**Обычная очередь**

#include <iostream>

using namespace std;

struct Node

{

int value;

Node\* next;

int key;

};

Node\* Queue\_Add(struct Node\* pHead, struct Node\* pTail, int& count, int& Last\_Del\_Key);

void Queue\_Print(struct Node\* pHead, struct Node\* pTail, int& count);

Node\* Queue\_Del(struct Node\* pHead, struct Node\* pTail, int& count, int& Last\_Del\_Key);

Node\* Queue\_Clear(struct Node\* pHead, int& count);

Node\* Queue\_Destroy(struct Node\* pHead, struct Node\* pTail);

Node\* Circle\_Queue(struct Node\* pHead, struct Node\* pTail);

void Find\_Key(struct Node\* phead);

void Find\_Position(struct Node\* pHead);

int main()

{

setlocale(0, "");

Node\* pHead = NULL;

Node\* pTail = NULL;

int count = 0;

int\* pcount = &count;

int Last\_Del\_Key = 0;

int\* pLast\_Del\_Key = &Last\_Del\_Key;

menu: int menu\_value = 0;

cout << "1 - Создание очереди;\n"

<< "2 - Вывод очереди на экран;\n"

<< "3 - Добавить в очередь\n"

<< "4 - Удалить из очереди\n"

<< "5 - В конец очереди (кольцевая очередь)\n"

<< "6 - Очистить очередь\n"

<< "7 - Уничтожить очередь\n"

<< "8 - Поиск по ключу\n"

<< "9 - Поиск позиции по значению\n"

<< "Выбор: "; cin >> menu\_value;

switch (menu\_value)

{

case 1:

{

pHead = new (Node);

pHead->next = NULL;

system("cls");

cout << "\n\n\tОчередь создана!\n" << endl;

system("pause");

system("cls");

goto menu;

}

case 2:

{

system("cls");

Queue\_Print(pHead, pTail, \*pcount);

system("pause");

system("cls");

goto menu;

}

case 3:

{

system("cls");

pTail = Queue\_Add(pHead, pTail, \*pcount, \*pLast\_Del\_Key);

system("pause");

system("cls");

goto menu;

}

case 4:

{

system("cls");

pHead = Queue\_Del(pHead, pTail, \*pcount, \*pLast\_Del\_Key);

if (pHead !=NULL)

{

if (pHead->next == NULL)

{

pTail = NULL;

}

}

system("pause");

system("cls");

goto menu;

}

case 5:

{

system("cls");

pTail = Circle\_Queue(pHead, pTail);

system("pause");

system("cls");

goto menu;

}

case 6:

{

system("cls");

pHead = Queue\_Clear(pHead, \*pcount);

if (pHead !=NULL)

{

if (pHead->next == NULL)

{

pTail = NULL;

}

}

system("pause");

system("cls");

goto menu;

}

case 7:

{

system("cls");

pHead = Queue\_Destroy(pHead, pTail);

system("pause");

system("cls");

goto menu;

}

case 8:

{

system("cls");

Find\_Key(pHead);

system("pause");

system("cls");

goto menu;

}

case 9:

{

system("cls");

Find\_Position(pHead);

system("pause");

system("cls");

goto menu;

}

default:

{

if (!(pHead == NULL))

{

if (!(pHead->next == NULL))

{

Queue\_Clear(pHead, \*pcount);

}

Queue\_Destroy(pHead, pTail);

}

cout << "Происходит выход..." << endl;

system("pause");

}

}

return 0;

}

void Queue\_Print(struct Node\* pHead, struct Node\* pTail, int& count)

{

if (pHead == NULL)

{

cout << "\n\t\tОчередь не создана!\n\n";

}

else

{

if (pHead->next == NULL)

{

cout << "\n\t\tОчередь пустая!\n" << endl;

}

else

{

Node\* t;

for (t = pHead->next; t != NULL; t = t->next)

{

cout << "Значение равно = " << t->value

<< "\tЕго адрес = " << &\*t

<< "\tЕго ключ = " << t->key << endl;

}

}

cout << "\n\t\tАдрес первого элемента = " << pHead->next << endl << endl;

cout << "\n\t\tАдрес последнего элемента = " << pTail << endl << endl;

cout << "\n\t\tКоличество элементов в очереди = " << count << endl << endl;

}

}

Node\* Queue\_Add(struct Node\* pHead, struct Node\* pTail, int& count, int& Last\_Del\_Key)

{

if (pHead == NULL)

{

cout << "\n\tСначала необходимо создать очередь!\n\n";

}

else

{

Node\* t;

if (pHead->next == NULL)

{

count++;

t = new (Node);

cout << "Введите значение: "; cin >> t->value;

t->next = NULL;

t->key = count + 100;

pHead->next = t;

pTail = t;

}

else

{

count++;

t = pTail;

t->next = new (Node);

t = t->next;

cout << "Введите значение: "; cin >> t->value;

t->next = NULL;

if (Last\_Del\_Key == NULL)

{

t->key = count + 100;

}

else

{

t->key = Last\_Del\_Key;

Last\_Del\_Key = 0;

}

pTail = t;

}

}

return pTail;

}

Node\* Queue\_Del(struct Node\* pHead, struct Node\* pTail, int& count, int& Last\_Del\_Key)

{

if (pHead == NULL)

{

cout << "\n\t\tОчередь не создана!\n\n";

}

else

{

if (pHead->next == NULL)

{

cout << "\n\t\tОчередь пустая!\n" << endl;

}

else

{

Node\* t;

t = pHead->next;

cout << "\n\tЗначение удаляемого элемента = " << t->value;

cout << "\n\n\t\tЕго адрес = " << t << endl;

cout << "\n\t\tЕго ключ = " << t->key << endl << endl;

if (pTail == pHead->next)

{

pHead->next = t->next;

pTail = pHead->next;

count--;

Last\_Del\_Key = t->key;

}

else

{

pHead->next = t->next;

count--;

Last\_Del\_Key = t->key;

}

delete t;

t = nullptr;

}

}

return pHead;

}

Node\* Queue\_Clear(struct Node\* pHead, int& count)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n" << endl;

}

else

{

if (pHead->next == NULL)

{

cout << "\n\t\tОчередь пустая,очищать нечего!\n" << endl;

}

else

{

Node\* t;

while (pHead->next != NULL)

{

t = pHead->next;

pHead->next = t->next;

delete t;

count--;

}

cout << "\n\t\tОчистка очереди завершена!\n\n";

}

}

return pHead;

}

Node\* Queue\_Destroy(struct Node\* pHead, struct Node\* pTail)

{

if (pHead == NULL)

{

cout << "\n\t\tРазрушать нечего!\n" << endl;

}

else

{

if (!(pHead->next == NULL))

{

cout << "\n\t\tСначала очистите очередь!\n" << endl;

}

else

{

delete pHead;

delete pTail;

pHead = nullptr;

pTail = nullptr;

cout << "\n\t\tОчередь разрушена!\n" << endl;

}

}

return pHead;

}

Node\* Circle\_Queue(struct Node\* pHead, struct Node\* pTail)

{

if (pHead == NULL)

{

cout << "\n\t\tОчередь не создана!\n\n";

}

else

{

if (pHead->next == NULL)

{

cout << "\n\t\tОчередь пустая!\n" << endl;

}

else

{

if (pHead->next == pTail)

{

cout << "\n\t\tВ очереди всего один элемент.\n\n";

}

else

{

Node\* t;

t = pHead->next;

cout << "\n\tВ конец отправляется элемент = " << t->value;

cout << "\n\n\t\tЕго адрес = " << t << endl;

pHead->next = t->next;

pTail->next = t;

pTail = t;

pTail->next = NULL;

}

}

}

return pTail;

}

void Find\_Key(struct Node\* pHead)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!" << endl;

}

else

{

if (pHead->next == NULL)

{

cout << "\n\t\tОчередь пустая!" << endl;

}

else

{

int cin\_key = 0;

bool already = false;

cout << "Введите ключ = "; cin >> cin\_key;

Node\* t;

t = pHead->next;

for (; t != NULL; t = t->next)

{

if (t->key == cin\_key)

{

cout << "\n\t\tЗначение найдено!\n\n\t\tАдрес = " << t

<< "\n\n\t\tЗначение = " << t->value << endl << endl;

already = true;

}

}

if (!already)

{

cout << "\n\tЗначения по ключу не найдено!\n" << endl;

}

}

}

}

void Find\_Position(struct Node\* pHead)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n\n";

}

else

{

if (pHead->next == NULL)

{

cout << "\n\t\tОчередь пустая!\n\n";

}

else

{

Node\* t;

int cin\_value;

bool already = false;

cout << "Введите значение: "; cin >> cin\_value;

for (t = pHead->next; t != NULL; t = t->next)

{

if (t->value == cin\_value)

{

cout << "\n\t\tЗначение найдено!\n\n"

<< "\t\tЕго адрес = " << t

<< "\n\n\t\tЕго ключ = " << t->key << endl << endl;

already = true;

}

}

if (!already)

{

cout << "\n\t\tЗначение не найдено!" << endl << endl;

}

}

}

}

**Очередь на основе статического массива СРС**

#include <iostream>

using namespace std;

struct Node

{

int value;

int key;

};

Node\* Queue\_Add(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position, const int size, int& Last\_Del\_Key);

void Queue\_Print(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position);

Node\* Queue\_Del(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position, int& Last\_Del\_Key);

Node\* Circle\_Queue(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position);

Node\* Queue\_Clear(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position);

Node\* Queue\_Destroy(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position);

int main()

{

setlocale(0, "");

const int size = 5;

Node Arr\_Queue[size]{};

Node\* pHead = NULL;

Node\* pTail = NULL;

int Last\_Del\_Key = 0;

int\* pLast\_Del\_Key = &Last\_Del\_Key;

int position = 0;

int\* pposition = &position;

menu: int menu\_value = 0;

cout << "1 - Создание очереди;\n"

<< "2 - Вывод очереди на экран;\n"

<< "3 - Добавить в очередь\n"

<< "4 - Удалить из очереди\n"

<< "5 - В конец очереди (кольцевая очередь)\n"

<< "6 - Очистить очередь\n"

<< "7 - Уничтожить очередь\n"

<< "Выбор: "; cin >> menu\_value;

switch (menu\_value)

{

case 1:

{

pHead = new(Node);

pTail = new(Node);

pHead->key = (-996);

pTail->key = (-996);

pHead->value = NULL;

pTail->value = NULL;

system("cls");

cout << "\n\n\tОчередь создана!\n" << endl;

system("pause");

system("cls");

goto menu;

}

case 2:

{

system("cls");

Queue\_Print(pHead, pTail, Arr\_Queue, \*pposition);

system("pause");

system("cls");

goto menu;

}

case 3:

{

system("cls");

Queue\_Add(pHead, pTail, Arr\_Queue, \*pposition, size, \*pLast\_Del\_Key);

system("pause");

system("cls");

goto menu;

}

case 4:

{

system("cls");

Queue\_Del(pHead, pTail, Arr\_Queue, \*pposition, \*pLast\_Del\_Key);

system("pause");

system("cls");

goto menu;

}

case 5:

{

system("cls");

Circle\_Queue(pHead, pTail, Arr\_Queue, \*pposition);

system("pause");

system("cls");

goto menu;

}

case 6:

{

system("cls");

Queue\_Clear(pHead, pTail, Arr\_Queue, \*pposition);

system("pause");

system("cls");

goto menu;

}

case 7:

{

system("cls");

pHead = Queue\_Destroy(pHead, pTail, Arr\_Queue, \*pposition);

system("pause");

system("cls");

goto menu;

}

default:

{

cout << "Происходит выход..." << endl;

system("pause");

}

}

return 0;

}

Node\* Queue\_Add(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position, const int size, int& Last\_Del\_Key)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n\n";

}

else

{

if (position == size)

{

cout << "\n\t\tОчередь переполненная!\n\n";

}

else

{

cout << "Введите значение: ";

cin >> Arr\_Queue[position].value;

if (Last\_Del\_Key != 0)

{

Arr\_Queue[position].key = Last\_Del\_Key;

Last\_Del\_Key = 0;

}

else

{

Arr\_Queue[position].key = 100 + position;

}

if (position == 0)

{

pHead->key = position;

pTail->key = position;

}

else

{

pTail->key = position;

}

position++;

}

}

return 0;

}

void Queue\_Print(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n\n";

}

else

{

if (pHead->key == (-996))

{

cout << "\n\t\tОчередь пустая, выводить нечего!\n\n";

}

else

{

for (int i = 0; i < position; i++)

{

cout << i + 1 << " элемент = " << Arr\_Queue[i].value << "\tЕго ключ = " << Arr\_Queue[i].key << endl;

}

}

}

}

Node\* Queue\_Del(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position, int& Last\_Del\_Key)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n\n";

}

else

{

if (pHead->key == (-996))

{

cout << "\n\t\tОчередь пустая, удалять нечего!\n\n";

}

else

{

cout << "\t\tУдаление узла \n\t\tЕго значение = "

<< Arr\_Queue[0].value << "\n\t\tЕго ключ = "

<< Arr\_Queue[0].key << endl;

if (pHead->key == pTail->key)

{

Last\_Del\_Key = Arr\_Queue[0].key;

Arr\_Queue[0].value = NULL;

Arr\_Queue[0].key = NULL;

pHead->key = (-996);

pTail->key = (-996);

position--;

cout << "\n\t\tУдаление завершено!\n\n";

}

else

{

Last\_Del\_Key = Arr\_Queue[0].key;

for (int i = 0; i < position - 1; i++)

{

Arr\_Queue[i] = Arr\_Queue[i + 1];

}

cout << "\n\t\tУдаление завершено!\n\n";

Arr\_Queue[position - 1].value = NULL;

Arr\_Queue[position - 1].key = NULL;

position--;

pTail->key = position - 1;

}

}

}

return 0;

}

Node\* Circle\_Queue(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n\n";

}

else

{

if (pHead->key == (-996))

{

cout << "\n\t\tОчередь пустая\n\n";

}

else

{

if (pHead->key == pTail->key)

{

cout << "\n\t\tВ очереди всего один элемент.\n\n";

}

else

{

Node\* t = new(Node);

t->value = Arr\_Queue[0].value;

t->key = Arr\_Queue[0].key;

for (int i = 0; i < position - 1; i++)

{

Arr\_Queue[i] = Arr\_Queue[i + 1];

}

Arr\_Queue[position - 1].value = t->value;

Arr\_Queue[position - 1].key = t->key;

delete t;

t = nullptr;

}

}

}

return 0;

}

Node\* Queue\_Clear(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n\n";

}

else

{

if (pHead->key == (-996))

{

cout << "\n\t\tОчередь пустая, удалять нечего!\n\n";

}

else

{

for (int i = 0; i < position; i++)

{

Arr\_Queue[i].value = NULL;

Arr\_Queue[i].key = NULL;

}

position = NULL;

pHead->key = (-996);

pTail->key = (-996);

cout << "\n\t\tОчистка очереди завершена!\n\n";

}

}

return 0;

}

Node\* Queue\_Destroy(struct Node\* pHead, struct Node\* pTail, struct Node Arr\_Queue[], int& position)

{

if (pHead == NULL)

{

cout << "\n\t\tСоздайте очередь!\n\n";

}

else

{

if (!(pHead->key == (-996)))

{

cout << "\n\t\tСначала очистите очередь!\n\n";

}

else

{

delete pHead;

delete pTail;

pHead = nullptr;

pTail = nullptr;

cout << "\n\t\tОчередь разрушена!\n" << endl;

}

}

return pHead;

}