**МИНОБРНАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ   
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ТУЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт прикладной математики и компьютерных наук

Кафедра информационной безопасности

**ЯЗЫКИ ПРОГРАММИРОВАНИЯ**

Отчет по выполнению лабораторной работы №14  
Вариант №8

Выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ст. гр.230711 Павлова Виктория Сергеевна

Проверила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. каф. ИБ Басалова Галина Валерьевна

Тула 2022

# ЛАБОРАТОРНАЯ РАБОТА №14. ВВЕДЕНИЕ В ООП

## ЦЕЛЬ РАБОТЫ

Изучить основные понятия ООП: «объект», «класс», «инкапсуляция»; познакомиться со способами описания классов и объектов в языке С++; познакомиться с возможностью перегрузки операторов и использования конструкторов объектов класса; разработать приложения по своим вариантам заданий.

## ЗАДАНИЕ НА РАБОТУ

**Задание 1.** Ознакомиться с теоретическим материалом, приведенным в пункте «Краткие теоретические положения» данных методических указаний, а также с конспектом лекций и рекомендуемой литературой по данной теме.

**Задание 2.** Разработать программу по своему варианту. Для этого создать h-файл с объявлением и определением класса, а затем разработать основную программу (cpp-файл с функцией main), в которой используется созданный класс. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

## ХОД РАБОТЫ

Согласно заданию варианта №8, необходимо разработать программу с для создания и использования объектов класса List (однонаправленный список). Для этого нужно написать тексты соответствующего h-файла и cpp-файла, а также разработать вспомогательную структуру Node, которая выражает узел списка. Описание класса List представлено в таблице 1.

Таблица 1 – Описание разработанного класса LinkedList

|  |  |  |
| --- | --- | --- |
| class LinkedList | | |
| **Поля/свойства (элементы данных) класса** | | |
| Название и тип | Описание | |
| Node<T>\* head | Корень списка | |
| int size | Количество узлов в списке | |
| **Шаблон для описания типа данных, хранящихся в узлах списка** | | |
| template <typename T> | | |
| **Методы (функции-элементы) класса** | | |
| Название и тип возвращаемого значения | Аргументы | Описание |
| LinkedList() | Node<T>\*head = nullptr, int size = 0 | Конструктор класса |
| LinkedList() | const LinkedList<T>& origList | Конструктор копий |
| bool IsEmpty() | void | Если список пуст, вернёт true |
| int Count() | void | Вывод на консоль количества узлов в списке |
| bool PushBack() | T element | Добавить узел в конец списка, в случае успеха вернёт true |
| void Print() | void | Вывод списка всех узлов и хранящихся в них значений |
| T GetValue() | const int& pos | Вывод значения по номеру узла |
| bool Add() | const T& data, const int& pos | Вставить узел на указанную позицию, в случае успеха вернёт true |
| Node<T>\* GetNode | int pos | Вспомогательная функция, возвращающая узел, который хранится на позиции pos |
| bool Cut () | int pos | Удалить узел, хранящийся на указанной позиции, в случае успеха вернёт true |
| ~List() | void | Деструктор класса |

Для реализации данного класса была создана структура Node (узел), описание которой представлено в таблице 2.

Таблица 2 – Описание разработанной структуры Node

|  |  |  |
| --- | --- | --- |
| struct Node | | |
| **Поля/свойства (элементы данных) структуры** | | |
| Название и тип | Описание | |
| T value | Значение, которое хранится в узле | |
| Node<T>\* next | Ссылка на следующий элемент списка | |
| **Шаблон для описания типа данных, хранящихся в узлах списка** | | |
| template <typename T> | | |
| **Методы (функции-элементы) структуры** | | |
| Название и тип возвращаемого значения | Аргументы | Описание |
| Node() | T data  Node<T>\*nextPtr = nullptr | Конструктор структуры |
| ~Node() | void | Деструктор структуры |

Помимо вышеупомянутых методов и функций, для работы с классом List также были перегружены следующие операторы:

1. Операторы ввода и вывода из потока >> и << соответственно;
2. Логическое операторы сравнения == и !=;
3. Оператор присваивания =;
4. Оператор сложения списков +.

## КОД ПРОГРАММЫ

**Содержимое файла LinkedList.h**

#pragma once

#include <iostream>

using namespace std;

template <typename T>

struct Node

{

T value;

Node<T>\* next;

Node(T data, Node<T>\* nextPtr = nullptr)

{

value = data;

next = nextPtr;

}

~Node(){}

};

template <typename T>

class LinkedList

{

public:

Node<T>\* head;

int size;

LinkedList()

{

head = nullptr;

size = 0;

}

LinkedList(const LinkedList<T>& origList)

{

head = nullptr;

size = 0;

Node<T>\* origPtr = origList.head;

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

{

(\*this).PushBack(origPtr->value);

origPtr = origPtr->next;

}

}

LinkedList<T> operator+(const LinkedList<T>& q)

{

LinkedList<T> newList;

Node<T>\* curPtr = head;

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

{

newList.PushBack(curPtr->value);

curPtr = curPtr->next;

}

Node<T>\* qPtr = q.head;

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

{

newList.PushBack(qPtr->value);

qPtr = qPtr->next;

}

return newList;

}

LinkedList<T>& operator=(const LinkedList<T>& q)

{

(\*this).Clear();

Node<T>\* qPtr = q.head;

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

{

(\*this).PushBack(qPtr->value);

qPtr = qPtr->next;

}

return \*this;

}

bool operator==(const LinkedList<T>& q)

{

if (size != q.size) return false;

Node<T>\* curPtr = head;

Node<T>\* qPtr = q.head;

int i = 0, k = 0;

while (i < q.size)

{

if (curPtr->value == qPtr->value) k++;

curPtr = curPtr->next;

qPtr = qPtr->next;

i++;

}

return (k == q.size);

}

bool operator!=(const LinkedList<T>& q)

{

if (size == q.size) return false;

Node<T>\* curPtr = head;

Node<T>\* qPtr = q.head;

int i = 0, k = 0;

while (i < q.size)

{

if (curPtr->value != qPtr->value) k++;

curPtr = curPtr->next;

qPtr = qPtr->next;

i++;

}

return (k != q.size);

}

friend ostream& operator << (ostream& output, const LinkedList<T>&p)

{

Node<T>\* curPtr = p.head;

cout << "\n\t\t\t\tСписок:\n";

for (int i = 0; i < p.size; i++)

{

cout << "\t\t\t\t" << i+1 << ". " << curPtr->value << "\n";

curPtr = curPtr->next;

}

return output;

}

friend istream& operator >> (istream& input, LinkedList<T>& p)

{

T value;

cout << "\n\t\t\t\tВведите новый элемент: "; cin >> value;

p.PushBack(value);

return input;

}

void Print()

{

Node<T>\* curPtr = head;

cout << "\n\t\t\t\tСписок:\n";

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

{

cout << "\t\t\t\t" << i + 1 << ". " << curPtr->value << "\n";

curPtr = curPtr->next;

}

}

void PushBack(const Node<T>\* &node) //добавить узел node

{ //в конец списка

Node<T>\* newNode(node->value);

if (head == nullptr)

{

head = newNode;

size++;

return true;

}

GetNode(size - 1)->next = newNode;

size++;

return true;

}

void PushBack(const T &data) //добавить новый узел со

{ //значением data в конец списка

T newData = data;

Node<T>\* node = new Node<T>(newData);

if (head == nullptr)

{

head = node;

size++;

return;

}

GetNode(size - 1)->next = node;

size++;

}

bool Add(const T data, const int& pos)

{

if (pos == 0) //добавить в начало

{

Node<T>\* newNode = new Node<T>(data);

Node<T>\* temp = head;

newNode->next = temp;

head = newNode;

size++;

return true;

}

if (pos > 0 && pos < size) //добавить в середину

{

Node<T>\* newNode = new Node<T>(data);

newNode->next = GetNode(pos);

GetNode(pos - 1)->next = newNode;

size++;

return true;

}

return false;

}

Node<T>\* GetNode(const int &pos)

{

if (pos == 0) return head;

Node<T>\* nodePtr = nullptr;

if (pos > -1 && pos < size)

{

nodePtr = head;

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

{

nodePtr = nodePtr->next;

}

}

return nodePtr;

}

T GetValue(const int& pos)

{

return GetNode(pos)->value;

}

bool PopBack()

{

if (GetNode(size - 1) != nullptr)

{

delete GetNode(size - 1);

size--;

return true;

}

return false;

}

bool Cut(const int& pos)

{

if (pos == 0) //удаляем корень

{

head = head->next;

size--;

return true;

}

if (pos == size - 1) //удаляем последний

{

return (\*this).PopBack();

}

if (pos > 0 && pos < size - 1) //удаляем из середины

{

GetNode(pos - 1)->next = GetNode(pos)->next;

delete GetNode(pos);

size--;

return true;

}

return false;

}

bool IsEmpty()

{

return head == nullptr;

}

int Count()

{

return size;

}

void Clear()

{

Node<T>\* ptr = head;

Node<T>\* next = nullptr;

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

{

next = head->next;

delete head;

head = next;

}

size = 0;

}

~LinkedList()

{

Node<T>\* ptr = head;

Node<T>\* next = nullptr;

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

{

next = head->next;

delete head;

head = next;

}

size = 0;

}

};

**Содержимое файла main.cpp:**

#include <iostream>

#include <string>

#include <Windows.h>

#include "LinkedList.h"

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char option;

cout << "\n\t\t\t\t!!!! Для работы необходимо создать список. Создать его?";

cout << "\n\n\t\t\t\t---> Меню <---\n\n";

cout << "\t\t\t\t1 - Создать новый список\n";

cout << "\t\t\t\t2 - Пропустить\n";

cout << "\t\t\t\t---> Номер действия: "; cin >> option;

if (option == '1') {

system("cls");

LinkedList <string> list;

do

{

cout << "\n\n\t\t\t\t---> Меню <---\n\n";

cout << "\t\t\t\t1 - Добавить элемент в конец списка\n";

cout << "\t\t\t\t2 - Вывести список всех элементов\n";

cout << "\t\t\t\t3 - Вставить элемент на указанную позицию\n";

cout << "\t\t\t\t4 - Удалить узел, находящийся на указанной позиции\n";

cout << "\t\t\t\t5 - Получить количество имеющихся узлов\n";

cout << "\t\t\t\t6 - Вывести значение, хранящее в указанном узле\n";

cout << "\t\t\t\t7 - Проверить, пуст ли список\n";

cout << "\t\t\t\t8 - Выход\n";

cout << "\t\t\t\t---> Номер действия: "; cin >> option;

switch (option)

{

case '1': {

string val;

cout << "\n\t\t\t\tВведите элемент для добавления в список: ";

getline(cin >> ws, val); list.PushBack(val); break;

}

case '2': list.Print(); break;

case '3':

{

string val;

int pos;

cout << "\n\t\t\t\tВведите элемент для добавления в список: ";

getline(cin >> ws, val);

if (!list.IsEmpty())

{

cout << "\n\t\t\t\tВведите номер позиции (от 0 до " << list.size - 1 << "): ";

cin >> pos;

list.Add(val, pos);

break;

}

list.PushBack(val);

break;

}

case '4':

{

if (!list.IsEmpty())

{

int pos;

cout << "\n\t\t\t\tВведите номер позиции (от 0 до " << list.size - 1 << "): "; cin >> pos;

list.Cut(pos);

break;

}

cout << "\n\t\t\t\tСписок пуст\n";

break;

}

case '5': cout << "\n\t\t\t\tУзлов имеется: " << list.Count() << "\n"; break;

case '6':

{

int pos;

if (!list.IsEmpty()) {

cout << "\n\t\t\t\tВведите номер позиции (от 0 до " << list.size - 1 << "): "; cin >> pos;

cout << "\n\t\t\t\tЗначение, хранящееся в узле: " << list.GetValue(pos) << "\n";

break;

}

cout << "\n\t\t\t\tСписок пуст\n";

break;

}

case '7':

{

if (list.IsEmpty()) cout << "\n\t\t\t\tСписок пустой.\n";

else cout << "\n\t\t\t\tСписок не является пустым.\n";

break;

}

default: option = '8';

}

} while (option != '8');

system("cls");

}

else system("cls");

cout << "\n\t\t\t\t!!!! Работа завершена успешно!\n";

return 0;

}

## РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

Демонстрационный пример работы функции для добавления узла в конец списка и вывода количества узлов в списке:

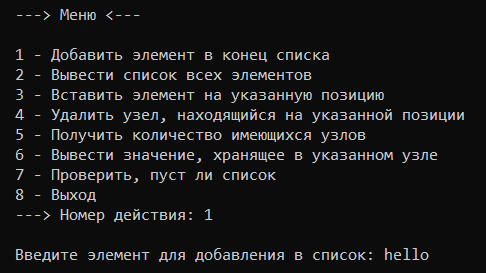


Рисунок 1 – Ввод нового элемента

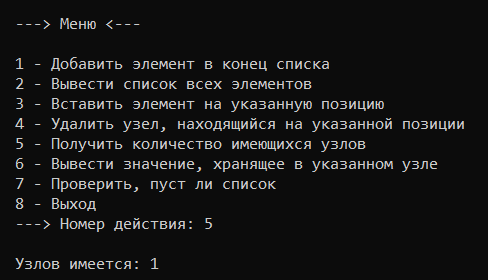


Рисунок 2 – Вывод количества узлов

Демонстрационный пример работы функции для добавления узла на указанную позицию и вывода списка всех узлов:

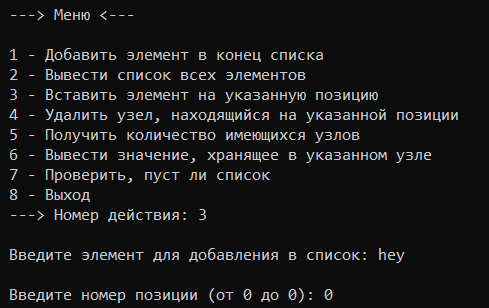


Рисунок 3 – Добавление узла на указанную позицию

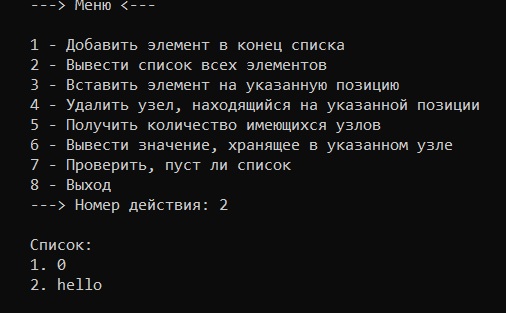


Рисунок 4 – Вывод списка всех узлов и хранящихся в них значений

Демонстрационный пример работы функции для проверки, является ли список пустым:

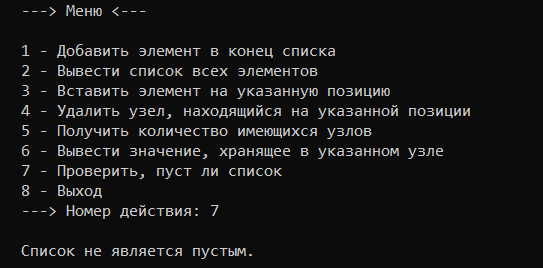


Рисунок 5 – Проверка, является ли список пустым

Демонстрационный пример работы функции для вывода значения по номеру узла:

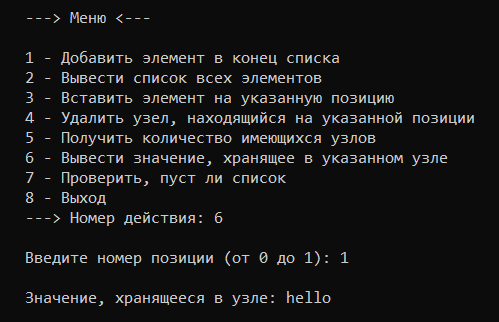


Рисунок 6 – Вывод значения по номеру узла

Демонстрационный пример работы функции для удаления узла с указанной позиции:

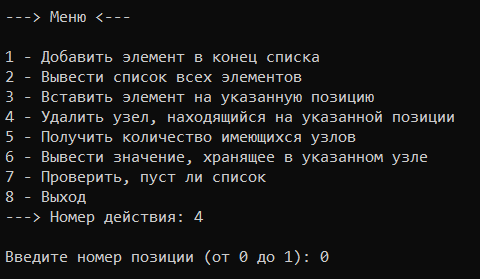


Рисунок 7 – Удаление узла с указанной позиции

# ВЫВОД

В ходе выполнения лабораторной работы были изучены основные понятия ООП: «объект», «класс», «инкапсуляция», способы описания классов и объектов в языке С++, возможности перегрузки операторов и использования конструкторов объектов класса. Для практического применения изученных понятия было разработано приложение по варианту.