МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут комп'ютерних наук та інформаційних технологій

Кафедра програмного забезпечення

ЗВІТ

До лабораторної роботи № 2

На тему: *"Документування етапів проектування та кодування програми"*

З дисципліни: *"Вступ до інженерії програмного забезпечення"*

Лектор:

доцент Левус Є. В.

Виконав:

ст. гр. ПЗ-15

Хвещук І.С.

Прийняв:

асист. Самбір А. А.

« ____ » _____ 2022 р.

∑= ____ .....

Львів – 2022

Тема: документування етапів проектування та кодування програми.

Мета: навчитися документувати основні результати етапів проектування та кодування найпростіших програм.

Теоретичні відомості

Варіант 24

Питання 28(Як записуються класи та їх складові у мові С++?)

Для опису класу використовують ключове слово class. Існують три модифікатори доступи private, protected, public. Для збереження принципу інкапсуляції дані класу повинні бути недоступні поза межами класу, а працювати з ними можна тільки за допомогою методів цього класу.

Приклад класу:

```
class Human
{
size_t age;
std::string name;
public:
        Human();
        Human(const size_t age, const std::string& name);
        Human(const Human& copyHuman);
        size_t getAge() const;
};
```

Питання 16(Скільки входів та виходів має блок циклу? Відповідь пояснити.)

Зазвичай блок циклу має одну точку входу і одну або декілька виходу. У певних ситуаціях цикл може не мати точки виходу. У екзотичних випадках через оператор goto і break можна зробити декілька точок входу та виходу.

Питання 10

Алгоритм – це певна послідовність дій виконання яких вирішує поставлену задачу. Яскравим прикладом алгоритму може слугувати алгоритм Евкліда для пошуку НСД. Алгоритм звучить так:

Допоки а != в

повторяти

якщо а>в, то а -= в

інакше в -= а

кінець

Завдання

Частина І. У розробленій раніше програмі до лабораторної роботи з дисципліни «Основи програмування» внести зміни – привести її до модульної структури, де модуль – окрема функція-підпрограма. У якості таких функцій запрограмувати алгоритми зчитування та запису у файл, сортування, пошуку, редагування, видалення елементів та решта функцій згідно варіанту.

Частина ІІ. Сформувати пакет документів до розробленої раніше власної програми:

1. схематичне зображення структур даних, які використовуються для збереження інформації ;

2. блок-схема алгоритмів – основної функції й двох окремих функційпідпрограм (наприклад, сортування та редагування);

3. текст програми з коментарями та оформлений згідно вище наведених рекомендацій щодо забезпечення читабельності й зрозумілості.

Для схематичного зображення структур даних, блок-схеми алгоритму можна використати редактор MS-Visio або інший редактор інженерної та ділової графіки.

1. **схематичне зображення структур даних, які використовуються для збереження інформації ;**

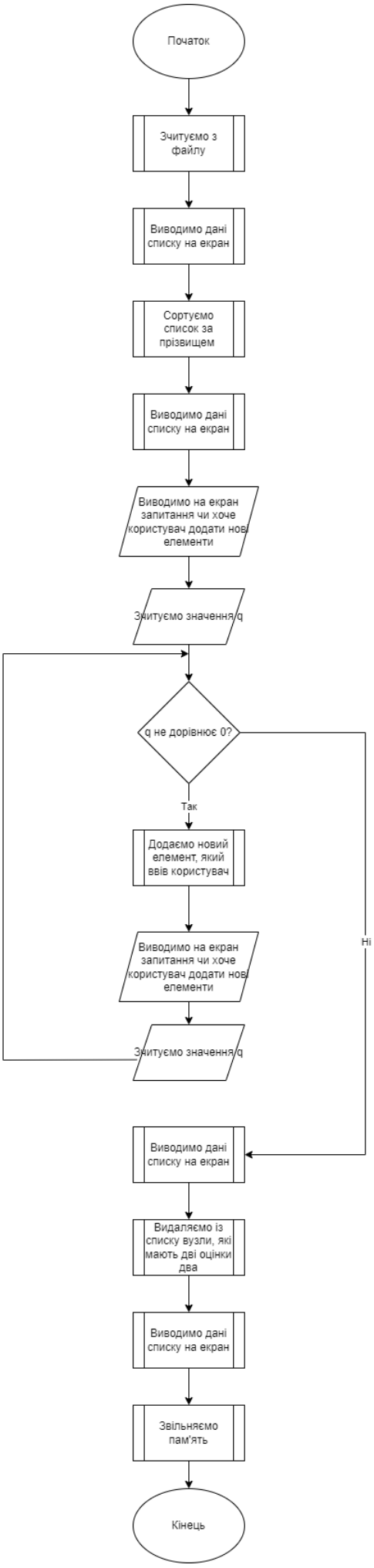
```
2. #define STR_LEN 20
3. #define MARKS 5
```

StudentInformation
char surname[STR_LEN] char name[STR_LEN] char date[STR_LEN] int marks[MARKS]

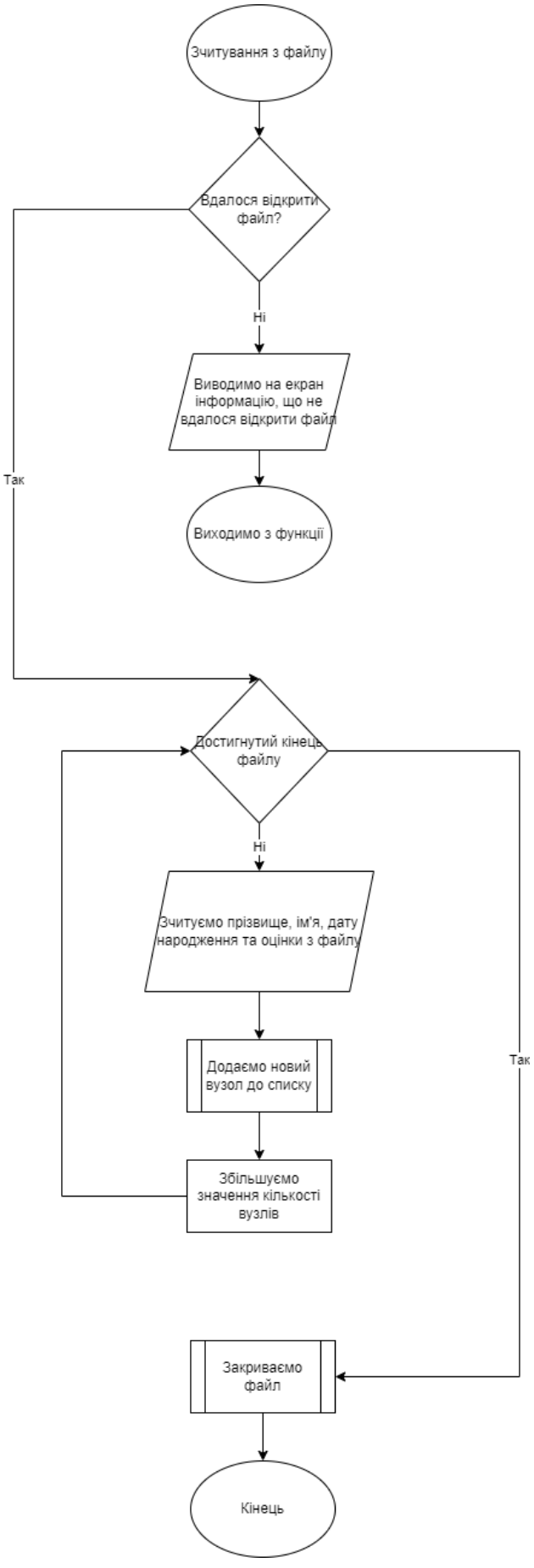
SessiaList
studentInformation student struct SessiaList *nextNode

2. блок-схема алгоритмів – основної функції й двох окремих функційпідпрограм (наприклад, сортування та редагування);

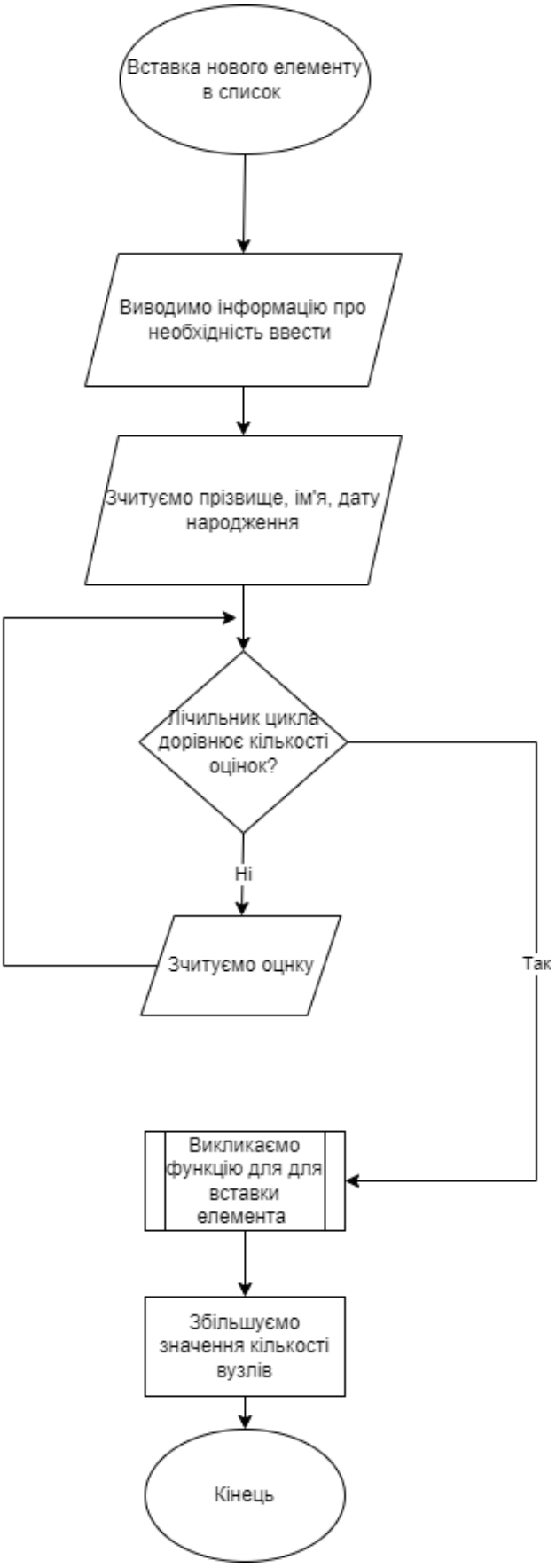
Основна функція


Початок
Зчитуємо з файлу
Виводимо дані списку на екран
Сортуємо список за прізвищем
Виводимо дані списку на екран
Виводимо на екран запитання чи хоче користувач додати нові елементи
Зчитуємо значення q
q не дорівнює 0?
Так
Ні
Додаємо новий елемент, який ввів користувач
Виводимо на екран запитання чи хоче користувач додати нові елементи
Зчитуємо значення q
Виводимо дані списку на екран
Видаляємо із списку вузли, які мають дві оцінки два
Виводимо дані списку на екран
Звільняємо пам'ять
Кінець

Функція для зчитування з файлу



Вставка нового елемента в список



Код програми

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "MYFUNC.h"
int main(void)
{
    sessiaList *beginOfStudentList = NULL;

    readFromFile(&beginOfStudentList);
    printSessionList(&beginOfStudentList);
    sortBySurname(&beginOfStudentList);
    printSessionList(&beginOfStudentList);
    int wantToAdd = 0;
    _getch();
    system("cls");
    printf("Enter q (if q > 0, you'll add new member)");
    scanf("%d",&wantToAdd);

    while(wantToAdd){
    addNewNodeOfListByUser(&beginOfStudentList);
    _getch();
    system("cls");
    printf("Enter q:\n");
    scanf("%d",&wantToAdd);
    }

    printSessionList(&beginOfStudentList);
    deleteStudentsWithTwoMarksTwo(&beginOfStudentList);
    printSessionList(&beginOfStudentList);
    free(&beginOfStudentList);
    return 0;
}
```

listFunctions.h

```c
#ifndef MYFUNC_H_INCLUDED
#define MYFUNC_H_INCLUDED
#include "MyFunc.c"
#define STR_LEN 20
#define MARKS 5

struct StudentInformation;
struct SessiaList;

studentInformation temporaryNodeOfList;

void addNewNode(studentInformation student, sessiaList **begin);
int isEmpty(sessiaList **beginOfList);
void printSessionList(sessiaList **beginOfList);
void sortBySurname(sessiaList **beginOfList);
void deleteStudentsWithTwoMarksTwo(sessiaList **beginOfList);
void addAfterSort(studentInformation student, sessiaList **beginOfList);
void addNewNodeOfListByUser(sessiaList **beginOfList);
void readFromFile(sessiaList **beginOfList);

#endif // MYFUNC_H_INCLUDED
```

listFunctions.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define STR_LEN 20
#define MARKS 5

typedef struct StudentInformation
{
```

```
    char surname[STR_LEN];
    char name[STR_LEN];
    char date[STR_LEN];
    int marks[MARKS];

}studentInformation;

typedef struct SessiaList
{

    studentInformation student;
    struct SessiaList *nextNode;

}sessiaList;

int countOfNodes = 0;
studentInformation tempNode;

void addNewNode(studentInformation student, sessiaList **begin)//процедура для додавання нового вузла у однозв'язний
список
{
    sessiaList *beginCopy = *begin, *newNodeOfList = (sessiaList *)malloc(sizeof(sessiaList));

    if (newNodeOfList == NULL)
    {
        puts("Error with memory.");
        _getch();
        exit(2);
    }

    newNodeOfList->student = student;
    newNodeOfList->nextNode = NULL;

    if (beginCopy == NULL)
        *begin = newNodeOfList;
    else
    {
        while (beginCopy->nextNode != NULL)
            beginCopy = beginCopy->nextNode;
        beginCopy->nextNode = newNodeOfList;
    }
}

int isEmpty(sessiaList **beginOfList)//функція для перевірки чи однозв'язний список містить елементи
{
    if (*beginOfList == NULL)
        return 1;
    else
        return 0;
}

void printSessionList(sessiaList **beginOfList)//процедура для виводу вмісту однозв'язного списку на екран
{
    _getch();
    system("cls");

    if (isEmpty(beginOfList))
    {
        puts("List is empty!");
        return;
    }
    else
    {
        printf("     SURNAME       NAME           DATE             MARKS \n");
        sessiaList *nodesIterator = *beginOfList;

        while (nodesIterator != NULL)
        {
            printf("|%10s| |%12s| \t|%5s|\t ", nodesIterator->student.surname, nodesIterator->student.name,
nodesIterator->student.date);

            for (int currentMark = 0; currentMark <MARKS; ++currentMark)
```

```
                printf("|% d|", nodesIterator->student.marks[currentMark]);

            printf("\n");
            nodesIterator = nodesIterator->nextNode;
        }
    }
}

void sortBySurname(sessiaList **beginOfList)//процедура для сортування списку за алфавітом по прізвищу
{
    _getch();
    system("cls");
    if (isEmpty(beginOfList))
    {
        puts("List is empty!");
        return;
    }
    else
    {
        sessiaList *copyOfBegin = *beginOfList;
        for (int currentPositionOfNode = 1; currentPositionOfNode < countOfNodes; currentPositionOfNode++)
        {
            copyOfBegin = *beginOfList;
            while (copyOfBegin->nextNode != NULL)
            {
                if (strcmp(copyOfBegin->student.surname, copyOfBegin->nextNode->student.surname) > 0)
                {
                    temporaryNodeOfList = copyOfBegin->student;
                    copyOfBegin->student = copyOfBegin->nextNode->student;
                    copyOfBegin->nextNode->student = temporaryNodeOfList;
                }
                copyOfBegin = copyOfBegin->nextNode;
            }
        }
        puts("List has sorted by surname.");
    }
}

void deleteStudentsWithTwoMarksTwo(sessiaList **beginOfList)//процедура для видалення елементів списку, які містять
два елемента marks зі значенням 2
{
    _getch();
    system("cls");

    if (isEmpty(beginOfList))
    {
        puts("List is empty!");
        return;
    }
    else
    {
        printf("\n    There are list of students, which have two marks '2'\n\n");
        printf("     SURNAME       NAME           DATE             MARKS \n");
        sessiaList *copyOfBegin = *beginOfList, *previousNode = NULL, *nextNodeOfList = NULL;

        while (copyOfBegin != NULL)
        {
            int counterOfMarks = 0;

            for (int currentMark = 0; currentMark <MARKS; ++currentMark)
                if (copyOfBegin->student.marks[currentMark] == 2)
                    ++counterOfMarks ;

            if (counterOfMarks == 2)
            {
                printf("%10s %12s \t%s\t ", copyOfBegin->student.surname, copyOfBegin->student.name, copyOfBegin-
>student.date);

                for (int currentMark = 0; currentMark <MARKS; currentMark++)
                    printf("% d", copyOfBegin->student.marks[currentMark]);

                printf("\n");
            }
```

```
            copyOfBegin = copyOfBegin->nextNode;
        }

        copyOfBegin = *beginOfList;

        while (copyOfBegin != NULL)
        {
            int counterOfMarks = 0;

            for (int currentMark = 0; currentMark <MARKS; ++currentMark)
                if (copyOfBegin->student.marks[currentMark] == 2)
                    ++counterOfMarks;

            if (counterOfMarks == 2 && previousNode == NULL)
            {
                nextNodeOfList = copyOfBegin->nextNode;
                free(copyOfBegin);
                copyOfBegin = nextNodeOfList;
                *beginOfList = copyOfBegin;
                countOfNodes--;
            }
            else if (counterOfMarks == 2 && previousNode != NULL)
            {
                nextNodeOfList = copyOfBegin->nextNode;
                free(copyOfBegin);
                copyOfBegin = nextNodeOfList;
                previousNode->nextNode = copyOfBegin;
                countOfNodes--;
            }
            else
            {
                previousNode = copyOfBegin;
                copyOfBegin = copyOfBegin->nextNode;
            }
        }
    }
}

void addAfterSort(studentInformation studentInfo, sessiaList **beginOfList)//процедура для додачі нового вузла у
відсортований список без порушення сортування
{
    sessiaList *copyOfBegin = *beginOfList, *newNode = (sessiaList *)malloc(sizeof(sessiaList));
    int resOfCompareSurnames = 0;

    if (newNode == NULL)
    {
        puts("Error with memory.");
        _getch();
        exit(2);
    }

    newNode->student = studentInfo;
    newNode->nextNode = NULL;

    if (copyOfBegin == NULL)
        *beginOfList = newNode;

    else
    {
        while (copyOfBegin->nextNode != NULL && (resOfCompareSurnames = strcmp(copyOfBegin->nextNode-
>student.surname,newNode->student.surname))<0)
        {
           copyOfBegin = copyOfBegin->nextNode;
        }

            newNode->nextNode = copyOfBegin->nextNode;
            copyOfBegin->nextNode = newNode;

    }
}

void addNewNodeOfListByUser(sessiaList **beginOfList)//процедура для додавання нового вузла введеного користувачем
{
```

```
    _getch();
    system("cls");
    studentInformation nodeForAdding;
    printf("Enter surname:\n");
    scanf("%s",&nodeForAdding.surname);
    printf("Enter name:\n",&nodeForAdding.name);
    scanf("%s",&nodeForAdding.name);
    printf("Enter date:\n");
    scanf("%s",&nodeForAdding.date);
    printf("Enter marks:\n");

    for(int currentMark = 0; currentMark < MARKS; ++currentMark)
        scanf("%d",&nodeForAdding.marks[currentMark]);

    addAfterSort(nodeForAdding, beginOfList);
    countOfNodes++;
}


void readFromFile(sessiaList **beginOfList)//процедура для зчитування даних з файлу
{
    FILE *userFile;

    if ((userFile = fopen("students.txt", "r")) == NULL)
    {
        puts("Error open file.");
        _getch();
        exit(1);
    }
    while (!feof(userFile))
    {
        fscanf(userFile, "%s %s %s", temporaryNodeOfList.surname, temporaryNodeOfList.name,
temporaryNodeOfList.date);
        for (int currentMark = 0; currentMark <MARKS; ++currentMark)
            fscanf(userFile, "%d", &temporaryNodeOfList.marks[currentMark]);

        addNewNode(temporaryNodeOfList, beginOfList);
        countOfNodes++;
    }
    fclose(userFile);
}
```

Висновок: під час виконання лабораторної роботи я ближче познайомився з деякими етапами життєвого циклу ПЗ – проектування та кодування. Також сформував певну документацію до раніше розробленої програми, а також схематичне зображення структур даних, блок-схеми алгоритмів.