Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский экономический университет имени Г.В. Плеханова»

Высшая школа кибертехнологий, математики и статистики
Кафедра информатики
Направление 38.03.05 Бизнес-информатика
Профиль «Цифровая трансформация бизнеса»

ОТЧЕТ

По выполнению практической работы №2

Выполнила:
студент группы 15.27Д-БИ19/22б
3 курса ВШКМиС
Нгуен Као Бач

Москва-2024

1. **Формулировка задачи варианта 14:**

https://onlinegdb.com/QwikQ_HTL

Дан линейный однонаправленный список L

1. Разработать функцию, которая создает из значений узлов списка L два новых списка: L1 – из положительных элементов массива L; L2 – из остальных элементов списка L.

2. Разработать функцию, которая удаляет из списка L2 все отрицательные элементы.

3. Разработать функцию, которая в списке L1 узел с максимальным значением размещает перед первым узлом.

2.Определение списка операций над списком, которые выявлены в процессе исследования задач дополнительного задания.

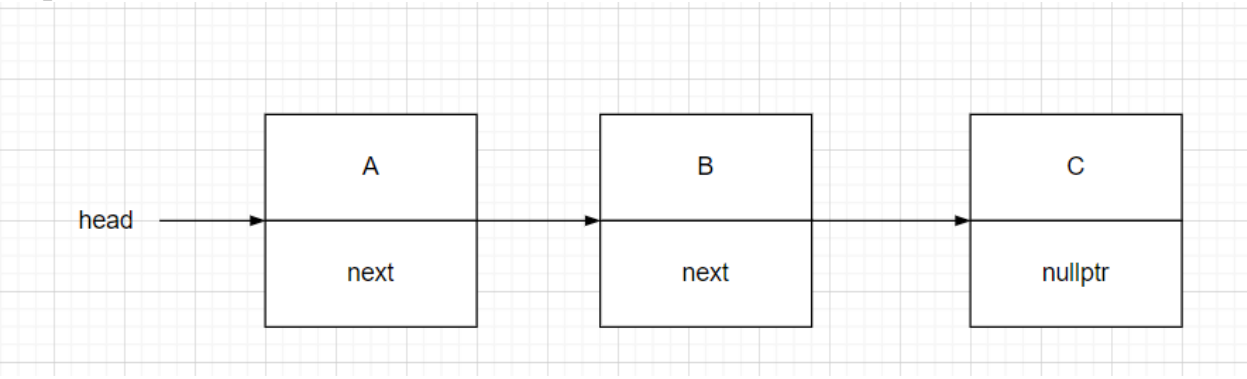
2.1 Определить структуру узла однонаправленного списка в соответствии с вариантом.

```
struct Node {
    int data;
    Node* next;
};
```

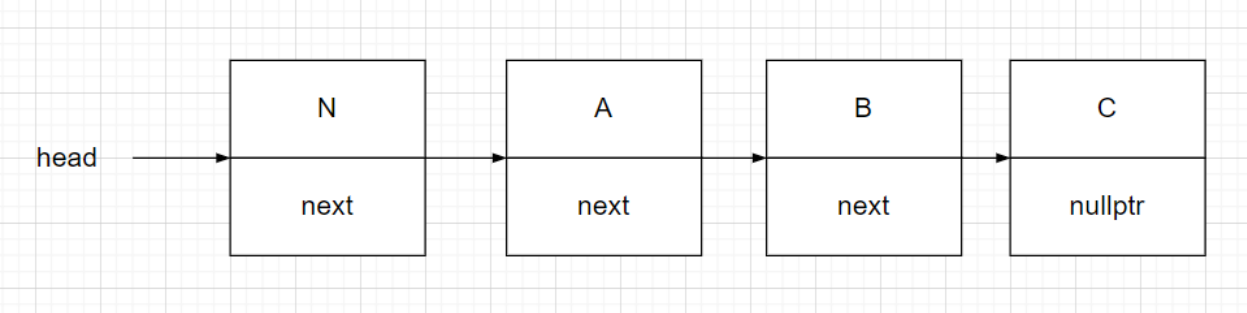
2.2 Изобразить (рисунок) для каждой операции полученного списка процесс выполнения операции на существующем однонаправленном списке.

А. Вставьте кнопку в начало списка

Перед вставкой

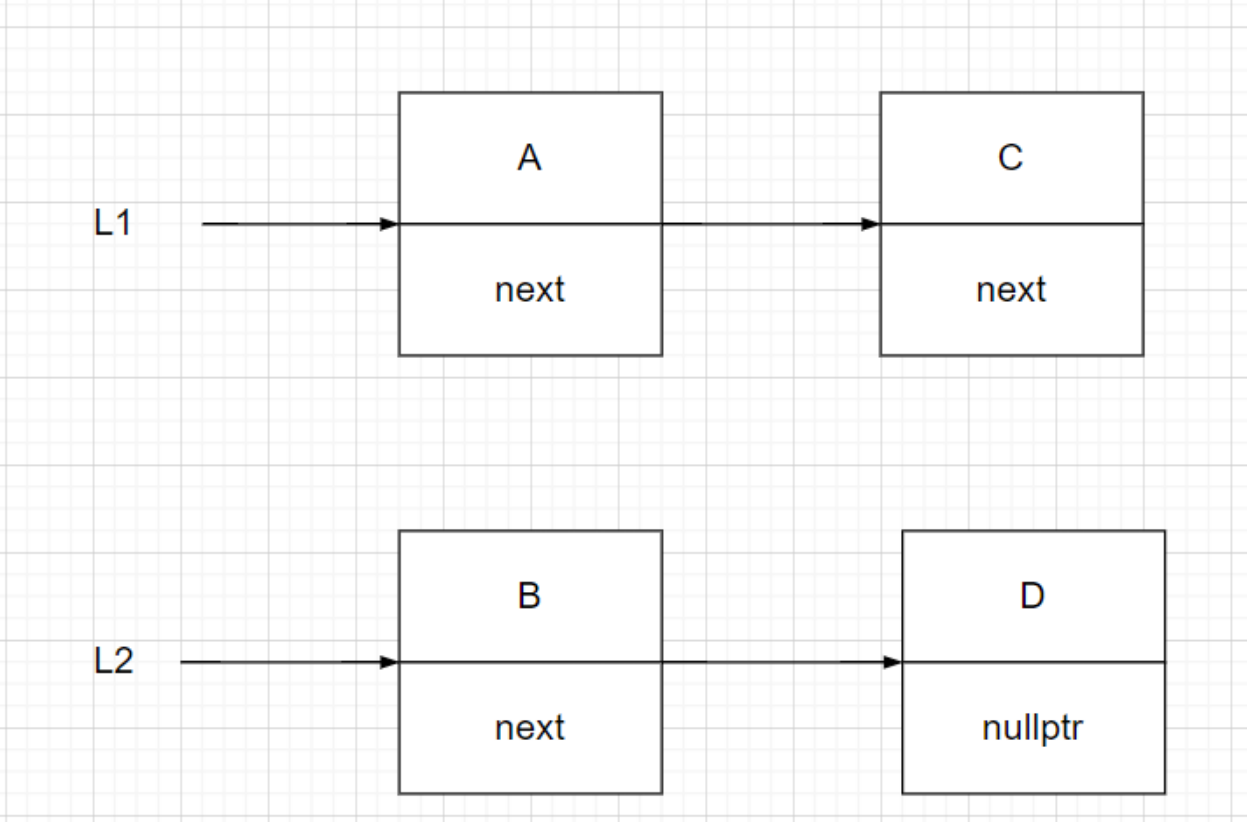


После вставки нового узла N в исходный список

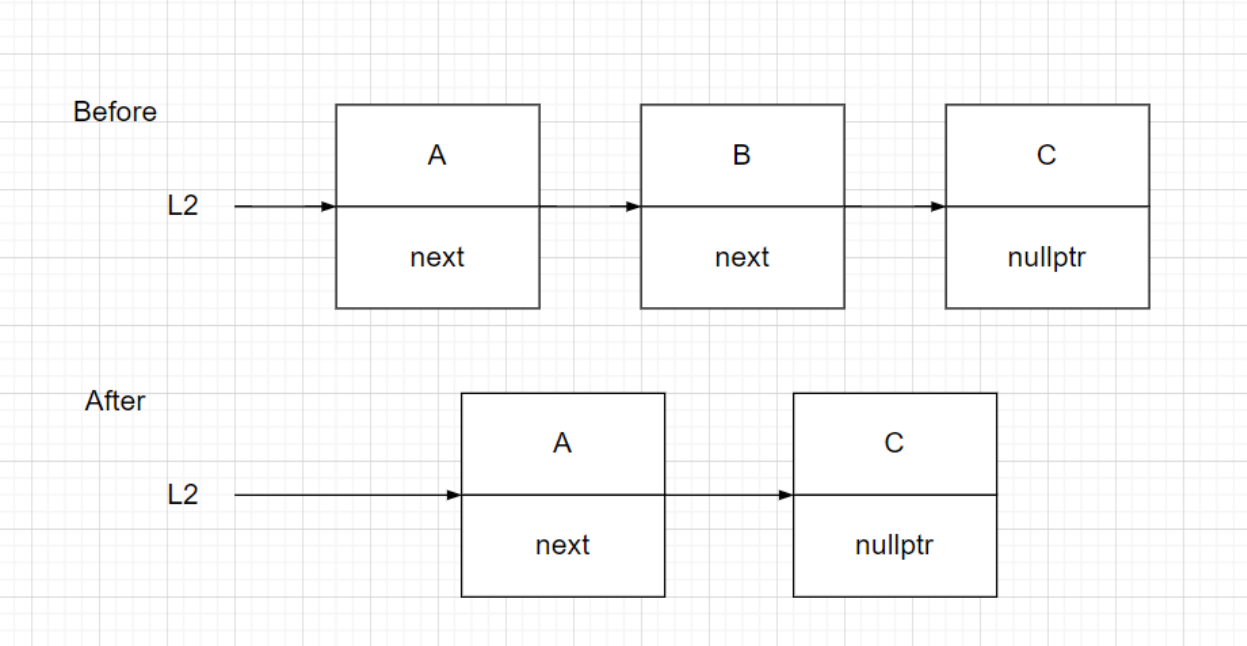


B. Разделение списка на L1 и L2

Разделение исходный список на L1 (содержащий положительные числа) и L2 (оставшиеся элементы). Предположим, что A и C положительны.

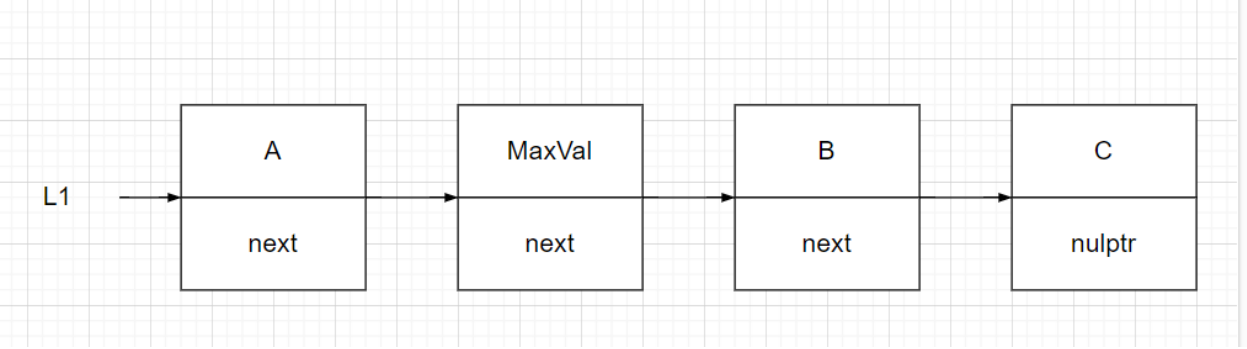


C. Удаление отрицательных элементов из L2: Если B < 0

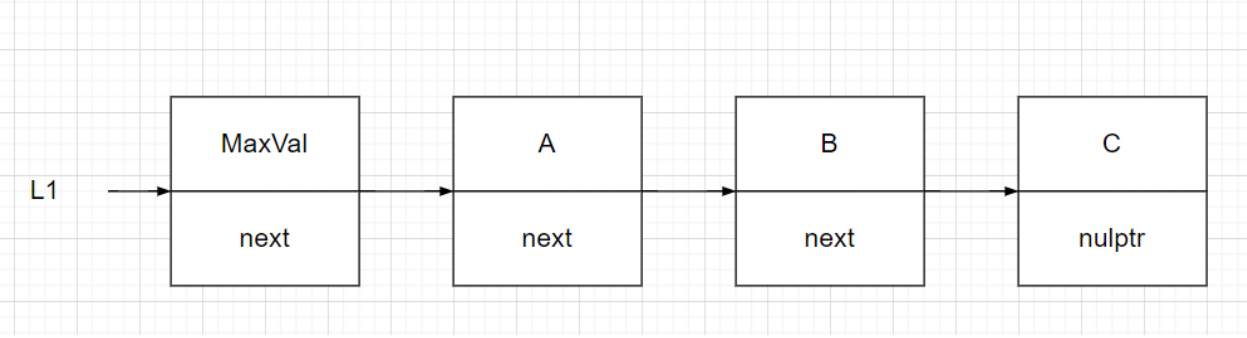


D. Перемещение максимального элемента в начало L1

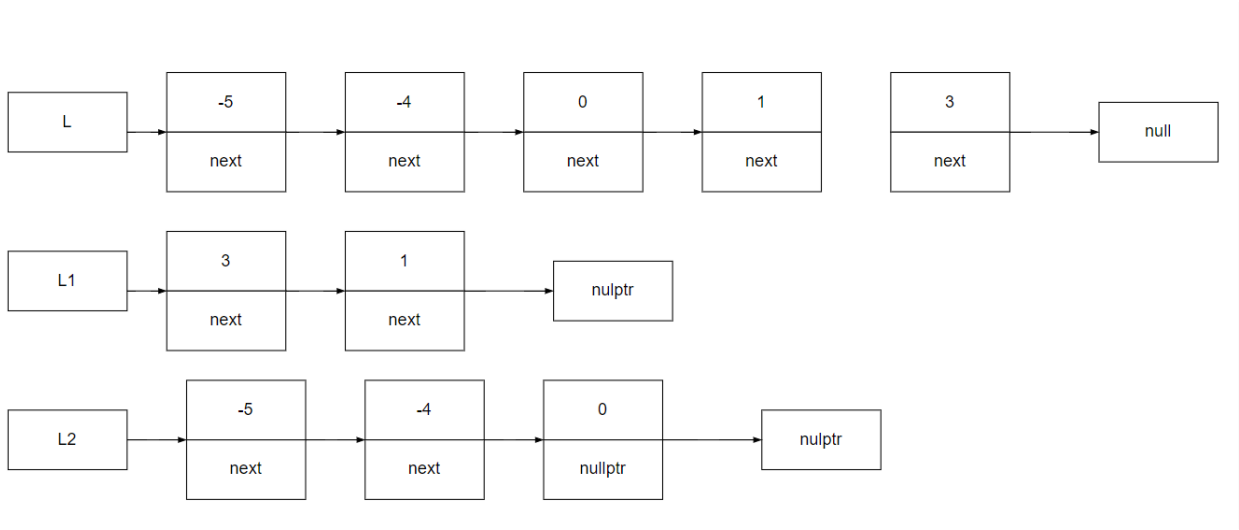
Before



After



2.3 Изобразите структуру данных, которая будет использоваться в операциях.



2.4 Привести алгоритм выполнения операции

1. Функция insertFront - Добавление узла в начало списка

Алгоритм:

Создается новый узел newNode, который содержит значение value и указывает своим полем next на текущий первый узел списка head.

Затем head переназначается на newNode, так что newNode становится новым первым узлом в списке.

Эта функция позволяет нам добавлять элементы в начало связанного списка, что удобно для быстрого ввода данных.

2. Функция deleteByKey - Удаление узла по значению

Алгоритм:

Сначала проверяется, содержит ли head значение key, которое нужно удалить:

Если да, то head обновляется, чтобы указывать на следующий узел, затем temp удаляется.

Если нет, то выполняется поиск узла с ключом key путём обхода списка:

Если узел с ключом найден, то prev->next обновляется для пропуска узла temp, и temp удаляется.

Если key не найден, функция завершает выполнение.

Эта функция удаляет узел из списка на основе значения (ключа), указанного пользователем.

3. Функция splitList - Разделение списка L на L1 и L2

Алгоритм:

Выполняется обход каждого узла в списке L.

Если значение узла положительное, узел добавляется в список L1 (список с положительными значениями) с помощью функции insertFront.

Если значение не положительное (отрицательное или 0), узел добавляется в список L2.

Процесс продолжается до конца списка L.

Эта функция делит исходный список на два новых: один содержит положительные значения, а второй — все остальные значения.

4. Функция removeNegatives - Удаление отрицательных элементов из списка L2

Алгоритм:

Выполняется обход списка L2 с проверкой каждого узла:

Если узел содержит отрицательное значение и является первым, то L2 обновляется для пропуска узла, а сам узел удаляется.

Если узел находится в середине или в конце, то обновляется ссылка prev, чтобы пропустить узел с отрицательным значением, и он удаляется.

Если узел не содержит отрицательного значения, prev и temp перемещаются на следующий узел.

Эта функция помогает удалить все отрицательные значения из списка L2.

5. Функция moveMaxToFront - Перемещение максимального элемента в начало списка L1

Алгоритм:

Сначала maxNode указывается на L1 (предполагается, что первый узел — максимальный).

Выполняется обход списка L1 для поиска узла с наибольшим значением:

Если находится узел с большим значением, maxNode и prevMax обновляются.

Если maxNode не является первым узлом, ссылки обновляются так, чтобы maxNode стал первым узлом в списке L1.

Эта функция находит и перемещает максимальное значение в начало списка L1.

6. Функция printList - Вывод списка на экран

Алгоритм:

Выполняется обход списка от head, печатается значение каждого узла, а затем переходит к следующему узлу.

Печатается nullptr, когда список завершается.

Эта функция позволяет вывести значения списка на экран, чтобы пользователь мог видеть результат.

7. Функция inputList - Ввод списка L от пользователя

Алгоритм:

Просит пользователя ввести каждое значение до тех пор, пока не будет введено x.

Преобразует введённую строку в целое число и добавляет его в начало списка L с помощью функции insertFront.

Эта функция позволяет пользователю самостоятельно вводить значения в список L гибким способом.

8. Функция menu - Отображение и выполнение выбранных действий

Алгоритм:

Отображает меню с опциями для пользователя:

1: Отобразить список L.

2: Разделить список L на L1 и L2.

3: Удалить отрицательные элементы из L2.

4: Переместить максимальный элемент в начало L1.

5: Выйти из программы.

Выполняет функцию, соответствующую выбранному пользователем действию.

Это основная управляющая функция программы, которая позволяет пользователю выполнять запросы на связанный список.

2.5 Привести таблицу тестов для тестирования каждой операции

Тестовый случай	Ввод	Ожидаемый вывод	Описание
1. Ввод списка L	9 10 0 -3 6 8 -7 -5 5 3 x	Список L: 9 -> 10 -> 0 -> -3 -> 6 -> 8 ->-7 -> -5 -> 5 -> 3-> nullptr	Проверка ввода данных и отображения списка.
2. Отображение списка L	1	Список L: 3 -> 5 -> -5 -> -7 -> 8 -> 6 -> -3 -> 0 -> 10 -> 9 -> nullptr	Убедитесь, что список L отображается правильно после ввода.
3. Разделение списка на L1 и L2	2	Список Л1 (положительные элементы):9 -> 10 -> 6 -> 8 -> 5 -> 3 -> nullptr Список L2 (остальные элементы):0 -> -3 -> -7 -> -5 -> nullptr	3. Разделение списка на L1 и L2
4. Удаление отрицательных элементов из L2	3	L2 после удаления отрицательных элементов: 0 -> nullptr	Проверка удаления отрицательных элементов из списка L2.
5. Перемещение максимального элемента в начало L1	4	L1 после перемещения самого большого элемента:10 -> 9 -> 6 -> 8 -> 5 -> 3 -> nullptr	Проверка перемещения максимального элемента в начало списка L1.
6. Выход из программы	5	Выход из программы.	Проверка функции выхода из программы.

3. Представить код программы

```cpp
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

// Функция создает новую кнопку вверху списка.
void insertFront(Node*& head, int value) {
    Node* newNode = new Node{value, head};
    head = newNode;
}

// Функция удаления узла по значению
void deleteByKey(Node*& head, int key) {
    Node* temp = head, *prev = nullptr;
    if (temp != nullptr && temp->data == key) {
        head = temp->next;
        delete temp;
        return;
    }
    while (temp != nullptr && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == nullptr) return;
    prev->next = temp->next;
    delete temp;
}
```

```
31
32  // Функция делит список L на L1 и L2.
33  void splitList(Node* L, Node*& L1, Node*& L2) {
34      while (L != nullptr) {
35          if (L->data > 0)
36              insertFront(L1, L->data);
37          else
38              insertFront(L2, L->data);
39          L = L->next;
40      }
41  }
42
43  // Функция для удаления отрицательных элементов в L2
44  void removeNegatives(Node*& L2) {
45      Node* temp = L2;
46      Node* prev = nullptr;
47
48      while (temp != nullptr) {
49          if (temp->data < 0) {
50              if (temp == L2) {  // Где первый узел имеет отрицательное значение
51                  L2 = temp->next;
52                  delete temp;
53                  temp = L2;
54              } else {
55                  prev->next = temp->next;
56                  delete temp;
57                  temp = prev->next;
58              }
59          } else {
60              prev = temp;
61              temp = temp->next;
62          }
63      }
64  }
```

```
65
66  // Функция перемещает самый большой элемент в начало списка L1.
67  void moveMaxToFront(Node*& L1) {
68      if (L1 == nullptr) return;
69
70      Node *maxNode = L1, *prevMax = nullptr;
71      Node *temp = L1, *prev = nullptr;
72
73      while (temp != nullptr) {
74          if (temp->data > maxNode->data) {
75              maxNode = temp;
76              prevMax = prev;
77          }
78          prev = temp;
79          temp = temp->next;
80      }
81
82      if (maxNode == L1) return;
83
84      if (prevMax != nullptr)
85          prevMax->next = maxNode->next;
86      maxNode->next = L1;
87      L1 = maxNode;
88  }
89
90  // Функция выводит список на экран
91  void printList(Node* head) {
92      while (head != nullptr) {
93          cout << head->data << " -> ";
94          head = head->next;
95      }
96      cout << "nullptr" << endl;
97  }
98
```

```
98
99  //Функция импортирует список L от пользователя.
100 void inputList(Node*& L) {
101     cout << "Введите элементы для списка L (введите «x», чтобы закончить):";
102     while (true) {
103         string input;
104         cin >> input;
105         if (input == "x") break;
106         int value = stoi(input);
107         insertFront(L, value);
108     }
109 }
110
```

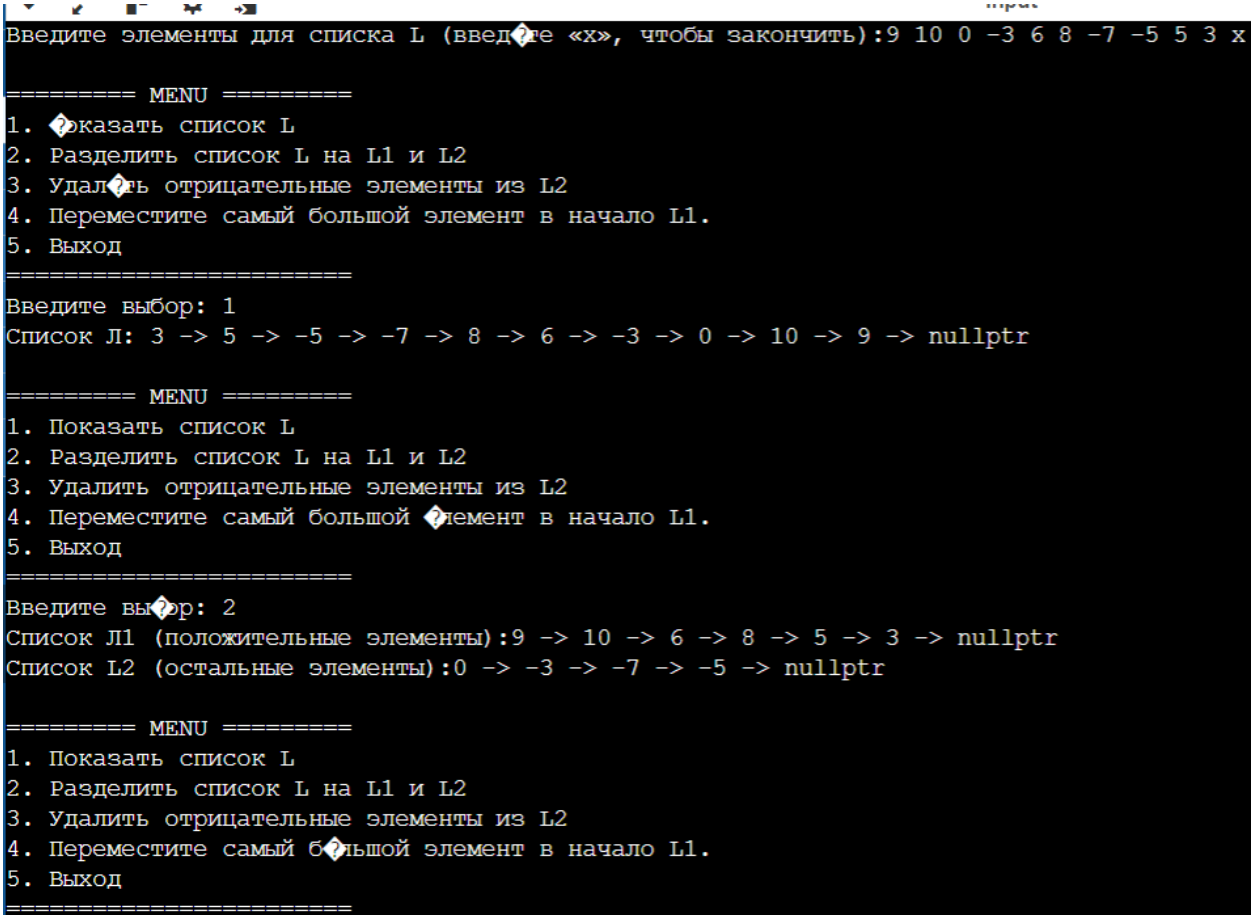
```
// Меню выполняет запросы после импорта списка
void menu(Node* L) {
    Node *L1 = nullptr, *L2 = nullptr;
    int choice;
    do {
        cout << "\n========== MENU =========\n";
        cout << "1. Показать список L\n";
        cout << "2. Разделить список L на L1 и L2\n";
        cout << "3. Удалить отрицательные элементы из L2\n";
        cout << "4. Переместите самый большой элемент в начало L1.\n";
        cout << "5. Выход\n";
        cout << "========================\n";
        cout << "Введите выбор: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Список Л: ";
                printList(L);
                break;
            case 2:
                splitList(L, L1, L2);
                cout << "Список Л1 (положительные элементы):";
                printList(L1);
                cout << "Список L2 (остальные элементы):";
                printList(L2);
                break;
            case 3:
                removeNegatives(L2);
                cout << "L2 после удаления отрицательного элемента:";
                printList(L2);
                break;
            case 4:
                moveMaxToFront(L1);
                cout << "L1 после перемещения самого большого элемента:";
                printList(L1);
                break;
            case 5:
                cout << "Выход из программы\n";
                break;
```

```
            default:
                cout << "Неверный выбор. Пожалуйста, попробуйте еще раз.\n";
        }
    } while (choice != 5);
}

int main() {
    Node* L = nullptr;
    inputList(L);  // Пользователь вводит список L
    menu(L);       // Отображает меню после входа в список L.
    return 0;
}
```

4. Представить результат тестирования программы: скриншоты выполнения каждой операции.



```
Вве◆ите выбор: 3
L2 после удаления ◆трицательного элемента:0 -> nullptr

========= MENU =========
1. Показать список L
2. Разделить список L на L1 и L2
3. Удалить отрицательные элементы из L2
4. Переместите самый бо◆ьшой элемент в начало L1.
5. Выход
========================
Введ◆те выбор: 4
L1 после перемещения самого большого элемента:10 -> 9 -> 6 -> 8 -> 5 -> 3 -> nullptr

========= MENU =========
1. Показать список L
2. Разделить список L на L1 и L2
3. Удалить отрицательные элементы из L2
4. Переместите самый большой элемент в начало L1.
5. Выход
========================
Введите выбор: 5
Выход из программы


...Program finished with exit code 0
Press ENTER to exit console.
```

5. Выводы

Благодаря процессу создания таких функций, как добавление, удаление, разделение, фильтрация и перемещение узлов, это упражнение помогло прояснить структуру данных связанного списка, а также усложнить операции управления и обработки данных.

6. Список информационных источников, которые были использованы при выполнении задания.

TopDev, & TopDev. (2023, August 11). *Односвязный список в C++*. TopDev. https://topdev.vn/blog/danh-sach-lien-ket-don-trong-c/

Blog T. (1970, January 1). [DSLK đơn]. Удалить средний узел в DSLK. *28Tech Blog*. https://blog.28tech.com.vn/dslk-don-xoa-node-giua-trong-dslk

Kim K. (2023, October 29). *Односвязный список — все подробности.* TEKY - Академия креативных технологий. https://teky.edu.vn/blog/danh-sach-lien-ket-don/