Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

# Тема: Лабораторная работа № 11 "Двусвязный список"

# Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Юхновец В.Г.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи**

Реализовать двусвязный список с методами добавления, удаления и просмотром элементов.

**Алгоритм программы**

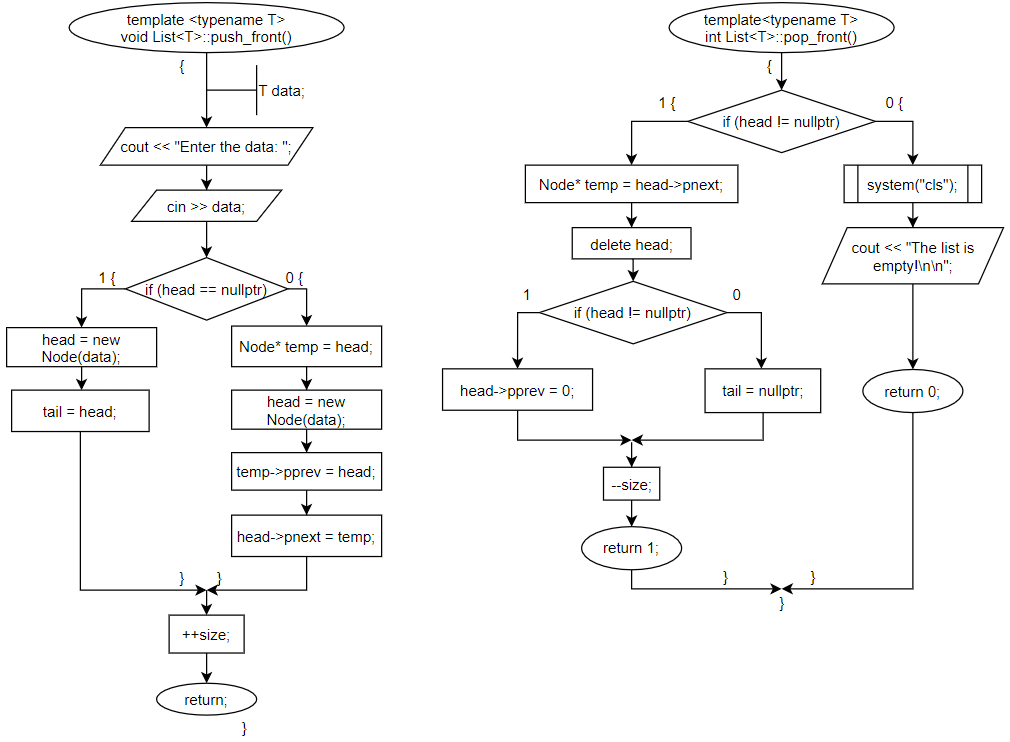


Рисунок 1 – метод добавления в начало и удаления из начала списка

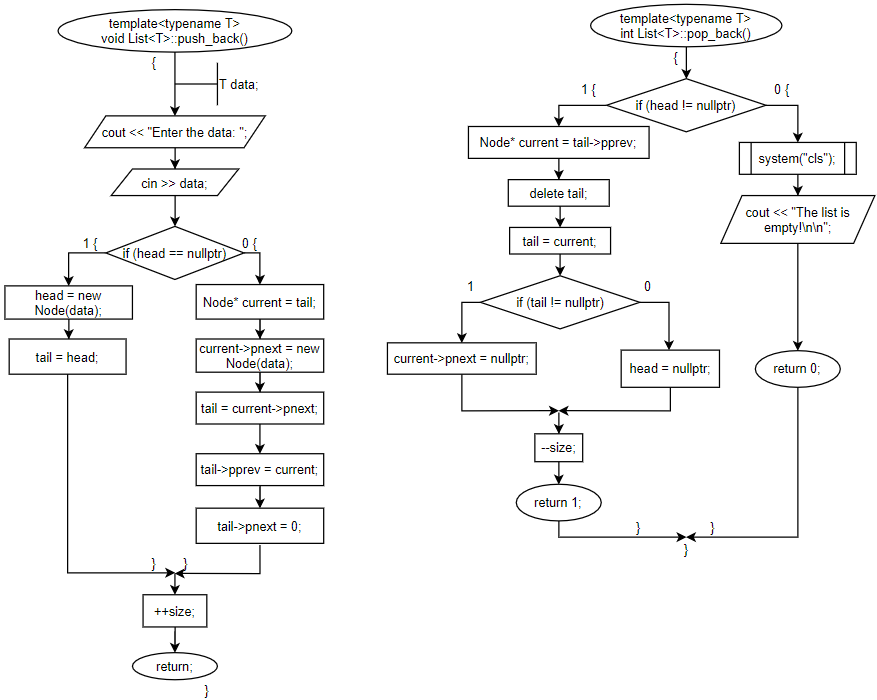


Рисунок 2 – метод добавления в конец и удаления из конца списка

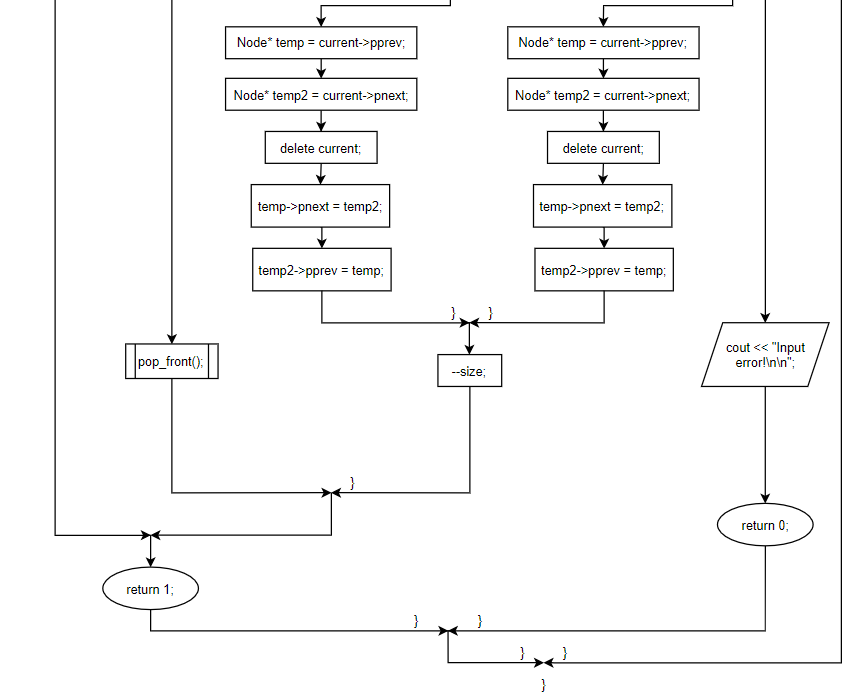
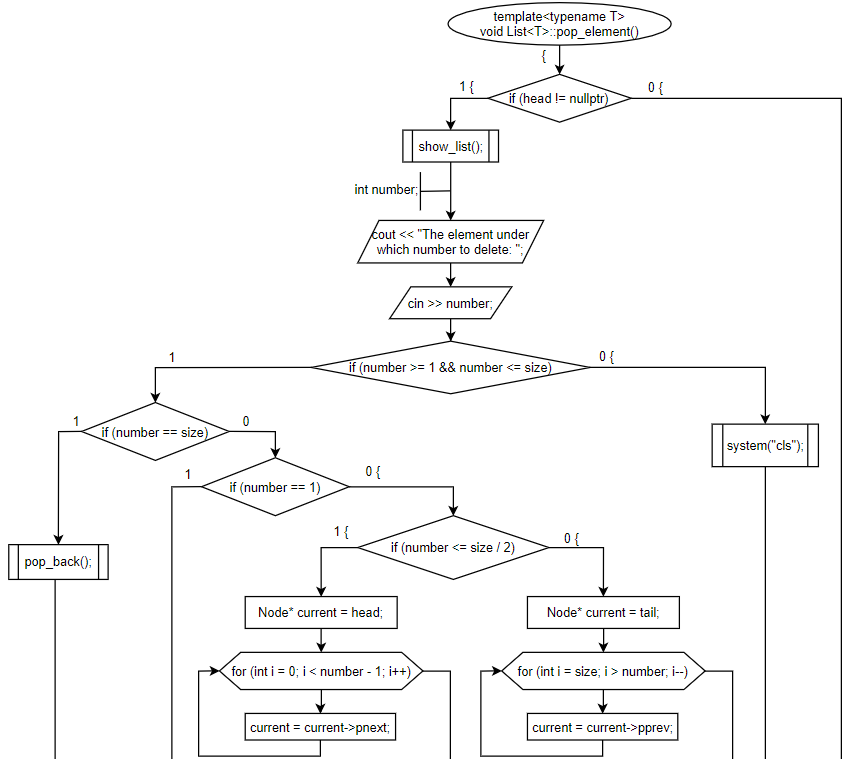


Рисунок 3 – метод удаления элемента по индексу

**Код программы**

#include <iostream>

#include <string>

using namespace std;

template <typename T>

struct List

{

void push\_front();

int pop\_front();

void push\_back();

int pop\_back();

int add\_after(int number);

int pop\_element(int number);

void add\_several\_elements();

void delete\_several\_elements();

void show\_list();

struct Node

{

Node(T data)

{

this->data = data;

}

Node\* pnext = nullptr;

Node\* pprev = nullptr;

T data;

};

Node\* head = nullptr;

Node\* tail = nullptr;

int size = 0;

};

template <typename T>

void List<T>::push\_front()

{

T data;

cout << "Enter the data: ";

cin >> data;

if (head == nullptr)

{

head = new Node(data);

tail = head;

}

else

{

Node\* temp = head;

head = new Node(data);

temp->pprev = head;

head->pnext = temp;

}

++size;

}

template<typename T>

int List<T>::pop\_front()

{

if (head != nullptr)

{

Node\* temp = head->pnext;

delete head;

head = temp;

if (head != nullptr)

head->pprev = 0;

else

tail = nullptr;

--size;

return 1;

}

else

{

system("cls"); cout << "The list is empty!\n\n";

return 0;

}

}

template<typename T>

void List<T>::push\_back()

{

T data;

cout << "Enter the data: ";

cin >> data;

if (head == nullptr)

{

head = new Node(data);

tail = head;

}

else

{

Node\* current = tail;

current->pnext = new Node(data);

tail = current->pnext;

tail->pprev = current;

tail->pnext = 0;

}

++size;

}

template<typename T>

int List<T>::pop\_back()

{

if (head != nullptr)

{

Node\* current = tail->pprev;

delete tail;

tail = current;

if (tail != nullptr)

current->pnext = nullptr;

else

head = nullptr;

--size;

return 1;

}

else { system("cls"); cout << "The list is empty!\n\n"; return 0; }

}

template<typename T>

void List<T>::show\_list()

{

if (head != nullptr)

{

int choice;

cout << "Enter 1 - list output from the beginning\nEnter 2 - list output from the end\n\n";

cin >> choice;

system("cls");

switch (choice)

{

case 1:

{

cout << "List:";

Node\* current = head;

for (size\_t i = 0; i < size; i++)

{

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

current = current->pnext;

}

cout << endl << endl;

break;

}

case 2:

{

cout << "List: ";

Node\* current = tail;

for (size\_t i = size; i > 0; i--)

{

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

current = current->pprev;

}

cout << endl << endl;

break;

}

default:

system("cls"); cout << "Input error!\n\n";

break;

}

}

else { system("cls"); cout << "The list is empty!\n\n"; }

}

template<typename T>

int List<T>::add\_after(int number)

{

if (head != nullptr)

{

if (number >= 0 && number <= size)

{

if (number == size)

push\_back();

else if (number == 0)

push\_front();

else

{

T data;

cout << "Enter data: ";

cin >> data;

if (number <= size / 2)

{

Node\* current = head;

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

current = current->pnext;

Node\* temp = current->pnext;

current->pnext = new Node(data);

current->pprev = current;

current = current->pnext;

current->pnext = temp;

++size;

}

else

{

Node\* current = tail;

for (int i = size; i > number; i--)

current = current->pprev;

Node\* temp = current->pnext;

Node\* temp2 = current;

current->pnext = new Node(data);

temp->pprev = current->pnext;

current = current->pnext;

current->pprev = temp2;

current->pnext = temp;

++size;

}

}

return 1;

}

else { system("cls"); cout << "Input error!\n\n"; return 0; }

}

else { system("cls"); cout << "The list is empty!\n\n"; return 0; }

}

template<typename T>

void List<T>::add\_several\_elements()

{

int n, choice;

T data;

cout << "Enter 1 - add items to the top of the list\nEnter 2 - add items after the selected item\nEnter 3 - add items to the end of the list\n\n";

cout << "Your choice of answer: ";

cin >> choice;

system("cls");

switch (choice)

{

case 1:

cout << "Number of elements to add: ";

cin >> n;

cout << endl;

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

push\_front();

system("cls"); cout << "Data has been successfully added!\n\n";

break;

case 2:

if (head != nullptr)

{

show\_list();

cout << "Number of added elements: ";

cin >> n;

int number;

cout << "After which element to add others: ";

cin >> number;

cout << endl;

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

add\_after(number++);

system("cls"); cout << "Data has been successfully added!\n\n";

}

else { system("cls"); cout << "The list is empty!\n\n"; }

break;

case 3:

cout << "Number of added elements: ";

cin >> n;

cout << endl;

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

push\_back();

system("cls"); cout << "Data has been successfully added!\n\n";

break;

default:

cout << "Input error. Try again.\n\n";

break;

}

}

template<typename T>

int List<T>::pop\_element(int number)

{

if (head != nullptr)

{

if (number >= 1 && number <= size)

{

if (number == size)

pop\_back();

else if (number == 1)

pop\_front();

else

{

if (number <= size / 2)

{

Node\* current = head;

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

current = current->pnext;

Node\* temp = current->pprev;

Node\* temp2 = current->pnext;

delete current;

temp->pnext = temp2;

temp2->pprev = temp;

}

else

{

Node\* current = tail;

for (int i = size; i > number; i--)

current = current->pprev;

Node\* temp = current->pprev;

Node\* temp2 = current->pnext;

delete current;

temp->pnext = temp2;

temp2->pprev = temp;

}

--size;

}

return 1;

}

else { system("cls"); cout << "Input error!\n\n"; return 0; }

}

else { system("cls"); cout << "The list is empty!\n\n"; return 0; }

}

template<typename T>

void List<T>::delete\_several\_elements()

{

if (head != nullptr)

{

int start, count, choice;

cout << "Enter 1 - delete to the left\nEnter 2 - delete to the right\n\nYour choice of answer: ";

cin >> choice;

cout << "\nWhich element to start deleting: ";

cin >> start;

if (start >= 1 && start <= size)

{

cout << "Number of deleted items: ";

cin >> count;

if (start == 1 && choice == 2 && ((start + count) <= (size +1)))

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

pop\_front();

else if (start == size && choice == 1 && (start - count >= 0))

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

pop\_back();

else if (choice == 2 && ((start + count) <= (size + 1)))

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

pop\_element(start);

else if (choice == 1 && (start - count >= 0))

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

pop\_element(start--);

else { system("cls"); cout << "Input error!\n\n"; return; }

system("cls"); cout << "Data has been successfully deleted!\n\n";

}

else { system("cls"); cout << "Input error!\n\n"; }

}

}

int main()

{

system("chcp 1251 >> null");

List<string> list1;

List<int> list2;

int data;

int f = 1;

while (f)

{

int choice;

cout << "Enter 1 - add an element to the beginning of the list\nEnter 2 - add an element to the end of the list\nEnter 3 - add an element after the specified\nEnter 4 - add several elements\nEnter 5 - remove an element from the beginning of the list\nEnter 6 - remove an element from the end of the list\nEnter 7 - remove the specified item\nEnter 8 - delete several items\nEnter 9 - view the list\nEnter 10 - terminate the program\n\n";

cout << "Your choice of answer: ";

cin >> choice;

switch (choice)

{

case 1:

system("cls");

list1.push\_front();

system("cls");

cout << "Data has been successfully added!\n\n";

break;

case 2:

system("cls");

list1.push\_back();

system("cls");

cout << "Data has been successfully added!\n\n";

break;

case 3:

if (list1.head != nullptr)

{

system("cls");

list1.show\_list();

int number;

cout << "After which element number to add a new one: ";

cin >> number;

if (list1.add\_after(number) == 1)

{

system("cls");

cout << "The data has been successfully deleted!\n\n";

}

}

else { system("cls"); cout << "The list is empty!\n\n"; }

break;

case 4:

system("cls");

list1.add\_several\_elements();

break;

case 5:

system("cls");

if (list1.pop\_front() == 1)

{

system("cls");

cout << "The data has been successfully deleted!\n\n";

}

break;

case 6:

system("cls");

if (list1.pop\_back() == 1)

{

system("cls");

cout << "The data has been successfully deleted!\n\n";

}

break;

case 7:

system("cls");

list1.show\_list();

if (list1.head != nullptr)

{

int number;

cout << "Element under which number to delete: ";

cin >> number;

list1.pop\_element(number);

system("cls"); cout << "Data has been successfully deleted!\n\n";

}

break;

case 8:

system("cls");

list1.show\_list();

list1.delete\_several\_elements();

break;

case 9:

system("cls");

list1.show\_list();

break;

case 10:

system("cls"); cout << "Program shutdown.\n";

f = 0;

break;

default:

system("cls"); cout << "Input error. Try again.\n\n";

break;

}

}

return 0;

}

**Результаты работы программы**

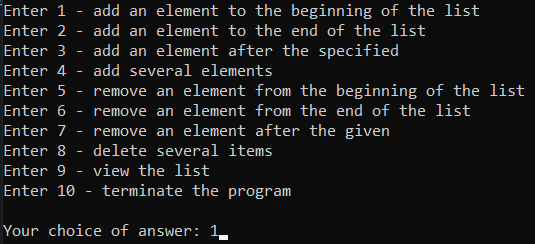


Рисунок 4 – меню программы с выбором ответа «1»



Рисунок 5 – добавление 10 в начало списка

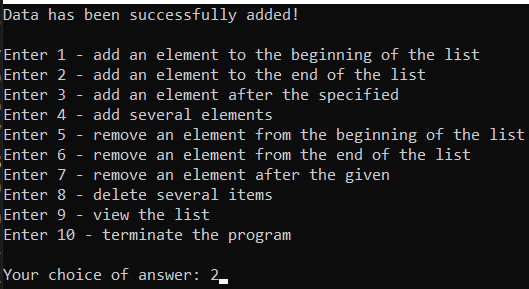


Рисунок 6 – меню программы с выбором ответа «2»



Рисунок 7 – добавление 20 в конец списка

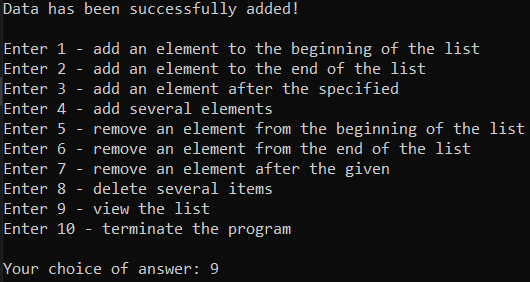


Рисунок 8 – меню программы с выбором ответа «9»

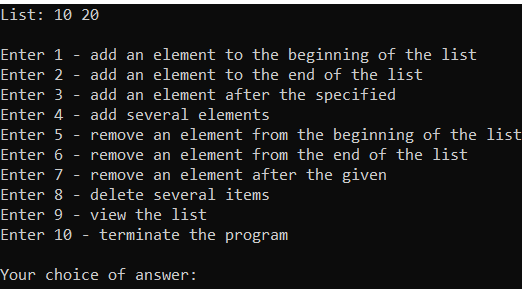


Рисунок 9 – вывод списка

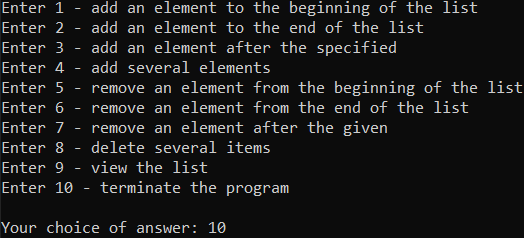


Рисунок 10 – меню программы с выбором ответа «10»

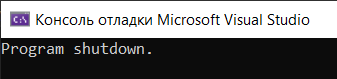


Рисунок 11 – конец работы программы