#include <iostream>

using namespace std;

struct Queue //очередь

{

int data;

Queue\* prev, \* next;

};

void appendLeft(Queue\*\*, Queue\*\*, int); // Добавялет элемент в очередь

void appendRight(Queue\*\*, Queue\*\*, int); // Добавялет элемент в очередь

int pop(Queue\*\*, Queue\*\*); // Удаляет элемент из очереди и возвращает значение

void delQueue(Queue\*\*, Queue\*\*); // Удаляет весю очередь

int menu(); // Меню для пользователя

void addElementLeft(Queue\*\*, Queue\*\*); // Запрашивает элемент и добавляет в очередь слева

void addElementRight(Queue\*\*, Queue\*\*); // Запрашивает элемент и добавляет в очередь справа

void showQueue(Queue\*, Queue\*); // Выводит очередь в консоль

void showQueueRev(Queue\*, Queue\*); // Выводит очередь в консоль в обратном порядке

void task(Queue\*\*, Queue\*\*); // Задание

int main()

{

Queue\* begin = NULL, \* end = NULL;

while (true)

{

switch (menu())

{

case 1:

addElementLeft(&begin, &end);

break;

case 2:

addElementRight(&begin, &end);

break;

case 3:

showQueue(begin, end);

break;

case 4:

showQueueRev(begin, end);

break;

case 5:

delQueue(&begin, &end);

break;

case 6:

task(&begin, &end);

break;

case 7:

cout << "program is over";

return 0;

default:

cout << "Wrong choice\n";

}

cout << endl;

}

}

int menu()

{

int choice, elem;

cout << "1) Add element to begin" << endl;

cout << "2) Add element to end" << endl;

cout << "3) Show queue" << endl;

cout << "4) Show queue (reversed)" << endl;

cout << "5) Delete queue" << endl;

cout << "6) Individual task" << endl;

cout << "7) Exit" << endl;

cout << ">>> ";

cin >> choice;

cout << endl;

return choice;

}

void appendLeft(Queue\*\* begin, Queue\*\* end, int var)

{

Queue\* elem = new Queue;

elem->data = var;

elem->prev = NULL; // Задаём изначально все пустые значения

elem->next = NULL;

if (\*begin == NULL)

{

\*begin = \*end = elem; // Если очередь пуста, то элемент одновремено является и головой и хвостом

}

else

{

(\*begin)->prev = elem; // Иначе задаём текущему 1-му наш в качестве левого соседа и меняем голову на новый

elem->next = \*begin;

\*begin = elem;

}

}

void appendRight(Queue\*\* begin, Queue\*\* end, int var)

{

Queue\* elem = new Queue;

elem->data = var;

elem->prev = NULL; // Задаём изначально все пустые значения

elem->next = NULL;

if (\*begin == NULL)

{

\*begin = \*end = elem; // Если очередь пуста, то элемент одновремено является и головой и хвостом

}

else

{

(\*end)->next = elem; // Иначе задаём текущему последнему наш в качестве правого соседа и меняем хвост на новый

elem->prev = \*end;

\*end = elem;

}

}

int pop(Queue\* element)

{

// Сшиваем очередь, чтобы соседи не потеряли друг друга

if (element->prev != NULL)

{

element->prev->next = element->next;

}

if (element->next != NULL)

{

element->next->prev = element->prev;

}

int data = element->data;

delete element;

return data;

}

void showQueue(Queue\* begin, Queue\* end)

{

// Проверяем, что в очереди что-то есть

if (begin == NULL)

{

cout << "Queue is empty\n";

return;

}

Queue\* current = begin;

cout << "Queue: ";

while (current != NULL)

{

cout << current->data << ' ';

current = current->next;

}

cout << endl;

}

void showQueueRev(Queue\* begin, Queue\* end)

{

// Проверяем, что в очереди что-то есть

if (begin == NULL)

{

cout << "Queue is empty\n";

return;

}

Queue\* current = end;

cout << "Queue: ";

while (current != NULL)

{

cout << current->data << ' ';

current = current->prev;

}

cout << endl;

}

void delQueue(Queue\*\* head, Queue\*\* end)

{

// Проверяем, что в очереди что-то есть

if (\*head == NULL)

{

cout << "Queue is empty\n";

return;

}

//cout << "Removing the queue... ";

// Удаляем всё за 1-м элементом

while ((\*head)->next != NULL)

{

pop((\*head)->next);

}

// Удаляем 1-ый элемент

pop(\*head);

\*head = \*end = NULL;

cout << "Queue is empty now\n";

}

void addElementLeft(Queue\*\* begin, Queue\*\* end)

{

int elem;

cout << "Enter a new element: ";

cin >> elem;

appendLeft(begin, end, elem);

}

void addElementRight(Queue\*\* begin, Queue\*\* end)

{

int elem;

cout << "Enter a new element: ";

cin >> elem;

appendRight(begin, end, elem);

}

void task(Queue\*\* begin, Queue\*\* end)

{

// Проверяем, что в очереди что-то есть

if (\*begin == NULL)

{

cout << "Queue is empty\n";

return;

}

// Становимся в начало, пробегаемся по всей очереди, обновляя min/max,

// затем меняем их данные местами и выводим очередь в консоль

Queue\* current = \*begin;

Queue\* min = current, \* max = current;

while (current != \*end)

{

if (current->data < min->data)

{

min = current;

}

//if (current->data > max->data)

//{

// max = current;

//}

current = current->next;

}

if (current->data < min->data)

{

min = current;

}

while (current != \*end)

{

//if (current->data < min->data)

//{

// min = current;

//}

if (current->data > max->data)

{

max = current;

}

current = current->next;

}

if (current->data > max->data)

{

max = current;

}

//swap(min->data, max->data);

// Если swap нельзя

int temp = min->data;

min->data = max->data;

max->data = temp;

showQueue(\*begin, \*end);

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#include <iostream>

#include <Windows.h>

using namespace std;

struct Spis1 {

int info;

Spis1\* next;

}\*begin1, \*end1;

Spis1\* Create(Spis1\*\* b, Spis1\*\* e, int value) { // Create(&begin1, &end1, value);

Spis1\*t = new Spis1;

t->info = value;

t->next = NULL;

if (\*b == NULL)

\*b = \*e = t;

else {

(\*e)->next = t;

\*e = t;

}

return t;

}

void View(Spis1\*begin1) { // View(begin1);

Spis1\*t = begin1;

if (begin1 == NULL) {

cout << "Список пуст! Заполните список!!" << endl;

return;

}

while (t != NULL) {

cout << t->info << endl;

t = t->next;

}

}

void Task(Spis1 \* begin1) {

Spis1\* t = NULL, \*t1, \*max\_begin1 = NULL, \*min\_begin1 = NULL;

int r;

int max = -999;

int min = 999;

do {

for (t1 = begin1; t1->next != t; t1 = t1->next)

if (t1->info > max) {

max = t1->info;

max\_begin1 = t1;

}

if (t1->info < min) {

min = t1->info;

min\_begin1 = t1;

}

t = t1;

} while (begin1->next != t);

min\_begin1->info = max;

max\_begin1->info = min;

}

void Del\_All(Spis1\*\*begin1) { // Del\_All(&begin1);

Spis1\*t;

while (\*begin1 != NULL) {

t = \*begin1;

\*begin1 = (\*begin1)->next;

delete t;

}

}

int main()

{

Del\_All(&begin1);

setlocale(LC\_ALL, "russian");

cout << "Выберите дейстие:" << endl;

int m, i, n, value;

bool choice = true;

Spis1\* begin1 = NULL;

Spis1 \* end1 = NULL;

while (choice) {

cout << "1)Заполнить список; \n2)Показать ваш список; \n3)Поменять значение max и min в списке; \n4)Освободить память; \n5)Выход из программы." << endl;

cin >> m;

switch (m) {

case 1:

if (begin1 != NULL) {

cout << "Очистите память!" << endl;

break;

}

cout << "Введите кол-во эл-ов списка:" << endl;

cin >> n;

cout << "Введите эл-ты списка:" << endl;

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

cin >> value;

Create(&begin1, &end1, value);

}

break;

case 2:

cout << "Просмотр списка:" << endl;

View(begin1);

break;

case 3:

Task(begin1);

cout << "Результат выполения:" << endl;

View(begin1);

break;

case 4:

Del\_All(&begin1);

cout << "Память очищена!" << endl;

break;

case 5:

exit(0);

default: choice = false;

}

}

return 0;

}