *Метод решения:* Если дек не пуст, вывести все элементы, иначе вывести "Dec is empty"

3)Структура входных и выходных данных

Внешнее представление входных данных:

последовательность символов, заданных с клавиатуры.

Внутреннее представление входных и выходных данных:

Динамический дек реализован динамической структурой с использованием класса, содержащей элементы.

Линейный двунаправленный ациклический список без заглавного звена. Каждое звено списка реализовано структурой

```
struct List
	{
		List *prev = NULL;
		List *next = NULL;
		Char Data;
	};
```

Внешнее представление выходных данных:

последовательность символов, выведенных на консоль

4)Алгоритм

Класс DEC:

```
#include "DEC.h"
#include <iostream>

using namespace std;

namespace Program
{
	DEC::DEC(void)
	{
	}
	DEC::~DEC(void)
	{
	}

	Процедура DEC::First(char data)
	{
		Если (IsEmpty())
		{
			begin = new List;
```

```
		Вывод << "Enter first element: ";
		Ввод >> begin->Data;
		begin->next = NULL;
		begin->prev = NULL;
		end = begin;
	}
	Иначе
	{
		Вывод << "Dec is not empty" << endl;
	}
}
Процедура DEC::PushBack(символ data)
{
	Если (IsEmpty())
	{
		First(data);
	}
	Иначе
	{
		List* temp = new List;
		temp->Data = data;
		temp->next = NULL;
		temp->prev = end;
		end->next = temp;
		end = temp;
	}
}
Процедура DEC::PushFront(символ data)
{
	Если (IsEmpty())
	{
		First(data);
	}
	Иначе
	{
		List* temp = new List;
		temp->Data = data;
		temp->prev = NULL;
		temp->next = begin;
		begin->prev = temp;
		begin = temp;
	}
}
Функция DEC::PopBack(char& temp)
{
	Если (!IsEmpty())
	{

		символ data;
		Если (end->prev == NULL)
		{
			data = end->Data;
			begin = NULL;
		}
		Иначе
		{
			temp = end->Data;
			List *del = end;
			end = end->prev;
			Удалить del;
			end->next = NULL;
		}
		Возвращаем true;

	}
```

```
	Иначе
	{
		Возвращаем false;
	}

}

Функция DEC::PopFront(char& temp)
{
	Если (!IsEmpty())
	{

		символ temp = begin->Data;
		List* del = begin;
		begin = begin->next;
		Удалить del;
		begin->prev = NULL;

		Возвращаем true;
	}
	Иначе
	{
		Возвращаем false;
	}
}

Функция DEC::IsEmpty()
{
	Если (begin == NULL)
	{
		Возвращаем true;
	}
	Иначе
	{
		Возвращаем false;
	}
}

Процедура DEC::Clear()
{
	Если (!IsEmpty())
	{
		List* f = begin->next;
		List* fg = begin;
		Пока (f != NULL)
		{
			Удалить fg;
			fg = f;
			f = f->next;
		}
		Удалить f;
		begin = NULL;
	}
}

Процедура DEC::Show_ALL()
{
	Если (IsEmpty())
	{
		Вывод << endl << "Dec is empty" << endl;
	}
	Иначе
	{
		for (List* flag = begin; flag != NULL; flag = flag->next)
		{
```

```
                                Вывод << flag->Data;
                                Если (flag->next != NULL)
                                {
                                        Вывод << "->";
                                }
                        }
                        Вывод << endl;
                }
        }

}
```

Программа:

```
#include<iostream>
#include"DEC.h"

using namespace std;
using namespace Program;

Процедура Instruction()
{
        Вывод << "1-вывод дека" << endl;
        Вывод << "2-проверка на пустоту" << endl;
        Вывод << "3-добавление в конец" << endl;
        Вывод << "4-добавление в начало" << endl;
        Вывод << "5-очистка очереди" << endl;
        Вывод << "6-вывод первого элемента" << endl;
        Вывод << "7-вывод последнего элемента" << endl;
        Вывод << "0-завершение программы" << endl;
}

int main()
{
        setlocale(LC_ALL, "rus");

        DEC LIST;

        Логическая переменная flag = true;
        Символ temp(0);
        Целое число num;

        Вывод << "Введите значение первого элемента: ";
        Ввод >> num;
        LIST.First(num);
        Вывод << endl;

        Пока (flag)
        {
                Instruction();
                Вывод <<
"////////////////////////////////////////////////////////////////////" << endl;

                Вывод << endl << ">>";
                Ввод    >> temp;
                Вывод << endl;

                Оператор ветвления (temp)
                {
                case 1://вывод очереди

                        LIST.Show_ALL();
                        break;
                case 2://проверка на пустоту
```

```
                        Если (LIST.IsEmpty())
                        {
                                Вывод << "Дек - пуст" << endl;
                        }
                        Иначе
                        {
                                Вывод << "Дек - не пуст" << endl;
                        }
                        break;
                case 3://добавление в конец

                        Вывод << endl << "Введите значение последнего элемента: " << endl;
                        Ввод >> num;
                        LIST.PushBack(num);
                        break;
                case 4://добавление в начало

                        Вывод << endl << "Введите значение начального элемента: " << endl;
                        Ввод >> num;
                        LIST.PushFront(num);
                        break;
                case 5://очистка очереди

                        LIST.Clear();
                        break;
                case 6://вывод первого элемента
                        Если (LIST.PopFront(num))
                        {
                                Вывод << num << endl;
                        }
                        Иначе
                        {
                                Вывод << "Dec - пуст" << endl;
                        }
                        break;
                case 7://вывод последнего элемента
                        Если (LIST.PopBack(num))
                        {
                                Вывод << num << endl;
                        }
                        Иначе
                        {
                                Вывод << "Dec - пуст" << endl;
                        }
                        break;
                case 0://завершение программы
                         Вывод << "Завершение программы"<<endl;
                        flag = false;
                        break;
                 default:
                        Вывод << "Такой команды нет"<<endl;
                }

                system("pause");
                system("cls");
        }

        LIST.Clear();

        Возвращаем 0;
}
```

5)Структура программы:

1) Процедура ввод первого элемента:

```
First(char data)
```

Входные данные: `data` – первый элемент.

2) Процедура добавления элемента в конец списка:

```
PushBack(char data)
```

Входные данные: `data` – элемент, добавляемый в конец списка.

3) Процедура добавления элемента в начало списка:

```
PushFront(char data)
```

Входные данные: `data` – элемент, добавляемый в начало списка.

4) Функция взятия последнего элемента в динамическом деке:

```
PopBack(char& temp)
```

Входные данные: `temp` – элемент, значение которого изменится на значение элемента из конца списка.

Выходные данные: `true` – если Дек не пуст, `false` – если Дек пуст.

5) Функция взятия первого элемента в динамическом деке:

```
PopFront(char& temp)
```

Входные данные: `temp` – элемент, значение которого изменится на значение элемента из начала списка.

Выходные данные: `true` – если Дек не пуст, `false` – если Дек пуст.

6) Функция проверки динамического дека на пустоту:

```
IsEmpty()
```

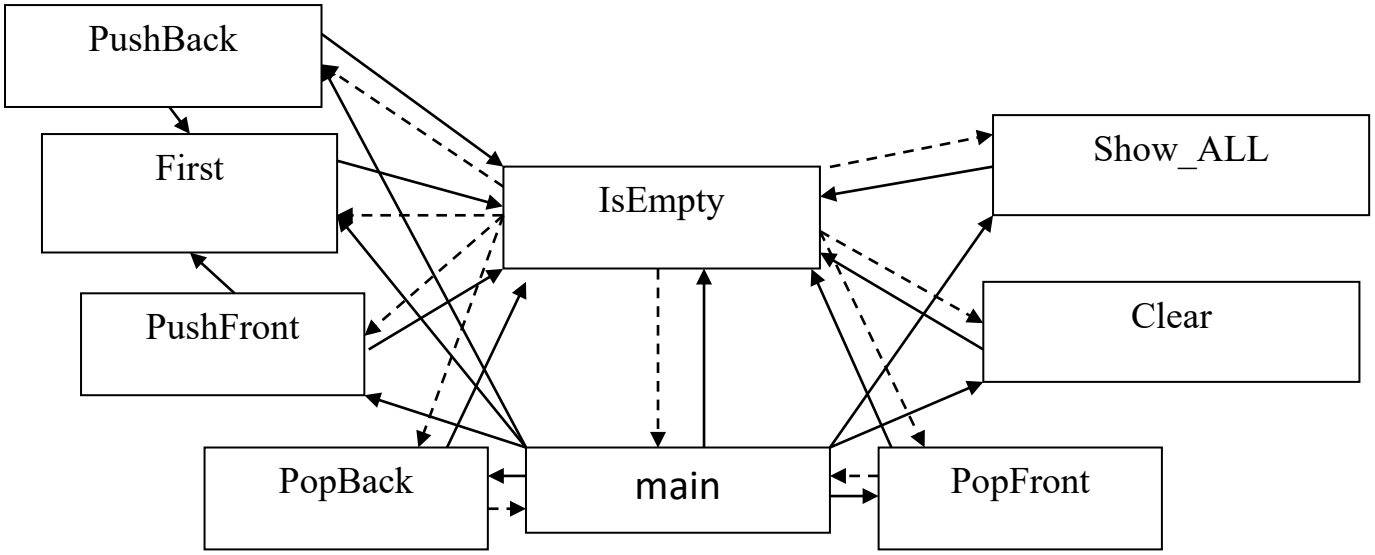
Выходные данные: False или True

7) Процедура очистки динамического дека:

```
Clear()
```

8) Процедура вывода динамического дека:

```
Show_ALL()
```


PushBack
First
PushFront
IsEmpty
Show_ALL
Clear
PopBack
main
PopFront

6)Текст программы:

Класс DEC:

```
#include "DEC.h"
#include<iostream>

using namespace std;

namespace Program
{
	DEC::DEC(void)
	{
	}
	DEC::~DEC(void)
	{
	}

	void DEC::First(char data)
	{
		if (IsEmpty())
		{
			begin = new List;
			begin->Data = data;
			begin->next = NULL;
			begin->prev = NULL;
			end = begin;
		}
	}
	void DEC::PushBack(char data)
	{
		if (IsEmpty())
		{
			First(data);
		}
		else
		{
			List* temp = new List;
			temp->Data = data;
			temp->next = NULL;
			temp->prev = end;
			end->next = temp;
			end = temp;
		}
	}
	void DEC::PushFront(char data)
	{
		if (IsEmpty())
		{
			First(data);
		}
		else
		{
			List* temp = new List;
			temp->Data = data;
			temp->prev = NULL;
			temp->next = begin;
			begin->prev = temp;
			begin = temp;
		}
	}
	bool DEC::PopBack(char& temp)
	{
		if (!IsEmpty())
		{
```

```
		if (end->prev == NULL)
		{
			temp = end->Data;
			begin = NULL;
		}
		else
		{
			temp = end->Data;
			List* del = end;
			end = end->prev;
			delete del;
			end->next = NULL;
		}

		return true;
	}
	else
	{
		return false;
	}
}
bool DEC::PopFront(char& temp)
{
	if (!IsEmpty())
	{
		if (begin->next == NULL)
		{
			temp = begin->Data;
			begin = NULL;
		}
		else
		{
		temp = begin->Data;
		List* del = begin;
		begin = begin->next;
		delete del;
		begin->prev = NULL;
		}

		return true;
	}
	else
	{
		return false;
	}
}
bool DEC::IsEmpty()
{
	if (begin == NULL)
	{
		return true;
	}
	else
	{
		return false;
	}
}
void DEC::Clear()
{
	if (!IsEmpty())
	{
		List* f = begin->next;
		List* fg = begin;
		while (f != NULL)
		{
```

```
                        delete fg;
                        fg = f;
                        f = f->next;
                }
                delete f;
                begin = NULL;
        }
}
void DEC::Show_ALL()
{
        if (IsEmpty())
        {
                cout << endl << "Dec is empty" << endl;
        }
        else
        {
                for (List* flag = begin; flag != NULL; flag = flag->next)
                {
                        cout << flag->Data;
                        if (flag->next != NULL)
                        {
                                cout << "->";
                        }
                }
                cout << endl;
        }
}
}
```

Программа:

```
#include<iostream>
#include"DEC.h"

using namespace std;
using namespace Program;

void Instruction()
{
        cout << "1-вывод дека" << endl;
        cout << "2-проверка на пустоту" << endl;
        cout << "3-добавление в конец" << endl;
        cout << "4-добавление в начало" << endl;
        cout << "5-очистка очереди" << endl;
        cout << "6-вывод первого элемента" << endl;
        cout << "7-вывод последнего элемента" << endl;
        cout << "0-завершение программы" << endl;
}

int main()
{
        setlocale(LC_ALL, "rus");

        DEC LIST;

        bool flag = true;
        int temp(0);
        char num;

        cout << "Введите первый элемент: ";
        cin >> num;
        LIST.First(num);
        cout << endl;

        while (flag)
```

```
	{
		Instruction();
		cout <<
"////////////////////////////////////////////////////////////////////////" << endl;

		cout << endl << ">>";
		cin >> temp;
		cout << endl;

		switch (temp)
		{
		case 1://вывод очереди

			LIST.Show_ALL();
			break;
		case 2://проверка на пустоту

			if (LIST.IsEmpty())
			{
				cout << "Dec - пуст" << endl;
			}
			else
			{
				cout << "Dec - не пуст" << endl;
			}
			break;
		case 3://добавление в конец

			cout << endl << "Введите значение последнего элемента: " << endl;
			cin >> num;
			LIST.PushBack(num);
			break;
		case 4://добавление в начало

			cout << endl << "Введите значение начального элемента: " << endl;
			cin >> num;
			LIST.PushFront(num);
			break;
		case 5://очистка очереди

			LIST.Clear();
			break;
		case 6://вывод первого элемента
			if (LIST.PopFront(num))
			{
				cout << num << endl;
			}
			else
			{
				cout << "Dec - пуст" << endl;
			}
			break;
		case 7://вывод последнего элемента
			if (LIST.PopBack(num))
			{
				cout << num << endl;
			}
			else
			{
				cout << "Dec - пуст" << endl;
			}
			break;
		case 0://завершение программы

			flag = false;
```

```
                        break;
                default:
                        cout << "Такой команды - нет" << endl;
                }

                system("pause");
                system("cls");
        }

        LIST.Clear();

        return 0;
}
```

7)Тесты:

№	**Ввод/вывод программы**
1	Вывод: Введите первый элемент: Ввод: 1 Вывод: 1-вывод дека 2-проверка на пустоту 3-добавление в конец 4-добавление в начало 5-очистка очереди 6-вывод первого элемента 7-вывод последнего элемента 0-завершение программы // Ввод: 3 Вывод: Введите значение последнего элемента: Ввод: 4 Вывод: //Справка Ввод: 4 Вывод: Введите значение начального элемента: Ввод: 2

Вывод:
//Справка
Ввод:
1
Вывод:
2->1->4
Вывод:
//Справка
Ввод:
6
Вывод:
2
Вывод:
//Справка
Ввод:
3
Вывод:
Введите значение последнего элемента:
Ввод:
7
Вывод:
//Справка
Ввод:
7
Вывод:
7
Вывод:
//Справка
Ввод:
2
Вывод:
Дек - не пуст
Вывод:
//Справка
Ввод:
5
Вывод:
//Справка
Ввод:
1
Вывод:
Dec is empty
Вывод:
//Справка
Ввод:

	9 Вывод: Завершение программы
2	Вывод: Введите первый элемент: Ввод: 1 Вывод: 1-вывод дека 2-проверка на пустоту 3-добавление в конец 4-добавление в начало 5-очистка очереди 6-вывод первого элемента 7-вывод последнего элемента 0-завершение программы // Ввод: 33 Вывод: Такой команды нет
3	Вывод: Введите первый элемент: Ввод: 1 Вывод: 1-вывод дека 2-проверка на пустоту 3-добавление в конец 4-добавление в начало 5-очистка очереди 6-вывод первого элемента 7-вывод последнего элемента 0-завершение программы // Ввод: 0 Вывод: Завершение программы

8) *Результат работы программы*:

Программа работает правильно, что подтверждают тесты.