Федеральное государственное бюджетное
образовательное учреждение высшего образования
"Московский авиационный институт
(Национальный исследовательский университет)"

Институт №8 "Информационные технологии и прикладная математика"

Кафедра №806 "Вычислительная математика и программирование"

КУРСОВАЯ РАБОТА

По курсу "Практикум программирования"

Задание №8.

"Линейные списки"

Выполнил: студент группы М8О-104Б-22
Тесля Данила Сергеевич
Руководитель:
Потенко Максим Алексеевич
Дата: 04.10.2023
Оценка:.....

Москва - 2023 г.

Содержание

1 Постановка задачи

Составить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры.

Вариант № [1, 4, 7] Тип элемента списка: целый, Тип списка: кольцевой двунаправленный, Нестандартное действие: удалить из списка элементы со значениями в заданном диапазоне

2 Ход решения

2.1 Алгоритм

Перед выполнением работы необходимо изучить материалы по созданию и обработке абстрактного типа данных – линейных списков на языке Си, изучить методы обработки списков и типы списков, а также изучить основы модульного программирование на Си и автоматизацию сборки программ модульной структуры на Си с использованием утилиты make. Создадим программу из трех файлов, один из которых будет главным файлом программы, в нем будет функция main, во втором файле будет реализация всех необходимых функций программы, и третий – заголовочный файл с определениями структур и функций. Так же дополнительно создадим файл в котором будут команды – указания для корректной сборки и компиляции программы, запускаться данный файл будет с помощью утилиты make. В заголовочном файле (.h) определим тип данных - структуру узла списка и напишем прототипы – определения всех функций программы. Далее в файле для реализации функций необходимо будет реализовать функции добавление нового узла в конец списка, добавления узла в начало списка, добавления узла по указанному индексу, извлечения узла из начала списка, извлечения узла из конца списка, удаление узла с указанным индексом, удаление узлов со значениями в заданном диапазоне, функцию получения размера списка, печати списка, удаления списка. Когда все эти функции будут реализованы – перейдем главному файлу программы – в функции main сделаем контекстное меню, где пользователь сможет произвести действия со списком по выбору – добавление нового узла в конец списка, добавления узла в начало списка, добавления узла по указанному индексу, извлечения узла из начала списка, извлечения узла из конца списка, удаление узла с указанным индексом, удаление узлов со значениями в заданном диапазоне, печать списка, сделать это можно например конструкцией switch case в цикле while, и считывать выбор пользователя, пока он не захочет завершить работу со списком.

2.2 Решение

Создадим текстовый файл Makefile, содержащий набор инструкций о том, как компилировать и линковать исходный код программы. Создадим программу из трех файлов, один из которых будет главным файлом программы – kp8main.c, в нем будет функция main, во втором файле kp8source.c будет реализация всех необходимых функций программы, и третий – заголовочный файл kp8header.h с определениями структур и функций. В заголовочном файле для начала воспользуемся директивами #ifndef –LIST-H #define –LIST-H #endif методом условной компиляции, для предотвращения повторной компиляции кода, подключим стандартные библиотеки языка Си. Далее в этом же файле введем определение структуры узла списка typedef struct -node Node; и так по условию варианта список, который необходимо задать – циклический двусвязный список, то полями этой структуры будут int value; - поле значения узла, struct -node* prev-node; - указатель на предыдущий узел struct -node* next-node; - указатель на следующий узел; .Также в заголовочном файле определим конструкцию enum для ее дальнейшего использования в операторе switch в основном файле программы. Далее в файле для реализации функций необходимо будет реализовать функции: Создание (инициализация) узла - create-node – функция принимает значение введенное пользователем, возвращает узел, имеющий это значение, в теле функции выделяется память под структуру типа Node, полю value присваивается переданное значение, а полям указателей на предыдущий и следующий узлы присваиваются пустые указатели. Добавление нового узла в конец списка - list-push-back – функция принимает указатель на список (на голову списка) , и значение узла, с которым необходимо добавить узел в список, в теле функции создается новый узел newnode, затем, если список пуст, то этот узел добавляется как единственный элемент списка – и он становится головой списка, если не пуст, то используется временный указатель last-node – хранящий указатель на последний узел списка, и затем происходит подстановка на его место нового узла – теперь он указывает на последний узел в списке. Добавления узла в начало списка – list-push-front - аналогично, но в конце новый узел заменяет предыдущий указатель на голову списка и сам становится головой. Добавления узла по указанному индексу - list-push-index - принимает те же аргументы, что и предыдущие функции и плюс к ним – индекс в списке, на который нужно вставить элемент, в теле функции вводится цикл for – и происходит проход по узлам списка до введенного индекса, новый узел добавляется аналогично с первыми двумя функциями добавления. Извлечение узла из начала списка – list-pop-front , функция работает аналогично с функцией добавления, но возвращает указатель на удаленный узел, в теле функции, в отличие от функции добавления, со-

здается временный указатель, которому присваивается указаетль на последний узел в списке и функция возвращает этот указатель, и затем в функции main происходит удаление этого временного указателя. Извлечения узла из конца списка - list-pop-back – аналогично, но возвращается указатель на голову списка, а изначальный указатель на голову списка смещается к следующему элементу, если он существует. Извлечение узла с указанным индексом list-pop-index – аналогично с функцией добавления узла по индексу, но возвращается указатель на удаленный узел, указатели переставляются таким образом, чтобы заполнить пустое место в списке. удаление узлов со значениями в заданном диапазоне funcdeleting – функция принимает указатель на голову списка и заданный интервал значения от и до, аналогично происходит прохождение по списку и поиск элементов удовлетворяющих условию, и их удаление по временному указателю, который указывает на каждый элемент списка, который удовлетворяет условию и подлежит удалению. Функцию получения размера списка getsize – происходит проход по списку и подсчет его элементов и возвращается их количество. Печати списка - print-list – происходит проход по списку и вывод значений каждого отдельного узла. Удаления списка free-list – создается временный указатель, осуществляется проход по списку и временный указатель в каждой итерации цикла указывает на следующий элемент списка, происходит удаление каждого из узлов, путем уничтожения временного указателя на каждый из них. Перейдем к главному файлу программы – в функции main сделаем контекстное меню, где пользователь сможет произвести действия со списком по выбору – добавление нового узла в конец списка, добавления узла в начало списка, добавления узла по указанному индексу, извлечения узла из начала списка, извлечения узла из конца списка, удаление узла с указанным индексом, удаление узлов со значениями в заданном диапазоне, печать списка, сделать это можно например конструкцией switch case в цикле while, и считывать выбор пользователя, пока он не захочет завершить работу со списком.

3 Код программы и тесты

https://github.com/tesla-2002/KP8

```
gcc -std=c99 -Wall -Werror kp8main.c kp8source.c -o kp8.out
tesla-2002@tesla2002-HP-ProBook-430-G7:~/labs/kp8$ valgrind --leak-check=full -s  ./kp8.out
==5134== Memcheck, a memory error detector
==5134== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5134== Using Valgrind-3.21.0 and LibVEX; rerun with -h for copyright info
==5134== Command: ./kp8.out
==5134==

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
1
Введите значение узла:
1

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
1
Введите значение узла:
0

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
2
Введите значение узла:
4
```

```
Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
2
Введите значение узла:
3

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
1
Введите значение узла:
-1

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
3
Введите значение узла:
0
Введите индекс добавляемого элемента:
4

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
8
```

```
Список:

... (-1), (0), (1), (4), (0), (3) ...

Size = 6

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
4
Удален узел: -1

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
5
Удален узел: 3

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
6
Введите индекс удаляемого элемента:
60
Элемента с таким индексом не существует!
```

```
Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
6
Введите индекс удаляемого элемента:
0
Удален узел: 0

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
8
Список:

... (1), (4), (0) ...

Size = 3

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
1
Введите значение узла:
-1
```

```
Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
2
Введите значение узла:
1

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
7
Введите диапазон:
От: -1
До: 1
```

```
Удален узел: 0
Удален узел: 1

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
8
Список:

... (4) ...
Size = 1

Что Вы хотите сделать?
 [1] - Добавить узел в начало
 [2] - Добавить узел в конец
 [3] - Добавить узел по индексу
 [4] - Удалить узел в начале
 [5] - Удалить узел в конце
 [6] - Удалить узел по индексу
 [7] - Удалить узлы в диапазоне значений
 [8] - Напечатать список
 [0] - Завершить
0
==5134==
==5134== HEAP SUMMARY:
==5134==     in use at exit: 0 bytes in 0 blocks
==5134==   total heap usage: 10 allocs, 10 frees, 2,240 bytes allocated
==5134==
==5134== All heap blocks were freed -- no leaks are possible
==5134==
```

4 Описание программы

4.1 Описание функций

create-node	создание узла списка
list-push-back	добавление узла в конец списка
list-push-front	добавление узла в начало списка
list-push-index	добавление узла по заданному индексу
list-pop-front	извлечение узла из начала списка
list-pop-back	извлечение узла из конца списка
list-pop-index	извлечение узла по заданному индексу
funcdeleting	удаление узлов со значениями в заданном диапазоне
getsize	получение размера списка
print-list	печать списка
free-list	удаление списка
main	главная функция программы

4.2 Описание ПО

ПК	HP ProBook 430 G7
ОС	Ubuntu 22.04.3 LTS
Интерпретатор команд	Bash
Редактор текстов	Sublime
Язык программирования	C
Местонахождение файла	/home/labs

5 Заключение

В ходе выполнения данной лабораторной работы я изучил и освоил на практике такой абстрактный тип данных на языке Си , как линейные списки, а более подробно изучил один из типов - кольцевой двунаправленный список, изучил и применил в своей программе основные методы обработки двусвязного списка – задание списка, добавление нового узла в конец списка, добавления узла в начало списка, добавления узла по указанному индексу, извлечения узла из начала списка, извлечения узла из конца списка, удаление узла с указанным индексом, удаление узлов со значениями в заданном диапазоне, обход списка, поиск определенных элементов и другие. Кроме того в данной лабораторной работе я научился создавать программы модульной структуры на Си, и автоматизировать сборку таких программ используя утилиту make и файл с инструкциями для этой утилиты. Все полученные мной знания и навыки пригодятся в выполнении лабораторных и курсовых работ в будущем.