Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

СПИСКИ

Отчёт по лабораторной работе №3

По дисциплине

«Структуры и алгоритмы обработки данных в ЭВМ»

Выполнил: студент гр. 439-3

_____ Мазовец А. С.

«___»_____ 2020 г.

Проверил: ассистент каф. АСУ

_____ Яблонский Я. В.

«___»_____ 2020 г.

Томск 2020

1 Задание на лабораторную работу

Вариант 12.

Подготовить текстовый файл, содержащий не менее 10 целых чисел. Прочитать данные из этого файла и сформировать список. Вычислить произведение сумм:

$$S = (x_1^2 - x_n^2)(x - x_{n-1}^2)\ldots(x_n^2 - x_1^2)$$

Вывести на экран исходный список и результаты вычислений. После завершения работы со списком освободить занимаемую им динамическую память.

2 Алгоритм решения задачи

1. Прочитать список из файла;
2. Вывести на экран исходный список;
3. В цикле от *1* до *n* считать разности квадратов элементов списка *L[i]* и *L[n-i]*
4. Очередной результат вычисления перемножать в дополнительную переменную *S;*
5. Вывести на экран *S* — результат вычислений и освободить память, занимаемую списком.

3 Листинг программы

Реализация класса АТД линейный список. Вспомогательный класс node.

asmazovec@case insert ./sem3.СиАОДвЭВМ.lab3 > cat list.hpp

```
namespace ads {
    template <typename T>
        class _node {
            public:
                T field;
                _node *next;
                _node (T data = 0);
                _node (const _node &) = delete;
                _node (_node &&n);
                _node &operator= (const _node &) = delete;
                _node &operator= (_node &&n);
                _node &operator= (T data);
                friend std::ostream& operator<< <> (std::ostream& o,
                                                    const _node<T> &n);
        };

    template <typename T>
    _node<T>::_node (T data): field (data), next(nullptr) {}

    template <typename T>
    _node<T>::_node (_node &&n)
        :field (n.field), next (n.next) {
        std::cout << "move clone node " << std::endl;
        n.next = nullptr;
    }

    template <typename T>
    _node<T> &_node<T>::operator= (_node<T> &&n) {
        std::cout << "move operator= node " << std::endl;
        if (&n == this)
            return *this;

        field = n.field;
        next = n.next;
        n.next = nullptr;
        return *this;
    }

    template <typename T>
    _node<T> &_node<T>::operator= (T data) {
        field = data;
        return *this;
    }

    template <typename T>
    std::ostream& operator<< (std::ostream& o, const _node<T> &n) {
        o << n.field;
        return o;
    }
}
```

Реализация класса АТД линейный список.

asmazovec@case insert ./sem3.СиАОДвЭВМ.lab3 > cat list.hpp

```
namespace ads {
    template <typename T>
    class list {
        private:
            mutable _node<T> node;
            _node<T> *end;
            _node<T> &getNode (int index) const;

        public:
            list();
            ~list();
            T &operator[] (int index);
            T operator[] (int index) const;
            list &push (T data, int index);
            list &pushStart (T data);
            list &pushBack (T data);
            T pop (int index);
            T popStart();
            T popBack();
            int len() const;
            bool isEmpty() const;
            friend std::ostream& operator<< <> (std::ostream& o,
                                                const list<T> &l);
    };

template <typename T>
    list<T>::list(): node (_node<T>()) {
        end = &node;
    }

    template <typename T>
    list<T>::~list() {
        _node<T> *cur = node.next;
        _node<T> *next;
        while (cur) {
            next = cur->next;
            delete cur;
            cur = next;
        }
        end = nullptr;
    }

    template <typename T>
    _node<T> &list<T>::getNode (int index) const {
        if (++index < 0)
            throw std::invalid_argument ("Invalid list index.");
        _node<T> *cur = &node;
        for (int i = 0; i < index; i++)
            if (cur->next)
                cur = cur->next;
            else
               throw std::out_of_range ("Index is out of range.");
        return *cur;
    }
```

```
template <typename T>
T &list<T>::operator[] (int index) {
    if (index < 0)
        throw std::invalid_argument ("Invalid list index.");
    if (isEmpty())
        throw std::out_of_range ("List is empty.");
    _node<T> *n = &getNode (index);
    return n->field;
}

template <typename T>
T list<T>::operator[] (int index) const {
    if (index < 0)
        throw std::invalid_argument ("Invalid list index.");
    if (isEmpty())
        throw std::out_of_range ("List is empty.");
    _node<T> *n = &getNode (index);
    return n->field;
}

template <typename T>
list<T> &list<T>::push (T data, int index) {
    if (index < 0)
        throw std::invalid_argument ("Invalid list index.");
    _node<T> *n = &getNode (index - 1);
    _node<T> *cur = new _node<T> (data);
    cur->next = n->next;
    n->next = cur;
    if (!cur->next)
        end = cur;
    return *this;
}

template <typename T>
list<T> &list<T>::pushStart (T data) {
    push (data, 0);
    return *this;
}

template <typename T>
list<T> &list<T>::pushBack (T data) {
    _node<T> *cur = new _node<T> (data);
    cur->next = nullptr;
    end->next = cur;
    end = cur;
    return *this;
}

template <typename T>
T list<T>::pop (int index) {
    if (index < 0)
        throw std::invalid_argument ("Invalid list index.");
    if (isEmpty())
        throw std::out_of_range ("List is empty.");
    _node<T> *n = &getNode (index - 1);
    int data = n->next->field;
    if (n->next->next)
        end = n;
    _node<T> *next = n->next->next;
```

```
        delete n->next;
        n->next = next;
        return data;
    }

    template <typename T>
    T list<T>::popStart() {
        return pop (0);
    }

    template <typename T>
    T list<T>::popBack() {
        return pop (len() - 1);
    }

    template <typename T>
    int list<T>::len() const {
        _node<T> *cur = &node;
        int len = 0;
        while (cur->next) {
            cur = cur->next;
            len++;
        }
        return len;
    }

    template <typename T>
    bool list<T>::isEmpty() const {
        return !node.next;
    }

    template <typename T>
    std::ostream& operator<< (std::ostream& o, const list<T> &l) {
        _node<T> *cur = &l.node;
        while (cur->next) {
            cur = cur->next;
            o << *cur << " ";
        }
        return o;
    }
}
```

asmazovec@case insert ./sem3.СиАОДвЭВМ.lab3 > cat main.cpp

```
#include "list.hpp"
#include <iostream>
#include <fstream>
#include <stdexcept>

using namespace ads;


void task() {
    list<int> inputL;

    /* чтение списка из файлов */
    int i;
    std::fstream file;

    file.open("I");
    while (file >> i)
        inputL.pushBack (i);
    file.close();

    std::cout << inputL << std::endl;

    int n = inputL.len();
    long S = 0;
    if (n) S = 1;

    int xi0, xi1;
    for (int i = 0; i < n; i++) {
        xi0 = inputL[i];
        xi1 = inputL[n - i - 1];
        std::cout << xi0*xi0 - xi1*xi1 << std::endl;
        S *= xi0*xi0 - xi1*xi1;
    }

    std::cout << S << std::endl;
    return ;
}


int main() {
    try {
        task();
    } catch (std::exception &exception) {
        std::cerr << "Standard exception: " << exception.what() <<
std::endl;
    } catch (...) {
          std::cerr << "Undetermined exception" << std::endl;
    }
    return 0;
}
```

4 Пример решения

Тест 1. Входные данные:

$$I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Ожидаемый результат работы программы — большое отрицательное число:

$$S = -81 * -63 * -45 * -27 * -9 * 9 * 27 * 45 * 63 * 81$$

Пример работы программы (рисунок 4.1 — Тест 1):

```
0 1 2 3 4 5 6 7 8 9
-81
-63
-45
-27
-9
9
27
45
63
81
-3113785639703025
```

Рисунок 4.1 — Тест 1.

5 Вывод

Была изучена АТД линейный список с динамически выделяемой памятью. Был реализован класс list, реализующий функционал и структуры данных изученной АТД. Решена поставленная задача.