**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 10

**На тему:** *“Динамічні структури даних”*

**З дисципліни:** *“Основи програмування”*

**Лектор:**

ст.викл. каф. ПЗ

Муха Т.О.

**Виконав:**

ст. гр. ПЗ-14

Губик А. С.

**Прийняв:**

ст.викл. каф ПЗ

Кутельмах Р. К.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

**Тема роботи:** динамічні структури даних.

**Мета роботи:** володіти практичними прийомами створення та опрацювання динамічних

списків.

**Індивідуальне завдання**

Виконати завдання з лабораторної роботи No 8, організувавши послідовність структур в

однозв’язний список. Реалізувати операцію вставки нового елемента у відсортований список і

операцію вилучення зі списку даних, які відповідають одній з наступних умов:

про студентів, які не мають оцінки 5;

**Теоретичні відомості**

2.1 Загальне поняття про динамічні структури

Динамічні структури даних характеризуються відсутністю фізичної суміжності елементів

структури в пам'яті (різні елементи «розкидані» в пам'яті), а також непостійністю та

непередбачуваністю розміру (в процесі виконання програми можуть додаватися нові елементи чи видалятися існуючі). Динамічні структури являють собою сукупність об’єктів (як правило,

розміщених у динамічній пам'яті), кожен з яких містить інформацію про адресу в пам'яті іншого

об’єкта, зв’язаного з ним.

Оскільки елементи динамічної структури розміщуються за непередбачуваними адресами

пам'яті, то адресу кожного елемента структури неможливо обчислити за адресою попереднього або наступного елемента. Між елементами динамічної структури встановлюються явні зв’язки за допомогою вказівників. Таке представлення даних у пам'яті називається зв’язним. Елемент

динамічної структури складається з двох полів:

 інформаційного поля (поля даних), яке містить ті дані, для яких власне і створюється

структура;

 поле зв'язків, в якому містяться один або декілька вказівників, що зв’язують даний елемент з іншими елементами структури.

Інформаційне поле може бути представлене не лише сукупністю даних вбудованих типів, але

й масивами, структурами тощо.

Розмір динамічної структури обмежується лише доступним обсягом машинної пам'яті. При

зміні логічної послідовності елементів структури вимагається не переміщення даних у пам'яті, а

лише корекція вказівників.

Однак, робота зі вказівниками вимагає більшої кваліфікації програміста, для зберігання

вказівників витрачається додаткова пам'ять, а доступ до елементів зв’язної структури може бути менш ефективним за часом. Для порівняння: при суміжному розміщенні однорідних даних у пам'яті (яке має місце в масиві) для обчислення адреси будь-якого елемента потрібно знати номер цього елемента та адреси початку області пам'яті (доступ прямий), а для доступу до елемента динамічної структури потрібно послідовно «перебрати» усі елементи динамічної структури від початку.

2.2 Списки

Список – це впорядкована множина, що складається зі змінного числа елементів, до яких

застосовні операції додавання та видалення елементів. Якщо вказівник посилається лише на один інший елемент списку, то такий список називається однонапрямленим.

Наприклад, елемент однонапрямленого списку може описуватися структурою:

struct Item {

int num;

struct Item\* next;

};

Відповідно, список схематично виглядатиме так:

First – це перший елемент списку («голова списку»), а останній елемент списку ні на що не

посилається – його поле next встановлене у NULL.

Елемент списку виділяється в динамічній пам’яті:

struct Item \*p = (struct Item\*)malloc(sizeof(struct Item));

Для додавання нового елемента на початок списку слід створити новий елемент X, в його

поле next записати вказівник на елемент списку, що в поточний момент є першим, і вказівникові,

який позначає перший елемент списку, присвоїти значення X. Наприклад:

struct Item \*p = (struct Item\*)malloc(sizeof(struct Item));

p->next = First;

First = p; /\* First є вказівником типу Item, припускаємо, що в

попередніх ділянках коду вказівник First посилався на перший елементсписку \*/

Схематично це можна зобразити так:

Видалення першого елемента списку можна здійснити так:

struct Item \*p = (struct Item\*)malloc(sizeof(struct Item));

p = First;

First = First->next; /\* тепер першим буде той елемент, який раніше був другим \*/

free(p); /\* звільняється пам'ять, яку займав перший елемент \*/

Схематично процес видалення першого елемента можна зобразити так:

struct Item \*p = (struct Item\*)malloc(sizeof(struct Item));

p = pp; p = p-next; /\* нехай pp – вже визначений раніше вказівник

на існуючий елемент списку \*/

pp->next = p->next; /\* тепер вказівник pp посилається на елемент

«через 1» \*/

Схематично видалення елемента виглядає так:

Якщо елемент посилається і на попередній, і на наступний елемент, то такий список є

двонапрямленим.

Якщо вказівник в останньому елементі не встановлений у NULL, а посилається на перший

елемент списку, то такий список називається кільцевим.

**Код програми**

Назва файлу: **sort.h**

void SortArray(struct student arr[], int size,

int (\*condition)(struct student\*, struct student\*));

int AgeUp(struct student \*a, struct student \*b);

Назва файлу: **table.h**

void ReadList(FILE \*f, struct student arr[], int size);

void PrintTable(FILE \*f, const struct student arr[], int size,

int (\*condition)(const struct student\*, float), float n);

float AverageRating(const struct student arr[], int size);

int isGreaterThanAverageRating(const struct student \*s,

float avr);

int True(const struct student \*s, float avr);

Назва файлу: **linked.h**

struct item \*MakeLinkedList(struct student arr[], int size);

void PrintLinkedListTable(FILE \*f, const struct item \*first);

void RemoveFromLinkedList(struct item \*\*pp,

int (\*cond)(struct student\*));

void InsertIntoLinkedList(struct item \*\*pp, struct student \*s,

int (\*cond)(struct student\*, struct student\*));

int llTrue(struct student \*s);

int hasNoFive(struct student \*s);

Назва файлу: **students.h**

struct student{

char surname[100];

char name[100];

int birthdate[3];

int ExamGrade[5];

};

struct item{

struct student student;

struct item \*next;

};

enum{

ExamGradesAmount = 5,

DateElements = 3

};

Назва файлу: **table.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "student.h"

#include "table.h"

enum{

SurnameSize = 14,

NameSize = 10,

};

void PrintSpaces(FILE \*f, int n)

{

for(int i = 0; i < n; i++)

fprintf(f, " ");

}

void PrintRow(FILE \*f, const struct student \*s, int n)

{

fprintf(f, "| %02d ", n);

fprintf(f, "| %s", s->surname);

PrintSpaces(f, SurnameSize - strlen(s->surname) / 2 - 1);

fprintf(f, "| %s", s->name);

PrintSpaces(f, NameSize - strlen(s->name) / 2 - 1);

fprintf(f, "| %02d.%02d.%04d ",

s->birthdate[0], s->birthdate[1], s->birthdate[2]);

fprintf(f, "| %02d ", s->ExamGrade[0]);

fprintf(f, "| %02d ", s->ExamGrade[1]);

fprintf(f, "| %02d ", s->ExamGrade[2]);

fprintf(f, "| %02d ", s->ExamGrade[3]);

fprintf(f, "| %02d |", s->ExamGrade[4]);

fprintf(f, "\n");

}

void PrintTable(FILE \*f, const struct student arr[], int size,

int (\*condition)(const struct student\*, float), float n)

{

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

fprintf(f, "| | | | дата |"\

" Список екзаменаційних оцінок |\n");

fprintf(f, "| № | Прізвище | Ім'я | |"\

"------------------------------|\n");

fprintf(f, "| | | | народження |"\

" ОП | КДМ | MA | ЛААГ | Іст. |\n");

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

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

if (condition(&arr[i], n))

PrintRow(f, &arr[i], i + 1);

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

}

float AverageRating(const struct student arr[], int size)

{

float avr = 0;

for(int i = 0; i < size; i++){

float sum = 0;

for(int j = 0; j < ExamGradesAmount; j++)

sum += arr[i].ExamGrade[j];

avr += sum / 5;

}

return avr / 27;

}

int isGreaterThanAverageRating(const struct student \*s, float avr)

{

float sum = 0;

for(int i = 0; i < ExamGradesAmount; i++)

sum += s->ExamGrade[i];

sum /= 5;

if (sum < avr)

return 1;

return 0;

}

int True(const struct student \*s, float avr)

{

return 1;

}

void ReadList(FILE \*f, struct student arr[], int size)

{

char str[200] = "";

for(int i = 0; i < size; i++){

fgets(str, 200, f);

char \*token = strtok(str, " \n");

strcpy(arr[i].surname, token);

token = strtok(NULL, " \n");

strcpy(arr[i].name, token);

for(int j = 0; j < 3; j++){

token = strtok(NULL, " .\n");

arr[i].birthdate[j] = atoi(token);

}

for(int j = 0; j < 5; j++){

token = strtok(NULL, " \n");

arr[i].ExamGrade[j] = atoi(token);

}

}

}

Назва файлу: **sort.c**

#include <string.h>

#include <stdlib.h>

#include "student.h"

#include "sort.h"

void swap(struct student \*a, struct student \*b)

{

struct student tmp = \*a;

\*a = \*b;

\*b = tmp;

}

void SortArray(struct student arr[], int size,

int (\*condition)(struct student\*, struct student\*))

{

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

for(int j = 0; j < size - i - 1; j++)

if(condition(&arr[j], &arr[j + 1]))

swap(&arr[j], &arr[j + 1]);

}

int MagicNumber(struct student \*s)

{

int res = 0, coef = 1;

for(int i = 0; i < DateElements; i++){

res += s->birthdate[i] \* coef;

coef \*= 100;

}

return res;

}

int AgeUp(struct student \*a, struct student \*b)

{

if(MagicNumber(a) < MagicNumber(b))

return 1;

return 0;

}

Назва файлу: **linked.c**

#include <stdio.h>

#include <stdlib.h>

#include "student.h"

#include "table.h"

#include "linked.h"

struct item \*MakeLinkedList(struct student arr[], int size)

{

struct item \*first = NULL, \*last = NULL, \*tmp;

for(int i = 0; i < size; i++){

tmp = malloc(sizeof(struct item));

tmp->student = arr[i];

tmp->next = NULL;

if(last){

last->next = tmp;

last = last->next;

}else{

first = last = tmp;

}

}

return first;

}

void PrintLinkedListTable(FILE \*f, const struct item \*first)

{

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

fprintf(f, "| | | | дата |"\

" Список екзаменаційних оцінок |\n");

fprintf(f, "| № | Прізвище | Ім'я | |"\

"------------------------------|\n");

fprintf(f, "| | | | народження |"\

" ОП | КДМ | MA | ЛААГ | Іст. |\n");

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

struct item \*tmp = first;

int i = 1;

while(tmp){

PrintRow(f, &tmp->student, i);

tmp = tmp->next;

i++;

}

fprintf(f, "---------------------------------------------"\

"-------------------------------\n");

}

void InsertIntoLinkedList(struct item \*\*pp, struct student \*s,

int (\*cond)(struct student\*, struct student\*))

{

while(\*pp && cond(s, &(\*pp)->student))

pp = &(\*pp)->next;

struct item \*tmp = malloc(sizeof(struct item));

tmp->next = \*pp;

tmp->student = \*s;

\*pp = tmp;

}

void RemoveFromLinkedList(struct item \*\*pp,

int (\*cond)(struct student\*))

{

while(\*pp){

if(cond(&(\*pp)->student)){

struct item \*tmp = \*pp;

\*pp = (\*pp)->next;

free(tmp);

}else{

pp = &(\*pp)->next;

}

}

}

int llTrue(struct student \*s)

{

return 1;

}

int hasNoFive(struct student \*s)

{

for(int i = 0; i < ExamGradesAmount; i++)

if(s->ExamGrade[i] == 5)

return 0;

return 1;

}

Назва файлу: **main.c**

#include <stdio.h>

#include "student.h"

#include "sort.h"

#include "table.h"

#include "linked.h"

#define LIST\_SIZE 27

int main()

{

FILE \*f;

struct student smb =

{

"Моставчук", "Петро",

{1, 1, 2005},

{12, 12, 12, 12, 12}

};

struct student arr[LIST\_SIZE];

struct item \*ll = NULL;

f = fopen("list.txt", "r");

ReadList(f, arr, LIST\_SIZE);

fclose(f);

SortArray(arr, LIST\_SIZE, AgeUp);

ll = MakeLinkedList(arr, LIST\_SIZE);

InsertIntoLinkedList(&ll, &smb, AgeUp);

RemoveFromLinkedList(&ll, hasNoFive);

InsertIntoLinkedList(&ll, &smb, AgeUp);

f = fopen("no5.txt", "w");

PrintLinkedListTable(f, ll);

fclose(f);

return 0;

}

**Протокол роботи**

**main.c**



**Висновки**

Однозвʼязний список є корисним, коли потрібно вставити або видалити елемент, проте знаходження потрбного елемента може потребувати повний прохід по списку.