Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних систем

**КУРСОВА РОБОТА**

з дисципліни «Алгоритмізація та програмування»

Тема «Програмування динамічної структури даних – однозв’язний список»

Студента (ки) \_\_1\_\_ курсу АІ-202 групи

Спеціальності 122 – «Комп’ютерні науки»

\_\_\_\_\_\_\_Сідельникова М.В.\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник доцент, к.т.н. Бабілунга О.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали

м. Одеса – 2021 рік

Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

студенту Сідельникову Микиті Валерійовичу група АІ-202

1. Тема роботи

«Програмування динамічної структури даних – однозв’язний список»

2. Термін здачі студентом закінченої роботи 03.06.2021

3. Початкові дані до проекту (роботи)

Варіант 20

Структура:

* прізвище пацієнта;
* номер облікової картки;
* вид роботи (протезування, пломбування, імплантація і ін.);
* вартість виконаної роботи;
* відмітка про оплату;
* сума заборгованості за лікування.

Програма повинна виконувати:

* додавання елемента;
* видалення елемента;
* можливість коригування даних;
* виведення всіх даних;
* виведення довідки про клієнта за введеним прізвищем;
* виведення довідки про боржників за видами робіт і загальної заборгованості;
* видалення з бази записів про пацієнтів, які не мають заборгованості за лікування;
* сортування за сумою заборгованості за лікування.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)

Вступ.

Теоретичні відомості про однозв’язний список.

Програмна реалізація однозв’язного списку.

Інструкція користувача.

Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Блок-схема алгоритму – 1 аркуш формату А1.

Завдання видано 16.03.21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Завдання прийнято до виконання 16.03.21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

КОД ПРОГРАМИ

main.c:

#include <stdlib.h>

#include <stdio.h>

#include <Windows.h>

#include "patientsList.h"

#include "menu.h"

List patients\_list;

void executor(MenuElement command);

void add\_client();

void get\_client();

void print\_all();

void print\_debtors();

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

while (1) {

print\_menu();

MenuElement command = scan\_command();

executor(command);

}

destructor(&patients\_list);

return 0;

}

void executor(MenuElement command) {

switch(command) {

case EXIT:

printf("\nРоботу програми зупинено.");

exit(0);

break;

case REMOVE\_CLIENT:

case MODIFY\_CLIENT:

case REMOVE\_DEBTLESS:

case SORT\_BY\_DEBT:

printf("\nФункція у розробці.");

press\_to\_continue();

break;

case ADD\_CLIENT:

add\_client();

break;

case PRINT\_ALL:

print\_all();

break;

case PRINT\_CLIENT:

get\_client();

break;

case PRINT\_DEBTORS:

print\_debtors();

break;

default:

printf("\nПомилка вводу: команда не існує. Спробуйте ще раз");

}

}

void add\_client() {

printf("\nДодавання пацієнта до списку. Оберіть спосіб вводу: \n1. Через термінал\n2. Через файл\n");

unsigned method;

while (1) {

scanf("%u", &method);

if (method != 1 && method != 2)

printf("Помилка вводу: такого способу не існує. Спробуйте ще раз: ");

else

break;

}

PatientData \*data;

if (method == 1) {

Status status1 = write\_from\_console(&data);

if (status1 == NULL\_POINTER) {

printf("\nНе вдалося прочитати інформацію.");

return;

}

}

else {

printf("\nФункція у розробці");

return;

}

Status status = push(&patients\_list, data);

switch (status) {

case OK:

printf("\nПацієнта успішно додано.");

break;

case NULL\_POINTER:

printf("\nПомилка у роботі програми.");

break;

case ELEMENT\_ALREADY\_EXISTS:

printf("\nПацієнт з таким ім'ям або номером картки вже існує.");

break;

default:

printf("\nНевідома помилка.");

}

press\_to\_continue();

}

void get\_client() {

printf("\nВивести інформацію про клієнта. Оберіть спосіб виводу: \n1. Через термінал\n2. Через файл\n");

unsigned method;

while (1) {

scanf("%u", &method);

if (method != 1 && method != 2)

printf("Помилка вводу: такого способу не існує. Спробуйте ще раз: ");

else

break;

}

if (method == 2) {

printf("\nФункція у розробці.");

return;

}

char name[150];

printf("\nВведіть ім'я:\n");

getchar();

gets(name);

ListNode \*client;

Status status = get\_element\_by\_name(&patients\_list, name, &client);

switch(status) {

case NULL\_POINTER:

printf("\nПомилка у роботі програми.");

break;

case ELEMENT\_NOT\_FOUND:

printf("\nПацієнта з таким ім'ям немає у списку.");

break;

case OK: {

Status status1 = write\_to\_console(client -> data);

if (status1 == NULL\_POINTER) {

printf("\nПомилка у роботі програми.");

return;

}

print\_line(70);

press\_to\_continue();

break;

}

default:

printf("\nНевідома помилка.");

}

}

void print\_all() {

printf("\nСписок клієнтів:\n");

Status status = print\_list(&patients\_list);

switch(status) {

case NO\_ELEMENTS:

printf("\nСписок пустий...");

break;

case NULL\_POINTER:

printf("\nПомилка в роботі програми.");

break;

case OK:

print\_line(70);

break;

default:

printf("\nНевідома помилка.");

}

press\_to\_continue();

}

void print\_debtors() {

printf("За яким крітерієм відобразити боржників? \n1. Тип робіт \n2. Сума боргу\n");

unsigned method;

while (1) {

scanf("%u", &method);

if (method != 1 && method != 2)

printf("Помилка вводу: такого способу не існує. Спробуйте ще раз: ");

else

break;

}

printf("\nСписок боржників:\n");

Status status;

switch (method) {

case 1:

status = print\_debtors\_type(&patients\_list);

break;

case 2:

status = print\_debtors\_debt(&patients\_list);

break;

}

switch(status) {

case NO\_ELEMENTS:

printf("\nБоржників немає...");

break;

case NULL\_POINTER:

printf("\nПомилка в роботі програми.");

break;

case OK:

print\_line(70);

break;

default:

printf("\nНевідома помилка.");

}

press\_to\_continue();

}

menu.h:

typedef enum MenuElement {

EXIT,

ADD\_CLIENT,

REMOVE\_CLIENT,

MODIFY\_CLIENT,

PRINT\_ALL,

PRINT\_CLIENT,

PRINT\_DEBTORS,

REMOVE\_DEBTLESS,

SORT\_BY\_DEBT,

} MenuElement;

void print\_line(unsigned len);

void print\_line\_to\_file(unsigned len, FILE \*f);

void print\_menu();

MenuElement scan\_command();

void press\_to\_continue();

menu.c:

#include <stdio.h>

#include "menu.h"

#define ITEMS\_AMOUNT 10

#define ITEM\_MAX\_LENGTH 110

#define MENU\_WIDTH 70

char menu\_items[ITEMS\_AMOUNT][ITEM\_MAX\_LENGTH] = {

"Вибір дії",

"1. Додати клієнта до списку",

"2. Видалити клієнта зі списку",

"3. Корегувати дані клієнта",

"4. Вивести список повністю",

"5. Вивести довідку клієнта",

"6. Вивести довідку про боржників",

"7. Видалити записи клієнтів, які не мають заборгованості",

"8. Сортувати записи за сумою заборгованості",

"0. Вихід з програми",

};

void print\_line(unsigned len) {

printf("\n");

for (unsigned i = 0; i < len; i++)

printf("-");

}

void print\_line\_to\_file(unsigned len, FILE \*f) {

if (!f)

return;

fprintf(f, "\n");

for (unsigned i = 0; i < len; i++)

fprintf(f, "-");

}

void print\_menu() {

printf("\n");

print\_line(MENU\_WIDTH);

printf("\n\t%s", menu\_items[0]);

print\_line(MENU\_WIDTH);

for (int i = 1; i < ITEMS\_AMOUNT; i++)

printf("\n\t%s", menu\_items[i]);

print\_line(MENU\_WIDTH);

printf("\n");

}

MenuElement scan\_command() {

printf("\nВведіть номер команди: ");

unsigned command;

scanf("%u", &command);

return command <= ITEMS\_AMOUNT ? command : 10;

}

void press\_to\_continue() {

printf("\nНатисніть Enter щоб продовжити... ");

getchar();

while (getchar() != '\n') {}

}

patientsList.h:

typedef enum WorkType {

PROSTHETICS = 1,

FILLINGS,

IMPLANTS,

INSPECTION,

EXTRACTION

} WorkType;

typedef enum Status {

OK,

NULL\_POINTER,

SIZE\_LIMIT,

ELEMENT\_ALREADY\_EXISTS,

INPUT\_ERROR,

ELEMENT\_NOT\_FOUND,

NO\_ELEMENTS

} Status;

typedef struct PatientData {

char name[150];

unsigned card\_number;

WorkType type;

unsigned cost;

unsigned paid;

unsigned debt;

} PatientData;

typedef struct ListNode {

PatientData \*data;

struct ListNode \*next;

} ListNode;

typedef struct List {

ListNode \*head;

} List;

Status destructor(List \*list);

Status push(List \*list, PatientData \*node);

Status write\_from\_console(PatientData \*\*result);

Status get\_element\_by\_name(List \*list, char name[], ListNode \*\*result);

Status write\_to\_console(PatientData \*data);

Status print\_list(List \*list);

Status print\_debtors\_type(List \*list);

Status print\_debtors\_debt(List \*list);

patientsList.c:

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "patientsList.h"

#include "menu.h"

#define LINE\_WIDTH 70

char types[5][60] = {

"Протезування",

"Пломбування",

"Імплантація",

"Звичайний огляд",

"Видалення зуба"

};

Status destructor(List \*list) {

if (!list)

return NULL\_POINTER;

ListNode \*tmp;

while (!list -> head) {

tmp = list -> head -> next;

free(list -> head);

list -> head = tmp;

}

return OK;

}

Status push(List \*list, PatientData \*data) {

if (!list || !data)

return NULL\_POINTER;

ListNode \*cur = list -> head;

while (cur) {

if (!strcmp(cur -> data -> name, data -> name) || cur -> data -> card\_number == data -> card\_number)

return ELEMENT\_ALREADY\_EXISTS;

cur = cur -> next;

}

ListNode \*tmp = (ListNode \*) malloc(sizeof(ListNode));

tmp -> data = data;

tmp -> next = list -> head;

list -> head = tmp;

return OK;

}

Status write\_from\_console(PatientData \*\*result) {

if (!result)

return NULL\_POINTER;

printf("\nВвод даних:");

PatientData \*tmp = (PatientData \*) malloc(sizeof(PatientData));

printf("\nВведіть ім'я та/або прізвище пацієнта:\n");

getchar();

gets(tmp -> name);

printf("\nВведіть номер картки пацієнта: ");

scanf("%u", &(tmp -> card\_number));

printf("\nВиди робіт:");

for (int i = 0; i < 5; i++)

printf("\n%d. %s", i + 1, types[i]);

unsigned type;

while (1) {

printf("\nВведіть номер виду роботи: ");

scanf("%u", &type);

if (type <= 5 && type != 0)

break;

else

printf("Помилка вводу: такого виду робіт не існує. Спробуйте інший варіант: ");

}

tmp -> type = type;

printf("\nВведіть вартість виконаної роботи (у грн): ");

scanf("%u", &(tmp -> cost));

printf("\nВведіть оплачену суму (у грн): ");

scanf("%u", &(tmp -> paid));

if (tmp -> paid < tmp -> cost) {

tmp -> debt = tmp -> cost - tmp -> paid;

printf("\nЗаборгованність становить %d грн.", tmp -> debt);

} else {

tmp -> debt = 0;

printf("\nЗаборгованності немає.");

}

\*result = tmp;

return OK;

}

Status get\_element\_by\_name(List \*list, char name[], ListNode \*\*result) {

if (!list || !result)

return NULL\_POINTER;

ListNode \*cur = list -> head;

while (cur) {

if (!strcmp(cur -> data -> name, name)) {

\*result = cur;

return OK;

}

cur = cur -> next;

}

return ELEMENT\_NOT\_FOUND;

}

Status write\_to\_console(PatientData \*data) {

if (!data)

return NULL\_POINTER;

print\_line(LINE\_WIDTH);

printf("\nІм'я: %s", data -> name);

printf("\nНомер картки: %d", data -> card\_number);

printf("\nТип послуг: %s", types[data -> type]);

printf("\nВартість роботи: %d грн", data -> cost);

printf("\nОплачено: %d грн", data -> paid);

printf("\nЗаборгованність: %d грн", data -> debt);

return OK;

}

Status print\_list(List \*list) {

if (!list)

return NULL\_POINTER;

ListNode \*cur = list -> head;

if (!cur)

return NO\_ELEMENTS;

while (cur) {

write\_to\_console(cur -> data);

cur = cur -> next;

}

return OK;

}

Status print\_debtors\_type(List \*list) {

if (!list)

return NULL\_POINTER;

ListNode \*cur = list -> head;

unsigned empty = 1;

while (cur) {

if (cur -> data -> debt) {

empty = 0;

print\_line(LINE\_WIDTH);

printf("\n%s: %s", cur -> data -> name, types[cur -> data -> type]);

}

cur = cur -> next;

}

if (empty)

return NO\_ELEMENTS;

return OK;

}

Status print\_debtors\_debt(List \*list) {

if (!list)

return NULL\_POINTER;

ListNode \*cur = list -> head;

unsigned empty = 1;

while (cur) {

if (cur -> data -> debt) {

empty = 0;

print\_line(LINE\_WIDTH);

printf("\n%s: %u", cur -> data -> name, cur -> data -> debt);

}

cur = cur -> next;

}

if (empty)

return NO\_ELEMENTS;

return OK;

}

СКРИНШОТИ РЕЗУЛЬТАТІВ РОБОТИ

































