Министерство науки и высшего образования РФ ФГБОУ ВПО

«Пермский государственный национальный исследовательский университет»

Кафедра прикладной математики и информатики

**Индивидуальное задание №2**

**«Очередь»**

***по дисциплине «Алгоритмизация и программирование-II»***

|  |  |
| --- | --- |
|  | **Выполнил:** |
|  | Студентка 1-го курса механико-математического факультета  Черткова С.М., группа ПМИ-1,2-2021 |
|  | **Проверил:** |
|  | Старший преподаватель кафедры прикладной математики и информатики ПГНИУ,  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шеина Т. Ю. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_ 2022 г. |

Пермь 2022

# Постановка задачи

Реализуйте структуру данных "очередь". Напишите программу, содержащую описание очереди и моделирующую работу очереди, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку.

Возможные команды для программы:

push n Добавить в очередь число n (значение n задается после команды). Программа должна вывести ok.

pop Удалить из очереди первый элемент. Программа должна вывести его значение.

front Программа должна вывести значение первого элемента, не удаляя его из очереди.

size Программа должна вывести количество элементов в очереди.

clear Программа должна очистить очередь и вывести ok.

exit Программа должна вывести bye и завершить работу.

Перед исполнением операций front и pop программа должна проверять, содержится ли в очереди хотя бы один элемент. Если во входных данных встречается операция front или pop, и при этом очередь пуста, то программа должна вместо числового значения вывести строку error.

Входные данные: Вводятся команды управления очередью, по одной на строке

Выходные данные: Требуется вывести протокол работы очереди, по одному сообщению на строке

# Определение идеи алгоритма,

# выбор методов решения и структур данных

Для решения задачи использовалась среда разработки Microsoft Visual Studio и язык программирования C++.

Очередь – это линейный список, элементы которого удаляются из начала списка, а добавляются в конце списка. Принцип работы очереди: «первый пришел - первый ушел» (FIFO – First Input First Output). Для работы с этим списком надо объявить два новых типа данных – узел списка и указатель на него. Также создается новый тип PNode, который необходим для описания дополнительных указателей. Head – указывает на начало списка, LastNode – на последний элемент.

Программа считывает из предложенного списка команду, которую необходимо выполнить. И в зависимости от выбора, выполняет то или иное действие:

1. push n: проверяем, пустая ли очередь. Если это так, то выделяется память под новый элемент, который добавляется в начало очереди. Иначе новый элемент Temp добавляется в конец очереди, после элемента LastNode. После последнему (новому) элементу присваивается значение, а также увеличивается размер очереди. Программа выводит ok.
2. pop: проверяем, пустая ли очередь. Если это так, то выводится ошибка. Иначе выводится первый элемент очереди, на который указывает Head, после чего он удаляется. Размер очереди уменьшается.
3. front: проверяем, пустая ли очередь. Если это так, то выводится ошибка. Иначе выводится первый элемент очереди, на который указывает Head.
4. size: в начале программы переменной присваивается значение «0». При выполнении команды push n размер увеличивается, pop – уменьшается, clear – обнуляется. Программа выводит количество элементов в очереди.
5. clear: выполняется проход по очереди с удалением просматриваемого элемента MyNode. Размер очереди обнуляется. Программа выводит ok.
6. exit: программа выводит bye и завершает работу. В начале программы переменной логического типа присваивается значение true, после выполнения команды exit – false, так выполняется выход из цикла.

# Тестирование программы

Тест 1.

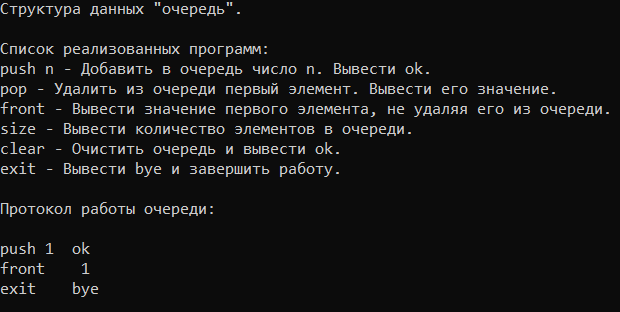
*ВХОДНЫЕ ДАННЫЕ / ОЖИДАЕМЫЙ РЕЗУЛЬТАТ:*

push 1 ok

front 1

exit bye

*РЕЗУЛЬТАТ, ПОЛУЧЕННЫЙ ПРИ ТЕСТИРОВАНИИ ПРОГРАММЫ:*

**

Тест 2.

*ВХОДНЫЕ ДАННЫЕ / ОЖИДАЕМЫЙ РЕЗУЛЬТАТ:*

size 0

push 1 ok

size 1

push 2 ok

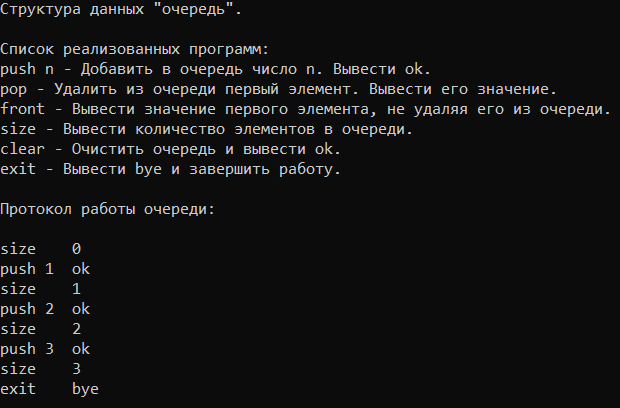
size 2

push 3 ok

size 3

exit bye

*РЕЗУЛЬТАТ, ПОЛУЧЕННЫЙ ПРИ ТЕСТИРОВАНИИ ПРОГРАММЫ:*

**

Тест 3.

*ВХОДНЫЕ ДАННЫЕ / ОЖИДАЕМЫЙ РЕЗУЛЬТАТ:*

push 8 ok

push 5 ok

size 2

clear ok

pop error

front error

push 4 ok

push 7 ok

front 4

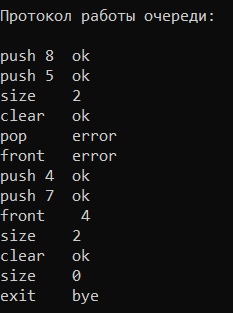
size 2

clear ok

size 0

exit bye

*РЕЗУЛЬТАТ, ПОЛУЧЕННЫЙ ПРИ ТЕСТИРОВАНИИ ПРОГРАММЫ:*

**

Тест 4.

*ВХОДНЫЕ ДАННЫЕ / ОЖИДАЕМЫЙ РЕЗУЛЬТАТ:*

push 5 ok

push 4 ok

push 3 ok

push 2 ok

push 1 ok

front 5

pop 5

size 4

pop 4

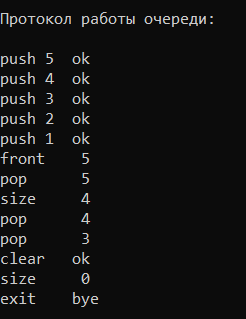
pop 3

clear ok

size 0

exit bye

*РЕЗУЛЬТАТ, ПОЛУЧЕННЫЙ ПРИ ТЕСТИРОВАНИИ ПРОГРАММЫ:*

**

Тест 5.

*ВХОДНЫЕ ДАННЫЕ / ОЖИДАЕМЫЙ РЕЗУЛЬТАТ:*

push 6 ok

push 6 ok

push 6 ok

size 3

front 6

pop 6

size 2

clear ok

front error

pop error

push 6 ok

pop 6

pop error

push 5 ok

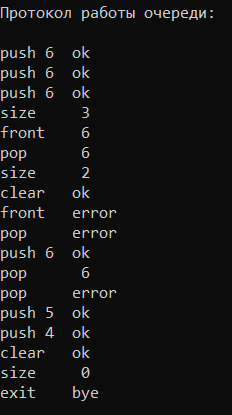
push 4 ok

clear ok

size 0

exit bye

*РЕЗУЛЬТАТ, ПОЛУЧЕННЫЙ ПРИ ТЕСТИРОВАНИИ ПРОГРАММЫ:*

**

# Текст программы

Исходный файл программы *ind2 Chertkova\_Sofya.cpp* прилагается к работе в электронном виде, ниже приводим его полностью.

#include <iostream>

#include <windows.h>

#include <string>

#include <iomanip>

using namespace std;

struct Node

{

int data; //информационный элемент

Node\* Next; //указатель на следующий элемент

};

typedef Node\* PNode;

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string komanda; //

int n, size = 0; //n - число, которое нужно добавить в очередь, size - размер очереди

bool flag = true; //flag - переменная, которое позволяет выйти из программы

PNode Head, LastNode; //Head - голова, LastNode - последний элемент

Head = LastNode = NULL;

cout << "Структура данных \"очередь\".";

cout << "\n\nСписок реализованных программ:\npush n - Добавить в очередь число n. Вывести ok.\npop - Удалить из очереди первый элемент. Вывести его значение. \

\nfront - Вывести значение первого элемента, не удаляя его из очереди.\nsize - Вывести количество элементов в очереди. \nclear - Очистить очередь и вывести ok. \

\nexit - Вывести bye и завершить работу.\n";

cout << "\nПротокол работы очереди: \n\n";

while (flag)

{

cin >> komanda; //ввод команды, которую необходимо выполнить

cout << "\033[F"; //удаление строки ввода

if (komanda == "push")

{

cin >> n; //ввод числа, которое необходимо добавить

PNode Temp; //Temp - новый элемент, LastNode - последний элемент

if (Head == NULL) //если очередь пустая, добавляем новый элемент в начало очереди

{

Head = new Node;

LastNode = Head;

Head->Next = NULL;

}

else //если очередь не пустая, добавляем новый элемент после последнего (конец очереди)

{

Temp = new Node;

LastNode->Next = Temp;

LastNode = Temp;

LastNode->Next = NULL;

}

LastNode->data = n; //присвоение новому элементу значения n

cout << "push " << n << setw(5) << "ok\n";

size++; //увеличение размера очереди

}

if (komanda == "pop")

{

if (Head == NULL) cout << "pop" << setw(11) << "error\n"; //если очередь пустая, то ошибка

else

{

cout << "pop" << setw(7) << Head->data << endl; //вывод первого элемента очереди

PNode OldNode = Head;

Head = OldNode->Next;

delete OldNode; //освобождение памяти первого элемента

size--; //уменьшение размера очереди

}

}

if (komanda == "front")

{

if (Head == NULL) cout << "front" << setw(9) << "error\n"; //если очередь пустая, то ошибка

else cout << "front " << setw(4) << Head->data << endl; //вывод первого элемента очереди

}

if (komanda == "size")

{

cout << "size" << setw(6) << size << endl; //размер очереди

}

if (komanda == "clear")

{

PNode MyNode = Head;

while (Head != NULL) //проход по очереди

{

MyNode = Head;

Head = Head->Next;

delete MyNode; //освобождение памяти

}

cout << "clear" << setw(6) << "ok\n";

size = 0; //обнуление размера очереди

}

if (komanda == "exit")

{

cout << "exit" << setw(8) << "bye\n"; //выход из программы

flag = false; //позволяет завершить работу программы

}

}

}