

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №8**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Структура хранения данных: линейный динамический список

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-06-22 |  | Кликушин В.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

ОГЛАВЛЕНИЕ

[1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА 3](#_Toc136268895)

[2 ОПРЕДЕЛЕНИЕ АТД 4](#_Toc136268896)

[3 РАЗРАБОТКА ПРОГРАММЫ 6](#_Toc136268897)

[3.1 СТРУКТУРА УЗЛА 6](#_Toc136268898)

[3.2 СТРУКТУРА СПИСКА 6](#_Toc136268899)

[3.3 ВКЛЮЧЕНИЕ ПЕРВОГО УЗЛА В ПУСТОЙ СПИСОК 6](#_Toc136268900)

[3.4 КОД ПРОГРАММЫ 7](#_Toc136268901)

[3.5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 15](#_Toc136268902)

[4 ВЫВОДЫ 20](#_Toc136268903)

[5 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ 21](#_Toc136268904)

# 1 УСЛОВИЕ ЗАДАЧИ И ЗАДАНИЕ ВАРИАНТА

Разработать и реализовать программу управлению линейным однонаправленным списком в соответствии с постановкой задачи, определенной вариантом задания.

Вариант задания включает:

− тип информационной части узла списка - char;

− постановку задачи.

Постановка задачи определяет исходные списки (или список) и задачи, которые требуется выполнить над списками (списком). В постановке задачи каждого варианта представлены три задачи, которые определяют операции над списком.

Условие задачи: Дан линейный однонаправленный список L, содержащий текст. В каждом узле один символ. Слово образуют подряд идущие символы. Слова разделены одним пробелом. 1. Переместить последнее слово в начало списка. 2. Удалить второе слово. 3. Заменить k-ое слово на новое слово. Длина нового слова может быть больше длины k-ого слова.

# 2 ОПРЕДЕЛЕНИЕ АТД

АТД node

{

Данные

data – информационная часть узла типа typeitem

next – указатель на следующий элемент списка (по умолчанию nullptr)

Операции

1. Заполнение списка с добавлением нового узла в начало списка

Предусловие. p\_head – указатель на элемент типа node.

Постусловие. Создает список путем добавления нового узла в начало. Нет возвращаемого значения.

**input\_list(node\* p\_head);**

1. Заполнение списка с добавлением нового узла в конец списка

Предусловие. p\_head – ссылка на указатель на элемент типа node.

Постусловие. Создает список путем добавления нового узла в конец. Нет возвращаемого значения.

**create\_list(node\*& p\_head);**

1. Вывод списка в консоль (для заполнения добавлением узлов в конец)

Предусловие. p\_head – указатель на элемент типа node.

Постусловие. Выводит в консоль список. Возвращает 1, если список пустой, и 0, если список заполнен.

**output\_list(node\* p\_head);**

1. Вывод списка в консоль (для заполнения добавлением узлов в начало)

Предусловие. p\_head – указатель на элемент типа node.

Постусловие. Выводит в консоль список. Возвращает 1, если список пустой, и 0, если список заполнен.

**rec\_output(node\* p\_head);**

1. Вставка нового узла в начало списка

Предусловие. p\_head – указатель на элемент типа node, x – указатель на элемент типа node.

Постусловие. Добавляет новый узел в начало списка. Возвращает 1, если список пустой, иначе 0.

**insert\_begin(node\* p\_head, node\* x);**

1. Вспомогательная функция вставки для перемещения последнего слова в начало

Предусловие. p\_head – ссылка на указатель на элемент типа node, position – позиция последнего пробела в предложении.

Постусловие. Изменяет исходный список путем перемещения последнего элемента в начало.

**insert(node\*& p\_head, int position);**

1. Основная функция переноса последнего слова в начало

Предусловие. p\_head – ссылка на указатель на элемент типа node.

Постусловие. Находит позицию последнего пробела и вызывает вспомогательную функцию. Возвращает 1, если список пустой или в нем не найдено пробелов, иначе 0.

**move\_last(node\*& p\_head);**

1. Поиск количества пробелов в строке

Предусловие. p\_head – указатель на элемент типа node.

Постусловие. Возвращает количество пробелов в строке.

**find\_space(node\* p\_head);**

1. Удаление второго слова

Предусловие. p\_head – указатель на элемент типа node.

Постусловие. Меняет исходный список путем удаления второго слова. Возвращает 1, если список пустой или в нем одно слово и 0 в противном случае.

**delete\_second\_word(node\* p\_head);**

1. Замена k-ого слова на новое

Предусловие. p\_head – ссылка на указатель на элемент типа node, k – позиция заменяемого слова.

Постусловие. Возвращает 1, если список пустой или позиция заменяемого слова задана неверно и 0 в противном случае.

**replace\_word(node\*& p\_head, int k);**

# 3 РАЗРАБОТКА ПРОГРАММЫ

## 3.1 СТРУКТУРА УЗЛА

typedef char typeitem;

struct node

{

typeitem data;

node\* next = nullptr;

};

## 3.2 СТРУКТУРА СПИСКА

Изображение выглядит как Шрифт, линия, белый, текст

Автоматически созданное описание

Рисунок 1 – Структура списка из 5 элементов

## 3.3 ВКЛЮЧЕНИЕ ПЕРВОГО УЗЛА В ПУСТОЙ СПИСОК

Создается пустой указатель, который и будет первым узлом. Дальше создается новый элемент типа node, в него записывается символ, p\_head присваивается q.

Изображение выглядит как рукописный текст, текст, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 2 –Включение первого узла в пустой список

## 3.4 КОД ПРОГРАММЫ

Код файла node.h:

#pragma once

#include <iostream>

#include <string>

using namespace std;

typedef char typeitem;

struct node

{

typeitem data;

node\* next = nullptr;

};

void input\_list(node\* p\_head);

void create\_list(node\*& p\_head);

int output\_list(node\* p\_head);

void rec\_output(node\* p\_head);

int insert\_begin(node\* p\_head, node\* x);

void insert(node\*& p\_head, int position);

int move\_last(node\*& p\_head);

int find\_space(node\* p\_head);

int delete\_second\_word(node\* p\_head);

int replace\_word(node\*& p\_head, int k);

Код файла node.cpp:

#include "node.h"

void input\_list(node\* p\_head)

{

node\* q;

string str;

cout << " Введите строку, каждый символ которой будет сохранен в отдельном узле: ";

cin.ignore();

getline(cin, str);

for (int i = 0; i < str.size(); i++)

{

q = new node();

q -> data = str[i];

insert\_begin(p\_head,q);

}

}

void create\_list(node\*& p\_head)

{

node\* q, \* q1 = nullptr;

string str;

cout << " Введите строку, каждый символ которой будет сохранен в отдельном узле: ";

cin.ignore();

getline(cin, str);

for (int i = 0; i < str.size(); i++)

{

q = new node();

q->data = str[i];

q->next = nullptr;

if (p\_head == nullptr)

{

p\_head = q;

q1 = p\_head;

}

else

{

q1->next = q;

q1 = q;

}

}

}

int output\_list(node\* p\_head)

{

if (p\_head == nullptr)

return 1;

node\* q = p\_head;

while (q != nullptr)

{

cout << q->data;

q = q->next;

}

cout << endl;

return 0;

}

void rec\_output(node\* p\_head)

{

node\* q = p\_head->next;

if (p\_head->next != nullptr)

{

rec\_output(p\_head->next);

}

if (p\_head->next != nullptr)

{

cout << p\_head->data;

}

else

{

cout << p\_head->data;

}

}

int insert\_begin(node\* p\_head, node\* x)

{

if (p\_head == nullptr)

{

return 1;

}

x->next = p\_head->next;

p\_head->next = x;

return 0;

}

void insert(node\*& p\_head, int position)

{

node\* q = p\_head;

node\* pos\_node = nullptr;

node\* prev\_pos = nullptr;

int i = 1;

while (q->next != nullptr)

{

if ((i + 1) == position)

{

pos\_node = q->next;

prev\_pos = q;

}

q = q->next;

i++;

}

if (pos\_node != nullptr)

{

prev\_pos->next = nullptr;

q->next = pos\_node;

node\* temp = pos\_node->next;

pos\_node->next = p\_head;

p\_head = temp;

}

}

int move\_last(node\*& p\_head)

{

if (p\_head == nullptr or find\_space(p\_head) == 0)

{

return 1;

}

node\* q = p\_head;

int last\_space = 1;

int i = 1;

while (q->next != nullptr)

{

if (q->data == ' ')

{

last\_space = i;

}

q = q->next;

i++;

}

insert(p\_head, last\_space);

return 0;

}

int find\_space(node\* p\_head)

{

if (p\_head == nullptr)

return 0;

node\* q = p\_head->next;

int count = 0;

int i = 0;

while (q != nullptr)

{

if (q->data == ' ')

{

count++;

}

q = q->next;

i++;

}

return count;

}

int delete\_second\_word(node\* p\_head)

{

if (find\_space(p\_head) != 0)

{

if (find\_space(p\_head) == 1)

{

while (p\_head->next->data != ' ')

{

p\_head = p\_head->next;

}

p\_head->next = nullptr;

return 0;

}

else

{

while (p\_head->next != nullptr)

{

if (p\_head->data == ' ')

{

node\* space = p\_head;

node\* tmp1 = p\_head->next;

if (find\_space(p\_head) != 0)

{

while (tmp1->data != ' ')

{

node\* tmp2 = tmp1;

tmp1 = tmp1->next;

delete tmp2;

}

space->next = tmp1->next;

delete tmp1;

return 0;

}

else

{

p\_head->next = nullptr;

return 0;

}

}

p\_head = p\_head->next;

}

}

}

return 1;

}

int replace\_word(node\*& p\_head, int k)

{

if (k <= 1 or k > (find\_space(p\_head) + 1))

{

return 1;

}

node\* q1 = p\_head;

int i = 1;

while (q1->next != nullptr)

{

if (q1->data == ' ')

{

i++;

if (i == k)

{

node\* first\_space = q1;

q1 = q1->next;

while (q1 != nullptr)

{

if (q1->data != ' ')

{

node\* tmp = q1;

q1 = q1->next;

delete tmp;

}

else

{

break;

}

}

string data;

cout << " Введите новое слово: ";

cin >> data;

node\* first = nullptr;

node\* last = nullptr;

for (int i = 0; i < data.size(); i++)

{

node\* q = new node;

q->data = data[i];

if (first == nullptr)

{

first = q;

last = first;

}

else

{

last->next = q;

last = q;

}

}

first\_space->next = first;

last->next = q1;

return 0;

}

}

q1 = q1->next;

}

return 0;

}

Код файла main.cpp:

#include <iostream>

#include "node.h"

using namespace std;

int main()

{

system("chcp 1251");

node\* q = nullptr;

node\* qq = new node;

int number;

while (true)

{

cout << "-----------------------------------------------------------------------------------------------------------" << endl;

cout << " Меню\n";

cout << " 1. Заполнить список с вставкой новых узлов в начало списка\n";

cout << " 2. Заполнить список вставкой новых узлов в конец списка\n";

cout << " 3. Вывести список\n";

cout << " 4. Переместить последнее слово в начало списка\n";

cout << " 5. Удалить второе слово\n";

cout << " 6. Заменить k - ое слово на новое слово\n";

cout << " 7. Завершить работу\n";

cin >> number;

switch (number)

{

case 1:

qq = new node;

input\_list(qq);

rec\_output(qq->next);

cout << endl;

break;

case 2:

q = nullptr;

create\_list(q);

break;

case 3:

if (output\_list(q))

{

cout << " Пустой список\n";

}

break;

case 4:

if (move\_last(q))

{

cout << " Список пустой или содержит одно слово\n";

}

else

{

output\_list(q);

}

break;

case 5:

if (delete\_second\_word(q))

{

cout << " Ошибка - список содержит одно слово или он пустой\n";

}

else

{

output\_list(q);

}

break;

case 6:

cout << " Введите позицию слова, которое необходимо заменить на новое: ";

cin >> number;

if (replace\_word(q, number))

{

cout << " Ошибка - слово с таким порядковым номером не найдено\n";

}

else

{

output\_list(q);

}

break;

case 7:

return 0;

default:

cout << " Нет такого пункта\n";

}

}

## 3.5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Результат тестирования функции перемещения последнего слова в начало

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4 – Результат тестирования функции перемещения последнего слова в начало

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5 – Результат тестирования функции перемещения последнего слова в начало

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6 – Результат тестирования функции удаления второго слова

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 – Результат тестирования функции удаления второго слова

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8 - Результат тестирования функции удаления второго слова

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9 – Результат тестирования функции замены k-ого слова

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 - Результат тестирования функции замены k-ого слова

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 11 - Результат тестирования функции замены k-ого слова

# 4 ВЫВОДЫ

В ходе выполнения практической работы были получены навыки работы с однонаправленным линейным списком. Были реализованы операции вставки узла в начало списка, создания списка и соответствующие операции варианта.

# 5 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Учебно-методическое пособие СиАОД (часть 1)

2. Приложение к практическим работам – СДО (online-edu.mirea.ru)