Федеральное государственное автономное образовательное учреждение

высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

КАФЕДРА СИСТЕМНОГО АНАЛИЗА И ТЕЛЕКОММУНИКАЦИИ

**ПРОВЕРИЛ ВЫПОЛНИЛ**

Старший преподаватель кафедры САиТ Студент группы КТбо1-2

Лапшин В.С. Кукса К.С.

«\_\_» \_\_\_\_\_\_\_\_\_\_2023 г. «\_\_» \_\_\_\_\_\_\_\_\_\_2023 г.

**ЛАБОРАТОРНАЯ РАБОТА №3**

По дисциплине

«Основы алгоритмизации и программирования»

Вариант 4

На тему

«Динамическое распределение памяти, списки»

**Таганрог 2023**

**Цель**: освоение способов представления и обработки данных в виде списков.

**Задачи**: написать программы для работы с односвязным и двусвязным спискам в соответствии с выданным вариантом задания. Предусмотреть в программах следующие функции:

1. Включить новый элемент в конец списка.
2. Включить новый элемент на заданное пользователем место списка.
3. Включить новый элемент после элемента с заданной информационной частью.
4. Включить новый элемент перед элементом с заданной информационной частью.
5. Включить новый элемент в середину списка.
6. Исключить элемент из середины списка.
7. Исключить элемент с заданной информационной частью.
8. Исключить элемент из конца списка.
9. Исключить элемент из заданного пользователем места списка.
10. Исключить элемент из головы списка.

Вариант задания: учет материальных ценностей бухгалтерии.

**Ход работы**:

**Составление математической модели**:

Связный список – динамическая структура данных, состоящая из узлов, содержащих указатели на следующий и/или предыдущий узел списка.

**Проектирование.**

**Реализация.**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

const int N = 50;

struct invent1

{

char invent\_name[N];

int invent\_count;

invent1\* next;

};

struct invent2

{

char invent\_name[N];

int invent\_count;

invent2\* next;

invent2\* prev;

};

struct input

{

char in\_name[N];

int in\_count;

} inp;

typedef invent1\* Pinv1;

typedef invent2\* Pinv2;

Pinv1 Head1 = NULL;

Pinv2 Head2 = NULL;

Pinv2 Tail2 = NULL;

void nameInput(int op)

{

printf("Введите название предмета: ");

if (op) {

fgets(inp.in\_name, N, stdin);

fgets(inp.in\_name, N, stdin);

}

else fgets(inp.in\_name, N, stdin);

strcpy(inp.in\_name, strtok(inp.in\_name, "\n"));

}

void countInput()

{

printf("Введите количество: ");

scanf("%d", &inp.in\_count);

}

Pinv1 CreateNode1(int opt)

{

nameInput(opt);

countInput();

Pinv1 NewNode = (struct invent1\*)malloc(sizeof(struct invent1));

strcpy(NewNode->invent\_name, inp.in\_name);

NewNode->invent\_count = inp.in\_count;

NewNode->next = NULL;

return NewNode;

}

void AddFirst1(Pinv1& Head, Pinv1 NewNode)

{

NewNode->next = Head;

Head1 = NewNode;

}

void AddAfter1(Pinv1 p, Pinv1 NewNode)

{

NewNode->next = p->next;

p->next = NewNode;

}

void AddLast1(Pinv1& Head, Pinv1 NewNode)

{

Pinv1 q = Head;

if (Head == NULL) {

AddFirst1(Head, NewNode);

return;

}

while (q->next) q = q->next;

AddAfter1(q, NewNode);

}

void AddBefore1(Pinv1& Head, Pinv1 p, Pinv1 NewNode)

{

Pinv1 q = Head;

if (Head == p) {

AddFirst1(Head, NewNode);

return;

}

while (q && q->next != p) q = q->next;

if (q) AddAfter1(q, NewNode);

}

Pinv1 Find1(Pinv1 Head, int opt3)

{

Pinv1 q = Head;

switch (opt3) {

case 0: {

nameInput(1);

while (q && strcmp(q->invent\_name, inp.in\_name) != 0)

q = q->next;

break;

}

case 1: {

countInput();

while (q && q->invent\_count != inp.in\_count)

q = q->next;

break;

}

default: { q = NULL; }

}

return q;

}

void DeleteNode1(Pinv1& Head, Pinv1 p)

{

Pinv1 q = Head;

if (Head == p)

Head1 = p->next;

else {

while (q && q->next != p)

q = q->next;

if (q == NULL) return;

q->next = p->next;

}

free(p);

}

Pinv1 NodeRunner1(Pinv1& Head, int pos)

{

Pinv1 inv = Head;

int now = 0;

while (now != pos) {

inv = inv->next;

now++;

}

return inv;

}

void ListPrint1(Pinv1 p)

{

puts("Односвязный список:");

if (p == NULL) puts("Список пуст.");

Pinv1 inv = p;

while (inv != NULL)

{

printf("%s - %d\n", inv->invent\_name, inv->invent\_count);

inv = inv->next;

}

}

Pinv2 CreateNode2()

{

Pinv2 NewNode = (struct invent2\*)malloc(sizeof(struct invent2));

strcpy(NewNode->invent\_name, inp.in\_name);

NewNode->invent\_count = inp.in\_count;

NewNode->next = NULL;

NewNode->prev = NULL;

return NewNode;

}

void AddFirst2(Pinv2& Head, Pinv2 NewNode)

{

NewNode->next = Head;

NewNode->prev = NULL;

Head2 = NewNode;

}

void AddAfter2(Pinv2 p, Pinv2 NewNode)

{

NewNode->prev = p;

NewNode->next = p->next;

p->next = NewNode;

}

void AddLast2(Pinv2& Head, Pinv2 NewNode)

{

Pinv2 q = Head;

if (Head == NULL) {

AddFirst2(Head, NewNode);

return;

}

while (q->next) q = q->next;

NewNode->prev = q;

NewNode->next = Tail2;

q->next = NewNode;

}

void AddBefore2(Pinv2& Head, Pinv2 p, Pinv2 NewNode)

{

Pinv2 q = Head;

if (Head == p) {

AddFirst2(Head, NewNode);

return;

}

while (q && q->next != p) q = q->next;

if (q) AddAfter2(q, NewNode);

}

Pinv2 Find2(Pinv2 Head, int opt3)

{

Pinv2 q = Head;

switch (opt3) {

case 0: {

while (q && strcmp(q->invent\_name, inp.in\_name) != 0)

q = q->next;

break;

}

case 1: {

while (q && q->invent\_count != inp.in\_count)

q = q->next;

break;

}

default: { q = NULL; }

}

return q;

}

void DeleteNode2(Pinv2& Head, Pinv2 p)

{

Pinv2 q = Head;

if (Head == p)

Head2 = p->next;

else {

while (q && q->next != p)

q = q->next;

if (q == NULL) return;

q->next = p->next;

}

free(p);

}

Pinv2 NodeRunner2(Pinv2& Head, int pos)

{

Pinv2 inv = Head;

int now = 0;

while (now != pos) {

inv = inv->next;

now++;

}

return inv;

}

void ListPrint2(Pinv2 p)

{

puts("Двусвязный список:");

if (p == NULL) puts("Список пуст.");

Pinv2 inv = p;

while (inv != NULL)

{

printf("%s - %d\n", inv->invent\_name, inv->invent\_count);

inv = inv->next;

}

}

int main()

{

setlocale(LC\_ALL, "RU");

int opt = 1, opt3 = -1, pos = -1, total = 0;

char name[N];

Pinv1 p1;

Pinv2 p2;

while (opt) {

puts("1. Включить новый элемент в конец списка.");

puts("2. Включить новый элемент на заданное пользователем место списка.");

puts("3. Включить новый элемент после элемента с заданной информационной частью.");

puts("4. Включить новый элемент перед элементом с заданной информационной частью.");

puts("5. Включить новый элемент в середину списка.");

puts("6. Исключить элемент из середины списка.");

puts("7. Исключить элемент с заданной информационной частью.");

puts("8. Исключить элемент из конца списка.");

puts("9. Исключить элемент из заданного пользователем места списка.");

puts("10. Исключить элемент из головы списка.");

puts("11. Вывести содержимое списков.");

puts("0. Закончить работу");

printf("Выберите действие: ");

scanf("%d", &opt);

puts("--------------------");

switch (opt) {

case 0: { break; }

case 1: {

AddLast1(Head1,CreateNode1(1));

AddLast2(Head2, CreateNode2());

puts("--------------------");

total++;

break;

}

case 2: {

printf("Введите номер позиции: ");

scanf("%d", &pos);

if (pos == 0) {

AddFirst1(Head1, CreateNode1(1));

AddFirst2(Head2, CreateNode2());

puts("--------------------");

total++;

break;

}

if (pos < 0 || pos >= total) {

puts("Неверный номер.");

puts("--------------------");

break;

}

AddAfter1(NodeRunner1(Head1, pos - 1), CreateNode1(1));

AddAfter2(NodeRunner2(Head2, pos - 1), CreateNode2());

puts("--------------------");

total++;

break;

}

case 3: {

printf("0 - название, 1 - количество: ");

scanf("%d", &opt3);

p1 = Find1(Head1, opt3);

p2 = Find2(Head2, opt3);

if (p1 == NULL || p2 == NULL) total--;

else {

AddAfter1(p1, CreateNode1(opt3));

AddAfter2(p2, CreateNode2());

}

puts("--------------------");

total++;

break;

}

case 4: {

printf("0 - название, 1 - количество: ");

scanf("%d", &opt3);

p1 = Find1(Head1, opt3);

p2 = Find2(Head2, opt3);

if (p1 == NULL || p2 == NULL) total--;

else {

AddBefore1(Head1, p1, CreateNode1(opt3));

AddBefore2(Head2, p2, CreateNode2());

}

puts("--------------------");

total++;

break;

}

case 5: {

pos = total / 2;

if (pos == 0) {

AddFirst1(Head1, CreateNode1(1));

AddFirst2(Head2, CreateNode2());

puts("--------------------");

total++;

break;

}

if ((total % 2) == 0) {

AddAfter1(NodeRunner1(Head1, pos - 1), CreateNode1(1));

AddAfter2(NodeRunner2(Head2, pos - 1), CreateNode2());

}

else {

AddBefore1(Head1, NodeRunner1(Head1, pos), CreateNode1(1));

AddBefore2(Head2, NodeRunner2(Head2, pos), CreateNode2());

}

puts("--------------------");

total++;

break;

}

case 6: {

pos = total / 2;

if (pos == 0) {

puts("Список пуст.");

puts("--------------------");

break;

}

if ((total % 2) == 0) {

DeleteNode1(Head1, NodeRunner1(Head1, pos - 1));

DeleteNode2(Head2, NodeRunner2(Head2, pos - 1));

}

else {

DeleteNode1(Head1, NodeRunner1(Head1, pos));

DeleteNode2(Head2, NodeRunner2(Head2, pos));

}

total--;

break;

}

case 7: {

printf("0 - название, 1 - количество: ");

scanf("%d", &opt3);

p1 = Find1(Head1, opt3);

p2 = Find2(Head2, opt3);

if (p1 == NULL || p2 == NULL) {

puts("Элемент не найден");

total++;

}

else {

DeleteNode1(Head1, p1);

DeleteNode2(Head2, p2);

}

puts("--------------------");

total--;

break;

}

case 8: {

if (total == 0) {

puts("Список пуст.");

puts("--------------------");

break;

}

Pinv1 q1 = Head1;

Pinv2 q2 = Head2;

for (int i = 0; i < total - 1; i++) {

q1 = q1->next;

q2 = q2->next;

}

DeleteNode1(Head1, q1);

DeleteNode2(Head2, q2);

total--;

break;

}

case 9: {

printf("Введите номер позиции: ");

scanf("%d", &pos);

if (pos < 0 || pos >= total) {

puts("Неверный номер.");

puts("--------------------");

break;

}

DeleteNode1(Head1, NodeRunner1(Head1, pos));

DeleteNode2(Head2, NodeRunner2(Head2, pos));

puts("--------------------");

total--;

break;

}

case 10: {

Pinv1 q1 = Head1;

Pinv2 q2 = Head2;

DeleteNode1(Head1, q1);

DeleteNode2(Head2, q2);

total--;

break;

}

case 11: {

ListPrint1(Head1);

puts("--------------------");

ListPrint2(Head2);

puts("--------------------");

}

default: break;

}

}

ListPrint1(Head1);

puts("--------------------");

ListPrint2(Head2);

puts("--------------------");

return 0;

}

**Результат работы программы.**

Входные данные:

Команды и данные, переданные через консоль, например:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Выходные данные:

Изображение выглядит как текст

Автоматически созданное описание

**Вывод:**

В ходе работы была написана программа, позволяющая работать с односвязным и двусвязным списками, получены знания о работе со связанными списками при помощи различных методов.