Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

­­­­­

Основы алгоритмизации и программирования

Отчет по лабораторной работе №12

«Очереди»

Выполнил: Усов А.М.

Студент группы 310901

Преподаватель: Василькова А. Н.

Минск 2023

Цель: сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием очередей.

# Задание 4. Система состоит из трех процессоров P1, P2, P3, очереди F, стека S и распределителя задач R. В систему поступают запросы на выполнение задач трёх типов – T1, T2 и T3, каждая для своего процессора. Поступающие запросы ставятся в очередь. Если в начале очереди находится задача Ti и процессор Pi свободен, то распределитель R ставит задачу на выполнение в процессор Pi, а если процессор Pi занят, то распределитель R отправляет задачу в стек и из очереди извлекается следующая задача. Если в вершине стека находится задача, процессор которой в данный момент свободен, то эта задача извлекается и отправляется на выполнение.

#include <iostream>

#include <time.h>

using namespace std;

template <typename T>

// двусвязанный спысок

class LinkedList {

private:

class Node {

private:

T data;

int index;

Node\* next;

Node\* prev;

public:

Node(T data, int index, Node\* next = nullptr, Node\* prev = nullptr) {

this->data = data;

this->next = next;

this->prev = prev;

this->index = index;

}

T getData() {

return data;

}

Node\* getNext() {

return next;

}

Node\* getPrev() {

return prev;

}

void setData(T data) {

this->data = data;

}

void setNext(Node\* next) {

this->next = next;

}

void setPrev(Node\* prev) {

this->prev = prev;

}

int getIndex() {

return index;

}

};

Node\* head;

Node\* tail;

int size;

public:

LinkedList() {

head = nullptr;

tail = nullptr;

size = 0;

}

void push\_first(T data) {

Node\* temp = new Node(data,size);

if (head == nullptr) {

head = temp;

tail = temp;

}

else {

temp->setNext(head);

head->setPrev(temp);

head = temp;

}

size++;

}

void push\_back(T data) {

Node\* temp = new Node(data,size);

if (tail == nullptr) {

head = temp;

tail = temp;

}

else {

temp->setPrev(tail);

tail->setNext(temp);

tail = temp;

}

size++;

}

int GetSize() {

return size;

}

void Delete(T data) {

Node\* temp = head;

while (temp != nullptr) {

if (temp->getData() == data) {

if (temp == head) {

head = head->getNext();

head->setPrev(nullptr);

}

else if (temp == tail) {

tail = tail->getPrev();

tail->setNext(nullptr);

}

else {

temp->getPrev()->setNext(temp->getNext());

temp->getNext()->setPrev(temp->getPrev());

}

size--;

delete temp;

return;

}

temp = temp->getNext();

}

}

void print() {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->getData() << " ";

temp = temp->getNext();

}

cout << endl;

}

Node\* GetById(int index) {

Node\* temp = head;

while (temp != nullptr) {

if (index == temp->getIndex()) {

return temp;

}

temp = temp->getNext();

}

cout << "nothing this element in list";

return nullptr;

}

void DeleteById(int index) {

Node\* temp = head;

while (temp != nullptr) {

if (temp->getIndex() == index) {

if(temp == head) {

head = head->getNext();

head->setPrev(nullptr);

}

else if (temp == tail) {

tail = tail->getPrev();

tail->setNext(nullptr);

}

else {

temp->getPrev()->setNext(temp->getNext());

temp->getNext()->setPrev(temp->getPrev());

}

size--;

delete temp;

return;

}

temp = temp->getNext();

}

cout << "nothing element in the list";

}

};

int main() {

// указание рандома

srand(time(NULL));

// объявление списка

LinkedList<int> list;

cout << "enter a size of list" << endl;

int n;

cin >> n;

// задание рандомных чисел в список

for (int i = 0; i < n; i++) {

int a = -10 + rand() % (10 - (-10) + 1);

list.push\_back(a);

}

// вывод списка

list.print();

// удаление четных записей в списке

for (int i = 0; i < n; i++) {

if (list.GetById(i)->getData() % 2 == 0) {

list.DeleteById(i);

}

}

// вывод списка

list.print();

}

# Результат работы программы представлен на рисунке 1.

# 

Рисунок 1 – Результат выполнения программы