# Министерство науки и высшего образования Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра прикладной математики

### Лабораторная работа № 1 по дисциплине «Структуры данных и алгоритмы»

|  |  |  |
| --- | --- | --- |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-82 |
| Студенты: | Хайдаев К.Е. |
| Преподаватель: | Еланцева И. Л. |
|  |  |
|  |  |
|  |  |

***Цель:*** изучить тип указатель; получить навыки в организации и об-работке однонаправленных списков.

1. **Условие задачи**

Даны действительные числа x1, x2, ..., xn (n >= 2 и заранее не-известно). Получить последовательность (x1 – xn), (x2 – xn), ..., (xn-1 – xn).

1. **Анализ задачи**

* Дано:

Действительные числа, заданные случайно, количество элементов в списке неизвестно (>=2);

* Результат:

Вывод на экран последовательности (x1 – xn), (x2 – xn), ..., (xn-1 – xn).

* Метод решения:

Создаем список, предоставляем пользователю ввод количества элементов списка, выводим последовательность, чистим память, выделенную под список.

1. **Структура основных входных и выходных данных**

* Внешние входные данные: количество элементов в списке – переменная size.

Внутреннее представление входных данных: Линейный список, struct list{list\* pNext; float data; int size = 0; } \*head = nullptr;

Внешние выходные данные: последовательность (x1 – xn), (x2 – xn), ..., (xn-1 – xn).

1. **Алгоритм решения задачи**

***main – основная функция:***

подключаем заголовочный файл List.h;

подключаем русский язык;

создаем указатель типа list и присваиваем адрес начала списка;

позволяем пользователю ввести количество элементов списка, учитывая что число должно быть больше 2;

в цикле for вызываем функцию at, создаем элементы и заполняем список случайными числами; выводим список в консоль;

с помощью цикла for выводим последовательность : at(i)-at(size-1);

***at(int nomer)-функция для обращения к nomer элементу списка:***

создаем указатель \*p типа list и присваиваем адрес начала списка;

с помощью цикла for присваиваем к указателю \*p номер соответствующий вызванному элементу;

возвращаем данные, которые содержит данный элемент.

***pop\_back()- удаляет последний элемент списка:***

создаем указатель \*p типа list и присваиваем адрес начала списка;

с помощью цикла for присваиваем к указателю \*p номер соответствующий вызванному элементу;

уменьшаем значение количества элементов списка head->size--;

удаляем последний элемент delete p;

***list clear() - удаление списка и освобождение памяти:***

с помощью цикла while вызываем функцию erase(0) и удаляем каждый 0 элемент списка пока head не будет указывать на nullptr;

***push\_back() – создает элемент списка:***

проверяем есть ли в списке первый элемент ( если указатель head указывает на nulptr, значит в списке еще нет элементов и нам нужно создать первый элемент и присвоить в этот элемент данные);

если в списке уже есть элементы, тогда мы находим последний элемент списка, затем создать новый элемент, присвоить указателю его адрес и занести данные в элемент;

затем прибавляем счетчик размера списка.

1. **Текст программы**

**List.h**

#pragma once

#include <iostream>

#include <ctime>

#include <cstdlib>

using namespace std;

struct list

{

list\* pNext;

float data;

int size = 0;

} \*head = nullptr;

//возвращает данные указанного элемента

float at(const int nomer)

{

if (nomer < head->size) {

list\* p = head;

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

{

p = p->pNext;

}

return p->data;

}

}

// удаляет последний элемент

void pop\_back()

{

if (head->size > 1)

{

list\* p = head;

for (int i = 1; i < head->size; i++)

p = p->pNext;

head->size--;

delete p;

p = nullptr;

}

else

{

head->size--;

delete head;

head = nullptr;

}

}

// удаляет выбранный элемент

bool erase(const int nomer)

{

if (head == nullptr)

{

return false;

}

else if (nomer == 0 && head->size == 1)

{

delete head;

head = nullptr;

return true;

}

else if (nomer == 0)

{

list\* p = head->pNext;

p->size = --head->size;

delete head;

head = p;

return true;

}

else if (nomer == head->size - 1)

{

list\* p = head;

for (int i = 1; i < nomer - 1; i++)

{

p = p->pNext;

if (p == nullptr)

{

return false;

}

}

list\* temp = p->pNext;

p->pNext = nullptr;

delete temp;

temp = nullptr;

head->size--;

return true;

}

else

{

list\* p = head;

for (int i = 1; i < nomer-1; i++)

{

p = p->pNext;

if (p == nullptr)

{

return false;

}

}

list\* temp = p->pNext;

p->pNext = (p->pNext->pNext);

delete temp;

temp = nullptr;

head->size--;

return true;

}

}

// чистит список

void list\_clear()

{

while (head != nullptr)

{

erase(0);

}

}

// добавляет элемент в конец списка

void push\_back(const float data)

{

if (head == nullptr)

{

head = new list;

head->pNext = nullptr;

head->data = data;

}

else

{

list\* p = head;

while (p->pNext != nullptr)

{

p = p->pNext;

}

p->pNext = new list;

p = p->pNext;

p->data = data;

p->pNext = nullptr;

}

head->size++;

}

int get\_size() {

if (head != nullptr)

return head->size;

else

return 0;

}

// добавляет перед nomer новый элемент

bool insert(const float data, const int nomer)

{

if (head == nullptr)

{

head = new list;

head->pNext = nullptr;

head->data = data;

head->size++;

return true;

}

else if (nomer==0)

{

list\* p = head;

head = new list;

head->pNext = p;

head->data = data;

head->size = p->size;

head->size++;

return true;

}

else

{

list\* p = head;

for (int i = 1; i < nomer-1; i++)

{

p = p->pNext;

if (p == nullptr)

{

push\_back(data);

return false;

}

}

list\* temp = new list;

temp->pNext = p->pNext;

p->pNext = temp;

temp->data = data;

head->size++;

return true;

}

}

// возвращает первый элемент списка

float first()

{

if(head!=nullptr)

return at(0);

}

// возвращает последний элемент списка

float last() {

if (head != nullptr)

return at(head->size - 1);

}

**source.cpp**

#include "List.h"

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "ru");

int size;

do

{

cout << "введите количество элементов в листе(должно быть 2 или больше элементов):\n";

cin >> size;

} while (size <= 1);

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

{

push\_back(rand() % 20);

cout << at(i) << "\t";

}

cout << endl<< "вывод последовательности : "<<endl;

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

{

cout << at(i) - at(size - 1) << "\t";

}

return 1;

}

1. **Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Примечание |
| 1. | Размер списка: 2 | 1 1  вывод последовательности :  0 | Проверка работы программы |
| 2. | Размер списка: 4 | 1 1 7 14  вывод последовательности :  -13 -13 -7 | Проверка работы программы |

1. **Результат отладки и их анализ:**

Программа выдала верный результат на всех тестах, значит задача решена.

