**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра математического обеспечения и применения ЭВМ**

**Отчет по лабораторной работе №1** по курсу «Структуры и алгоритмы обработки данных»

на тему

**«Представление линейных списков»**

**Вариант №3**

Выполнила: студент  группы КТбо1-1: Фабисяк А.А.

Проверил:

Доцент каф. МОП ЭВМ

Дроздов С.Н.

Таганрог 2020

**Цель работы**

Выполнение лабораторной работы заключается в разработке и отладке программы работы со списками, согласно полученному варианту задания.

**Формулировка задачи**

Для всех вариантов нужно реализовать меню со следующим минимальным набором операций со списком:

* инициализация пустого списка;
* уничтожение списка с освобождением памяти;
* добавление узла в голову списка;
* добавление узла в хвост списка;
* удаление узла из головы списка;
* удаление узла из хвоста списка;
* выдача текущего списка на экран.

1. Кроме того, в каждом варианте следует реализовать две дополнительные операции. Двусвязный линейный список. Дополнительные операции: a) добавить новый узел в указанную позицию; б) поменять местами первый и последний узлы (требуется поменять именно узлы, а не их значения).

**Листинг программы на языке С++**

//СиАОД\_Лаб.1\_Вар.3\_ФабисякА.А

#include<iostream>

using namespace std;

struct Node

{

int A;

Node\* Next;

Node \*Previous;

};

struct List

{

Node \*Tail;

Node \*Head;

int NodeCount;

};

void ListInitialize ( List \*list )

{

list -> Head = 0;

list -> Tail = 0;

list -> NodeCount = 0;

}

void TailAdd ( List \*Spisok, int B )

{

Node \*NewNode=new Node;

NewNode -> Next=0;

NewNode -> A=B;

if (Spisok -> Head == NULL && Spisok ->Tail == NULL)

{

NewNode -> Previous=0;

Spisok ->Head = NewNode;

Spisok -> Tail = NewNode;

}

else

{

NewNode -> Previous = Spisok ->Tail;

(Spisok ->Tail) -> Next = NewNode;

Spisok ->Tail = NewNode;

}

Spisok -> NodeCount++;

}

void HeadAdd( List \*Spisok, int B)

{

Node \*NewNode = new Node;

NewNode -> A = B;

NewNode -> Previous = NULL;

NewNode -> Next = Spisok ->Head;

if ( (Spisok ->Head) != NULL )

{

(Spisok ->Head) -> Previous = NewNode;

}

else Spisok ->Tail = NewNode;

(Spisok ->Head) = NewNode;

Spisok -> NodeCount++;

}

void Print(List Spisok)

{

Node \*Current = Spisok.Head;

if (Spisok.NodeCount > 0)

{

while (Current != NULL)

{

cout << Current->A << ' ';

Current = Current->Next;

}

}

else cout << "Список пуст.";

cout<<"\n";

}

void Delete( List \*Spisok)

{

Node \*Current;

while( Spisok ->Head!=NULL )

{

Current = Spisok ->Head;

Spisok ->Head = (Spisok ->Head ) -> Next;

delete Current;

Spisok -> NodeCount--; //будет равен 0 после удалления

}

Spisok ->Tail = NULL;

}

void DeletHead ( List \*Spisok)

{

if ( Spisok ->Head != Spisok ->Tail )

{

Spisok ->Head = (Spisok ->Head ) -> Next;

delete (Spisok ->Head) -> Previous;

(Spisok ->Head) -> Previous = NULL;

Spisok -> NodeCount--;

}

else

{

if ( Spisok ->Head != NULL)

{

delete (Spisok ->Head);

Spisok -> NodeCount--;

Spisok ->Head =NULL;

Spisok ->Tail = NULL;

}

}

}

void DeletTail ( List \*Spisok)

{

if ( Spisok ->Head != Spisok ->Tail )

{

Spisok ->Tail = (Spisok ->Tail) -> Previous;

delete (Spisok ->Tail) -> Next;

(Spisok ->Tail) -> Next = NULL;

Spisok -> NodeCount--;

}

else

{

if ( Spisok ->Head != NULL)

{

delete (Spisok ->Head);

Spisok -> NodeCount--;

Spisok ->Head =NULL;

Spisok ->Tail = NULL;

}

}

}

void NodeAdd (List \*Spisok, int B, int PositionNumber)

{

if ( Spisok ->Head == NULL && Spisok ->Tail == NULL)

{

HeadAdd(Spisok, B);

}

else

{

if (PositionNumber <= 0)

{

HeadAdd(Spisok,B);

}

else if ( PositionNumber >= Spisok -> NodeCount)

{

TailAdd(Spisok, B);

}

else

{

Node \*NewNode = new Node;

NewNode -> A = B;

Node \*Current = Spisok ->Head;

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

{

Current = Current -> Next;

}

NewNode -> Next = Current -> Next;

NewNode -> Previous = Current;

Current -> Next -> Previous = NewNode;

Current -> Next = NewNode;

Spisok -> NodeCount++;

}

}

}

void ChangeNode(List \*Spisok)

{

if (Spisok->Head != Spisok->Tail)

{

if (Spisok->NodeCount > 2)

{

Spisok->Tail->Next = Spisok->Head->Next;

Spisok->Head->Previous = Spisok->Tail->Previous;

Spisok->Tail->Previous->Next = Spisok->Head;

Spisok->Head->Next->Previous = Spisok->Tail;

Node \*TempNode = Spisok->Head;

Spisok->Head = Spisok->Tail;

Spisok->Tail = TempNode;

Spisok->Tail->Next = NULL;

Spisok->Head->Previous = NULL;

}

else

{

Spisok->Head->Previous = Spisok->Tail;

Spisok->Tail->Next = Spisok->Head;

Spisok->Tail->Previous = NULL;

Spisok->Head->Next = NULL;

Node \*TempNode = Spisok->Head;

Spisok->Head = Spisok->Tail;

Spisok->Tail = TempNode;

}

}

}

int main()

{

List Spisok;

ListInitialize( &Spisok);

int NumberCommand, B;

bool IsListInitialized = false;

while ( true )

{

cout << "Выберите номер команды: "<<"\n";

cout << "1 - Инициализировать список" << "\n" <<

"2 - Добавить элемент в голову списка" << "\n" <<

"3 - Добавить элемент в хвост списка" << "\n" <<

"4 - Добавить элемент в позицию списка" << "\n" <<

"5 - Удалить элемент из головы списка" << "\n" <<

"6 - Удалить элемент из хвоста списка" << "\n" <<

"7 - Обмен местами первого и последнего элементов списка" << "\n" <<

"8 - Вывод списка" << "\n" <<

"9 - Удаление списка" << "\n" <<

"10 - Завершение программы" << "\n";

cin>>NumberCommand;

if (NumberCommand == 9)

{

Delete(&Spisok);

break;

}

else

{

if (!IsListInitialized && 0 < NumberCommand && NumberCommand < 9)

{

ListInitialize(&Spisok);

IsListInitialized = true;

}

switch (NumberCommand)

{

case 1:

if (IsListInitialized)

{

cout << "Невозможно выполнить эту команду. Список уже инициализирован";

if (Spisok.NodeCount > 0) cout << " и не является пустым";

cout << ".\n";

}

else

{

ListInitialize(&Spisok);

IsListInitialized = true;

cout << "Список успешно инициализирован.\n";

}

break;

case 2:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

cout << "Введите значение нового узла: ";

cin >> B;

HeadAdd(&Spisok, B);

Print(Spisok);

}

break;

case 3:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

cout << "Введите значение нового узла: ";

cin >> B;

TailAdd(&Spisok, B);

Print(Spisok);

}

break;

case 4:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

int PositionNumber;

cout << "Введите через пробел значение нового узла и номер позиции: ";

cin >> B >> PositionNumber;

NodeAdd(&Spisok, B, PositionNumber);

Print(Spisok);

}

break;

case 5:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

DeletHead(&Spisok);

Print(Spisok);

}

break;

case 6:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

DeletTail(&Spisok);

Print(Spisok);

}

break;

case 7:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

ChangeNode(&Spisok);

Print(Spisok);

}

break;

case 8:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

Print(Spisok);

}

break;

case 9:

if (!IsListInitialized) cout << "Список еще не инициализирован!\n";

else

{

Delete(&Spisok);

cout << "Список успешно удален.\n";

}

break;

default:

cout << "Команда под номером "

<< NumberCommand

<< " не определена!\nПопробуйте снова!\n";

break;

}

}

}

return 0;

}

**Заключение**

В процессе выполнения лабораторной работы №1, я ознакомилась с технологией разработки и отладки работы с двусвязным линейным списком.