**Постановка задачи.**

Реализовать очередь на основе связного списка на с++.

**Введение.**

Очередь в программировании - это структура данных, которая используется для хранения элементов в порядке их поступления и обработки их в том же порядке. Очередь имеет два основных метода в своем интерфейсе: добавление элемента в конец очереди и удаление элемента из начала очереди. [Очередь работает по принципу FIFO (First In First Out), то есть первый элемент, добавленный в очередь, будет первым удаленным](https://habr.com/ru/articles/483944/).

**Решение.**

#include <iostream>

using namespace std;

class LinkedListNode {

public:

int data;

LinkedListNode\* next;

};

class LinkedList {

public:

LinkedListNode\* head;

LinkedList() {

head = NULL;

}

// Добавление элемента в конец списка

void AddElementAtEnd(int x) {

LinkedListNode\* current = new LinkedListNode;

current->data = x;

current->next = NULL;

if (head == NULL) {

head = current;

return;

}

LinkedListNode\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = current;

}

//Данный метод необходимо немного изменить, чтобы вместе с удаление последнего элемента он возвращад его.

// Удаление первого элемента списка

int RemoveFirstElement() {

if (head == NULL) {

cout << "List is empty" << endl;

return 0;

}

LinkedListNode\* temp = head;

int toReturn = temp->data;

head = head->next;

delete temp;

return toReturn;

}

//Печать списка

void PrintList() {

if (head == NULL) {

cout << "List is empty" << endl;

return;

}

else {

LinkedListNode\* current = head;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

}

};

class queue {

public:

LinkedList listInsideQueue;

//Вставка в конец

void enter(int x) {

listInsideQueue.AddElementAtEnd(x);

}

//Удаланение и получение последнего элемента.

int leave() {

return listInsideQueue.RemoveFirstElement();

}

//Печать очереди

void printQueue() {

listInsideQueue.PrintList();

}

};

int main() {

// Создание объекта очереди

queue exmapleQueue;

// Добавление элементов в конец очереди

exmapleQueue.enter(1);

exmapleQueue.enter(2);

exmapleQueue.enter(3);

exmapleQueue.enter(4);

//Печать очереди

exmapleQueue.printQueue();

// Удаление элементов из начала очереди

exmapleQueue.leave();

// Удаление элементов из начала очереди с получение последнего элемента.

cout<<exmapleQueue.leave()<<endl;

//Печать очереди

exmapleQueue.printQueue();

}

C:\Users\Максим\Downloads\2023-06-06_18-08-20.png

**Заключение.**

В результате выполнения задачи была создана очередь, реализованная поверх связного списка. Эта структура данных позволяет эффективно добавлять элементы в конец очереди и удалять элементы из ее начала. В итоге, благодаря данному коду, создание и использование очереди в программе стало гораздо удобнее и проще.