

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**

Отчет по выполнению практического задания № 5

**Тема**: **Структуры хранения линейных структур данных. Однонаправленный динамический список.**

Дисциплина: Структуры и алгоритмы обработки данных

|  |  |
| --- | --- |
| Выполнил студент | Дмитриев П. В. |
|  | Фамилия И.О. |
| Группа | ИКБО-33-21 |
|  | Номер группы |

Москва 2022

Вариант № 7

ЗАДАНИЕ 1

Реализуйте программу решения задачи варианта по использованию линейного однонаправленного списка. Информационная часть узла – int.

Задача варианта

Дан линейный однонаправленный список L

1) Разработать функцию, которая проверяет, есть ли в списке L два одинаковых элемента.

2) Разработать функцию, которая удаляет из списка L максимальное значение.

3) Разработать функцию, которая вставляет в список L новое значение перед каждым узлом в четной позиции.

1. **Разработка программы задачи 1**
   1. **Постановка задачи**

Дано. Дан линейный однонаправленный список L

Результат. Вывод отредактированного линейного однонаправленного списка

**1.2 Описание модели решения**

Программа запрашивает количество элементов в списке. Вызывается функция заполнения структуры (creat\_list(L, n)). В цикле for выделяется динамическая память под структуру, после в поле data вводятся данные с клавиатуры. Далее функция типа bool same\_elem(L) возвращает булевое значение, зависящее от наличия двух одинаковых элементов в структуре. Задача реализуется с помощью двух вложенных циклов while, в котором происходит последовательный перебор двух соседних элементов структуры. Если нашлись такие два элемента, то функция возвращает true, если таких элементов не нашлось, ничего не возвращается. Результат возврата функции передается в переменную b, которая проверяется условным оператором на истину.

**1.3 Декомпозиция:**

Определение прототипов функций:

1. Инициализация структуры данных

struct node {

int data;

node\* next;

};

1. Заполнение числами с клавиатуры структуры

Предусловие. \*&L – указатель на ссылку на структуру, n – количество элементов в структуре, определяется пользователем

Постусловие. Заполненная структура с данными

void creat\_list(node\*& L, int n)

1. Вывод структуры на экран

Предусловие. \*L – указатель на вершину списка

Постусловие. Вывод списка

void print\_list(node\* L)

1. Проверка списка на два одинаковых элемента

Предусловие. \*L – указатель на вершину списка

Постусловие. Вывод строки на экран

void print\_list(node\* L)

**1.4 Код программы:**

#include <iostream>

using namespace std;

struct node {

int data;

node\* next;

};

void creat\_list(node\*& L, int n) {

node\* q, \*q1;

for (int i = 1; i <= n; i++) {

q = new node;

q->next = NULL;

cin >> q->data;

if (L == NULL) {

L = q;

}

else {

q1 = L;

while (q1->next != NULL) {

q1 = q1->next;

}

q1->next = q;

}

}

}

void print\_list(node\* L) {

cout << "Вывод списка: ";

node\* q = L;

while (q != NULL) {

cout << q->data << " ";

q = q->next;

}

cout << "\n";

}

bool same\_elem(node\* L) {

node\* q = L;

while (q->next != NULL) {

while (q->next != NULL) {

q = q->next;

if (L->data == q->data) {

return true;

}

}

}

}

int main() {

setlocale(LC\_ALL, "ru");

node\* L = NULL;

int n;

bool b = false;

cout << "Введите количество чисел: ";

cin >> n;

creat\_list(L, n);

b = same\_elem(L);

if (b) {

cout << "В списке есть два одинаковых элемента";

}

else {

cout << "В списке нет одинаковых элементов";

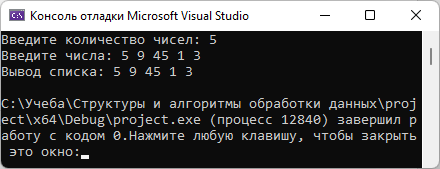
}

cout << "\n";

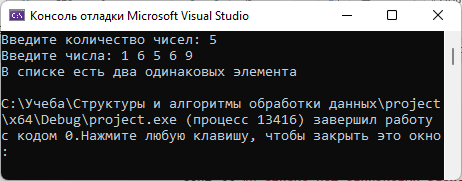
print\_list(L);}

* 1. **Скриншоты:**

1. Тестирование функции декомпозиции: вывод элементов списка на экран



1. Тестирование функции декомпозиции: проверка списка на два одинаковых элемента



1. **Разработка программы задачи 1**
   1. **Постановка задачи**

Дано. Дан линейный однонаправленный список L

Результат. Проверка списка на наличие двух одинаковых элементов

**2.2 Описание модели решения**

В данной задаче используются те же функции ввода и вывода данных списка, что и в предыдущей. Добавляются две функции: нахождение максимального числа и его удаление. Внутри цикла for сравниваются два соседних числа и если второе больше первого, то переменной L присваивается указатель на это число. Функция(find\_max(L)) возвращает указатель на максимальное число. Функция (del\_max(L, max\_n)) получает указатель на максимальное число в списке и удаляет его.

**2.3 Декомпозиция:**

Определение прототипов функций:

1. Нахождение максимального числа в списке

Предусловие. \*L – указатель на структуру

Постусловие. Указатель на максимальное число

struct node\* find\_max(node\* L)

1. Удаление максимального числа из списка

Предусловие. \*L – указатель на структуру, \*del – указатель на максимальное число

Постусловие. Список без максимального числа

struct node\* del\_max(node\* L, node\* del)

**2.4 Код программы:**

#include <iostream>

using namespace std;

struct node {

int data;

node\* next;

};

void creat\_list(node\*& L, int n) {

cout << "Введите числа: ";

node\* q, \*q1;

for (int i = 1; i <= n; i++) {

q = new node;

q->next = NULL;

cin >> q->data;

if (L == NULL) {

L = q;

}

else {

q1 = L;

while (q1->next != NULL) {

q1 = q1->next;

}

q1->next = q;

}

}

}

void print\_list(node\* L) {

cout << "Вывод списка: ";

node\* q = L;

while (q != NULL) {

cout << q->data << " ";

q = q->next;

}

cout << "\n";

}

bool same\_elem(node\* L) {

node\* q = L;

while (q->next != NULL) {

while (q->next != NULL) {

q = q->next;

if (L->data == q->data) {

return true;

}

}

}

}

struct node\* find\_max(node\* L) {

node\* q = L;

for (; q != NULL; q = q->next) {

if (q->data > L->data) {

L = q;

}

}

return L;

}

struct node \* del\_max(node\* L, node\* del) {

node\* q = L;

while (q->next != del) {

q = q->next;

}

q->next = del->next;

free(del);

return(q);

}

int main() {

setlocale(LC\_ALL, "ru");

node\* L = NULL;

int n;

bool b = false;

cout << "Введите количество чисел: ";

cin >> n;

creat\_list(L, n);

b = same\_elem(L);

if (b) {

cout << "В списке есть два одинаковых элемента";

}

else {

cout << "В списке нет одинаковых элементов";

}

cout << "\n";

node\* max\_n = find\_max(L);

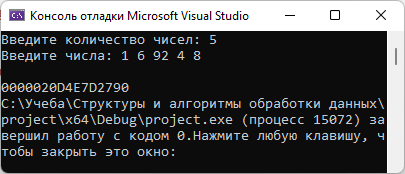
del\_max(L, max\_n);

print\_list(L);

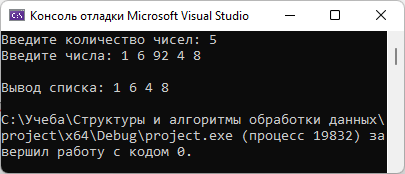
}

* 1. **Скриншоты:**

1. Тестирование функции декомпозиции: нахождение максимального числа в списке



1. Тестирование функции декомпозиции: удаление максимального числа из списка



Ответы на вопросы:

1) Различают три уровня для описания данных: концептуальный (с позиции администратора); реализации (с позиции прикладного программиста и пользователя); физический (с позиции системного программиста). На концептуальном уровне описывают объекты, атрибуты и значения данных. На уровне реализации имеют дело с записями, элементами данных и связями между записями.

2) Тип данных определяет формат представления данных в памяти компьютера; множество допустимых значений, которые может принимать переменная или константа данного типа; множество допустимых операций, применимых к этому типу. Все данные, используемые программой, должны принадлежать либо к стандартному (т.е. какому-либо заранее известному), либо к пользовательскому (т.е. определяемому программистом) типу данных.

3) Структура данных — это контейнер, который хранит данные в определенном макете. Этот «макет» позволяет структуре данных быть эффективной в некоторых операциях и неэффективной в других.

4) Структуры хранения данных могут быть статическими или динамическими. Статические структуры создаются при компиляции программы в области данных и их размер изменить нельзя во время работы программы. Они освобождают память, выделенную им, только по завершении программы. Динамические структуры создаются во время работы программы в динамической памяти, их размер может быть изменен во время работы программы. Память этой структуры хранения может и ее отдельных элементов быть освобождена во время работы программы.

5) Линейная структура данных - это тип структуры хранения данных, который хранит данные в непрерывном расположении, например массивы, которые хранят данные одного и того же типа в последовательном расположении в памяти.

6) Линейный список — это динамическая структура данных, каждый элемент которой посредством указателя связывается со следующим элементом. Из определения следует, что каждый элемент списка содержит поле данных (Data) (оно может иметь сложную структуру) и поле ссылки на следующий элемент (next).

7) Стек – это структура данных, в которой новый элемент всегда записывается в её начало (вершину) и очередной читаемый элемент также всегда выбирается из её начала.

8) Очередь – структура данных типа «список», позволяющая добавлять элементы лишь в конец списка, и извлекать их из его начала. Она функционирует по принципу FIFO (First In, First Out — «первым пришёл — первым вышел»), для которого характерно, что все элементы a1, a2, …, an-1, an, добавленные раньше элемента an+1, должны быть удалены прежде, чем будет удален элемент an+1.

9) Список - это линейная цепочка элементов с последовательным доступом и вставкой / удалением элементов в любом месте списка.

Стек - это доступ по схеме LIFO: "последним пришёл, первым ушёл". Самый очевидный пример - стопка листов: сверху кладём и сверху же забираем.

10) Односвязный.

11) a)O(n) б) O(n)

12) a) O(n) б) O(n)

13) Трюк Вирта заключается в том, чтобы указатель на удаляемый элемент перенаправить на элемент, следующий за удаляемым.

14) Односвязный список состоит из узлов. Узел содержит значение и указатель на следующий узел.

15) node\* q = L;

while (q != NULL) {

cout << q->data << " ";

q = q->next;

}

16) void lasttofirst(node\*& L) {

node\* last = nullptr;

node\* first = L;

while (first->next) {

last = first;

first = first->next;

}

last->next = 0;

first->next = L;

L = first;

}