

Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного
обеспечения и администрирования
информационных систем
Направление подготовки
математическое обеспечение и
администрирование
информационных систем
Форма обучения очная

Отчет

по лабораторной работе №4
«Линейные списки»

Выполнил:

студент группы 213

Тихонов Е.Е.

Проверил:

профессор кафедры ПОиАИС

Кудинов В.А.

Курск, 2021

Цель работы: Научиться составлять процедуры (функции), реализующие операции для работы с линейными списками.

Задание

1. Общие задания:

- a) Объединение двух двусвязных списков.
- b) Пересечение двух двусвязных списков.

2. Индивидуальное задание:

Реализовать алгоритм поиска простых чисел в диапазоне от 2 до 100 методом решета Эратосфена, используя циклический односвязный список с заглавным звеном.

Решение задания 1

Двусвязный список представлен шаблонным классом List<T>. Класс List использует вспомогательный класс Node<T>. Node - класс представляющий из себя «узел» который хранит данные, а также указатели на предыдущий узел в списке и следующий узел. List содержит поля указатели на первый и на последний узел(Node) списка. Также List содержит методы для добавления, удаления и поиска определенного узла. В List определены бинарные операции объединения (operator |) и пересечения (operator &) двух списков.

Текст программы для решения задания 1

```
#include <iostream>
#include <string>

using namespace std;

template <typename DataType>
class Node;

template <typename T>
class List
{
    friend class Node<T>;
    Node<T>* First;
    Node<T>* Last;
```

```
int Count = 0;
```

```
public:
```

```
int getCount()  
{  
    return Count;  
}
```

```
List()  
{  
    First = nullptr;  
    Last = nullptr;  
}
```

```
List(T* Array, int Count): List()  
{  
    for (size_t i = 0; i < Count; i++)  
    {  
        Add(Array[i]);  
    }  
}
```

```
List(List<T>& ListObj): List()  
{  
    int Count = ListObj.getCount();  
    for (size_t i = 0; i < Count; i++)  
    {  
        Add(ListObj.At(i));  
    }  
}
```

```
void Add(T Element)  
{  
    Node<T>* temp = new Node<T>(Element);  
  
    if (First == nullptr)  
    {
```

```

        First = temp;
        Last = temp;
    }
    else
    {
        Last->Next = temp;
        temp->Prev = Last;
        Last = temp;
    }
    Count++;
}

```

```

private: Node<T>* NodeAt(int Indx)
{
    Node<T>* Elem = First;
    for (size_t i = 0; i < Indx; i++)
    {
        Elem = Elem->Next;
    }
    return Elem;
}

```

```

public:
    T At(int Indx)
    {
        return NodeAt(Indx)->Data;
    }

```

```

void Remove(int Indx)
{
    Node<T>* Elem = NodeAt(Indx);
    Node<T>* Before = Elem->Prev;
    Node<T>* After = Elem->Next;

    if (Before != nullptr)
    {
        Before->Next = After;
    }
}

```

```

    }
    else
    {
        First = After;
    }

    if (After != nullptr)
    {
        After->Prev = Before;
    }
    else
    {
        Last = Before;
    }

    delete Elem;
    Count--;
}

bool IsEmpty()
{
    if (First == nullptr)
    {
        return true;
    }
    return false;
}

void ToString()
{
    Node<T>* Elem = this->First;
    while (Elem != nullptr)
    {
        cout << Elem->Data << " ";
        Elem = Elem->Next;
    }
    cout << endl;
}

```

```
}
```

```
List<T>* operator & (List<T> List2)
{
    List<T> newList;
    Node<T>* Node1 = this->First;
    for (int i = 0; i < this->Count; i++)
    {
        Node<T>* Node2 = List2.First;
        for (int j = 0; j < List2.Count; j++)
        {
            if(Node1->Data == Node2->Data)
            {
                newList.Add(Node1->Data);
            }
            Node2 = Node2->Next;
        }
        Node1 = Node1->Next;
    }
    return &newList;
}
```

```
List<T>* operator | (List<T> List2)
{
    List<T> newList(*this);

    Node<T>* Node1 = List2.First;
    for (int i = 0; i < List2.Count; i++)
    {
        bool Contains = false;
        Node<T>* Node2 = this->First;
        for (int j = 0; j < this->Count; j++)
        {
            if (Node1->Data == Node2->Data)
            {
                Contains = true;
                break;
            }
        }
    }
}
```

```

        }
        Node2 = Node2->Next;
    }

    if(!Contains)
    {
        newList.Add(Node1->Data);
    }

    Node1 = Node1->Next;
}
return &newList;
}
};

```

```

template <typename DataType>
class Node
{
    friend class List<DataType>;
    DataType Data;
    Node* Next;
    Node* Prev;
    Node(DataType Data = 0)
    {
        this->Data = Data;
        this->Next = nullptr;
        this->Prev = nullptr;
    }
};

```

```

int main()
{
    int* arr1 = new int[5]{ 1, 2, 3, 4, 5 };
    int* arr2 = new int[6]{ 4, 5, 6, 7, 8, 9 };

    List<int> list1(arr1, 5);
    List<int> list2(arr2, 6);
}

```

```

cout << "list 1: ";
list1.ToString();

cout << "list 2: ";
list2.ToString();

List<int>* ListAnd = list1 & list2;
cout << "list & : ";
ListAnd->ToString();

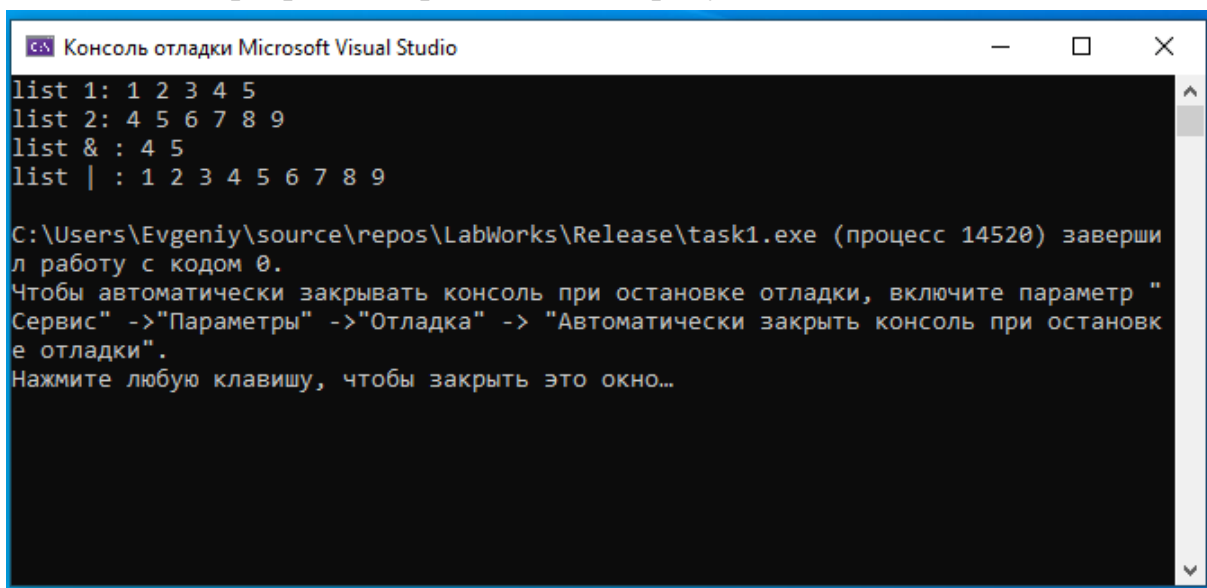
List<int>* ListOr = list1 | list2;
cout << "list | : ";
ListOr->ToString();

return 0;
}

```

Тест программы для решения задачи 1

Вывод программы представлен на рисунке 1.



```

Консоль отладки Microsoft Visual Studio
list 1: 1 2 3 4 5
list 2: 4 5 6 7 8 9
list & : 4 5
list | : 1 2 3 4 5 6 7 8 9

C:\Users\Evgeniy\source\repos\LabWorks\Release\task1.exe (процесс 14520) заверши
л работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "
Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановк
е отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

```

Рисунок 1 - Вывод программы.

Решение задания 2

Циклический список представлен шаблонным классом List<T>. Класс List использует вспомогательный класс Node<T>. Node - класс представляющий из себя «узел» который хранит данные, а также указатели на предыдущий узел в списке и следующий узел. List содержит поле указатель на первый узел(Node) списка. Последний и первый узлы в циклическом списке всегда связываются. То есть следующим для последнего узла будет первый, а предыдущий первому - это последний. В функции main реализован алгоритм решета Эратосфена для чисел от 2 до 100.

Текст программы для решения задания 2

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
template <typename DataType>
```

```
class Node;
```

```
template <typename T>
```

```
class List
```

```
{
```

```
    friend class Node<T>;
```

```
public:
```

```
    Node<T>* First;
```

```
    List()
```

```
    {
```

```
        First = nullptr;
```

```
    }
```

```
    void Add(T Element)
```

```
    {
```

```
        Node<T>* temp = new Node<T>(Element);
```

```
        if (First == nullptr)
```

```
        {
```

```

        First = temp;
        First->Prev = First;
        First->Next = First;
    }
    else
    {
        Node<T>* Last = First->Prev;

        Last->Next = temp;
        First->Prev = temp;

        temp->Prev = Last;
        temp->Next = First;
    }
}

```

```

Node<T>* Remove(Node<T>* Node)
{
    return Node->Delete();
}

```

public:

```

void ToString()
{
    Node<T>* Node = First;
    do {
        cout << Node->Data << " ";
        Node = Node->Next;
    } while (Node != First);
    cout << endl;
}

};

```

```

template <typename DataType>
class Node

```

```

{
    friend class List<DataType>;
public:
    DataType Data;
    Node<DataType>* Next;
    Node<DataType>* Prev;
    Node(DataType Data = 0)
    {
        this->Data = Data;
        this->Next = nullptr;
        this->Prev = nullptr;
    }

private:
    Node<DataType>* Delete()
    {
        Node<DataType>* Before = Prev;
        Node<DataType>* After = Next;

        if (Before != nullptr)
        {
            Before->Next = After;
        }

        if (After != nullptr)
        {
            After->Prev = Before;
        }

        delete this;
        return After;
    }
};

int main()
{
    List<int> list;

```

```

for (int i = 2; i <= 100; i++)
{
    list.Add(i);
}

Node<int>* Node1 = list.First;
do {
    Node<int>* Node2 = Node1->Next;

    while (Node2 != list.First)
    {
        if (Node2->Data % Node1->Data == 0)
        {
            Node2 = list.Remove(Node2);
        }
        else
        {
            Node2 = Node2->Next;
        }
    }
    Node1 = Node1->Next;

} while (Node1 != list.First);

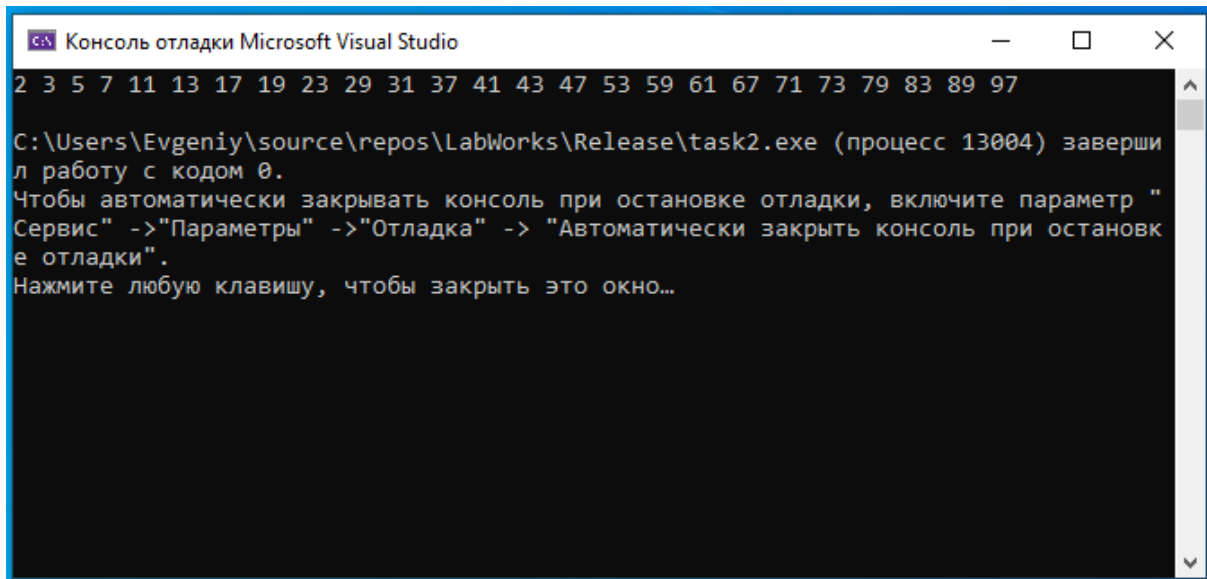
list.ToString();

return 0;
}

```

Тест программы для решения задачи 2

Вывод программы представлен на рисунке 2.



Консоль отладки Microsoft Visual Studio

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

C:\Users\Evgeniy\source\repos\LabWorks\Release\task2.exe (процесс 13004) заверши
л работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "
Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остано
вке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

Рисунок 2 - Вывод программы.