Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Алтайский государственный технический университет
им. И.И. Ползунова»
Факультет (институт) Информационных технологий
Кафедра Прикладная математика

Отчет защищен с оценкой_____
А.И.Потупчик
(подпись преподавателя) (инициалы, фамилия)
“____”_____ 2024 г.

Отчет
по лабораторной работе №4
Связные списки. Библиотека стандартных шаблонов (STL)
(название лабораторной работы)

по дисциплине_____Типы и структуры данных_____
(наименование дисциплины)

_____ ЛР 09.03.04.13.000 ОТ_____
(обозначение документа)

Студент группы ПИ-21_____А.А.Лихтинфельд_____
(инициалы, фамилия)
Преподаватель доцент, доцент_____А.И.Потупчик_____
(должность, ученое звание) (инициалы, фамилия)

Барнаул 2024

Вариант №1

Написать функцию, которая по списку L строит два новых списка: L1– из положительных элементов и L2 – из отрицательных элементов списка L.

Задание 1:

Разработать и отладить программу на языке C++, реализующую работу со списком в соответствии с вариантом. Выполнить оценку временной и емкостной сложности программы.

Код программы:

```
#include <iostream>
#include <string>
#include <locale>
using namespace std;
struct Node {
    double Value; // Информационное поле
    Node* Next;//адресное поле
};
//создание однонаправленного списка
void Make_Single_List(Node** Head) {
    Node* Tail = *Head;
    string input;
    double value;
    do {
        cout << "Введите значение (для завершения введите 'end'): ";
        cin >> input;
        if (input == "end") {
            break; // завершаем цикл при вводе "end"
        }
        try {
            value = stod(input); // Преобразование введенной строки в double
        }
        catch (...) {
            cout << "Некорректный ввод. Попробуйте еще раз." << endl;
```

```
                continue;
        }

        Node* NewNode = new Node;
        NewNode->Value = value;
        NewNode->Next = NULL;
        if (*Head == NULL) {
                *Head = NewNode;
                Tail = *Head;
        }
        else {
                Tail->Next = NewNode;
                Tail = Tail->Next;
        }
    } while (true);
}
//печать однонаправленного списка
void Print_Single_List(Node * Head) {
    if (Head != NULL) {
        cout << Head->Value << " ";
        Print_Single_List(Head->Next);
        //переход к следующему элементу
    }
    else cout << "\n";
}
//добавление в конец списка
void Insert_End(Node** Head, double Number) {
    Node* NewValue = new Node;
    NewValue->Value = Number;
    NewValue->Next = NULL;
    if (*Head == NULL) {
        *Head = NewValue;
    }
```

```
    else {
        Node* Current = *Head;
        while (Current->Next != NULL) {
            Current = Current->Next;
        }

        Current->Next = NewValue; // вставляем новый элемент в конец списка
    }
}
// освобождение памяти, выделенной под однонаправленный список
void Delete_Single_List(Node* Head) {
    if (Head != NULL) {
        Delete_Single_List(Head->Next);
        delete Head;
    }
}
//функция разбиения исходного списка на два - с + и - значениями
void Search(Node* L, Node** L1, Node** L2) {
    Node* Current = L;
    while (Current != NULL) {
        if (Current->Value >= 0) {
            Insert_End(L1, Current->Value); // добавляем положительный элемент в L1
        }
        else {
            Insert_End(L2, Current->Value); // добавляем отрицательный элемент в L2
        }
        Current = Current->Next;
    }
}
int main() {
    setlocale(LC_ALL, "Russian");
    Node* L = NULL;
    Make_Single_List(&L); // Ввод значений списка L
```

```
    cout << "Изначальный список L:\n";
    Print_Single_List(L);
    Node* L1 = NULL; // список положительных элементов
    Node* L2 = NULL; // список отрицательных элементов
    Search(L, &L1, &L2); // Разделение списка L на списки L1 и L2
    cout << "Список положительных элементов L1:\n";
    Print_Single_List(L1);
    cout << "Список отрицательных элементов L2:\n";
    Print_Single_List(L2);
    // Освобождение памяти
    Delete_Single_List(L);
    Delete_Single_List(L1);
    Delete_Single_List(L2);
    return 0;
}
```

Оценки сложности программы

временная сложность программы: O(n)

ёмкостная сложность: O(2n)

где n - кол-во вводимых чисел

Пример выполнения

```
Введите значение (для завершения введите 'end'): 123,56
Введите значение (для завершения введите 'end'): -89
Введите значение (для завершения введите 'end'): -35,5
Введите значение (для завершения введите 'end'): 4
Введите значение (для завершения введите 'end'): end
Изначальный список L:
123.56 -89 -35.5 4
Список положительных элементов L1:
123.56 4
Список отрицательных элементов L2:
-89 -35.5
```

\

Задание 2:

Разработать и отладить программу на языке C++, реализующую работу со списком в соответствии с вариантом, используя библиотеку стандартных шаблонов (STL).

Код программы:

```
#include <iostream>
#include <string>
#include <locale>
#include <forward_list>
using namespace std;
void Make_Forward_List(forward_list<double>& L) {
    string input;
    do {
        cout << "Введите значение (для завершения введите 'end'): ";
        cin >> input;
        if (input == "end") {
            break;
        }
        try {
            double value = stod(input);
            L.push_front(value);
        }
        catch (...) {
            cout << "Некорректный ввод. Попробуйте еще раз." << endl;
        }
    } while (true);
}
void Print_Forward_List(const forward_list<double>& L) {
    for (const auto& value : L) {
        cout << value << " ";
    }
    cout << endl;
}
void Search(const forward_list<double>& L, forward_list<double>& L1, forward_list<double>& L2) {
    for (const auto& value : L) {
        if (value >= 0) {
            L1.push_front(value);
```

```
        }
        else {
            L2.push_front(value);
        }
    }
}
int main() {
    setlocale(LC_ALL, "Russian");
    forward_list<double> L;
    Make_Forward_List(L);
    cout << "Изначальный список L:\n";
    Print_Forward_List(L);
    forward_list<double> L1;
    forward_list<double> L2;
    Search(L, L1, L2);
    cout << "Список положительных элементов L1:\n";
    Print_Forward_List(L1);
    cout << "Список отрицательных элементов L2:\n";
    Print_Forward_List(L2);
    return 0;
}
```

Оценки сложности программы

временная сложность программы: O(n)

ёмкостная сложность: O(2n)

где n - кол-во вводимых чисел

Пример выполнения

```
Введите значение (для завершения введите 'end'): 123,56
Введите значение (для завершения введите 'end'): -89
Введите значение (для завершения введите 'end'): -35,5
Введите значение (для завершения введите 'end'): 4
Введите значение (для завершения введите 'end'): end
Изначальный список L:
123.56 -89 -35.5 4
Список положительных элементов L1:
123.56 4
Список отрицательных элементов L2:
-89 -35.5
```