Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет комп’ютерних систем та автоматики

Кафедра КСУ

Лабораторна робота №8

з дисципліни: «Компютерні технології та програмування»

Тема: «Класи. Спадкування. Дружні функції»

Виконав

студент групи 2АКІТ-20б

Н. Бондар

Перевірив

к.т.н., доц.кафедри КСУ

Севастьянов В. М.

м.Вінниця, ВНТУ 2021

**Лабораторна робота №8**

**Тема: «Класи. Спадкування. Дружні функції»**

**Варіант 1.**

**Завдання :**

Загальні умови:

1. Створити клас-нащадок класу розробленого в попередній лабораторній роботі.

Відповідно до варіанту добавити у клас-нащадок відповідні функції-члени, дані-члени та

дружні функції.

2. Написати програму, яка буде демонструвати роботу з функціями-членами класу

та дружньою функцією. Для демонстрації роботи програма повинна містити меню. В

програмі використати один екземпляр класу для кожного конструктора.

**Варіант 1. Клас «Залікова книжка»**

Створити константну функцію-член, яка буде виводити на екран інформацію по

дисциплінам з балом вищим чим вказаний в аргументі функції.

Створити дружню функцію, яка буде рахувати і повертати середній бал по всім

дисциплінам.

**Хід роботи :**

**1. Ретельно вивчаю поставлене передімною завдання.**

**2. Створюю програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:**

**3. Відкомпільовую та відлагоджую програму.**

**4. Розробляю набір тестів і перевіряю роботу програми на них.**

**5. Відповідаю на контрольні запитання.**

**6. Роблю висновки.**

**1. Розписую структуру Discipline :**

struct Discipline {

char \* discipline;

int mark;

char \* teacher\_last\_name;

const char \* faculty = "FKSA";

void show\_faculty() {

std::cout << "Faculty : " << faculty << std::endl;

}

Discipline() {

discipline = NULL;

mark = 0;

teacher\_last\_name = NULL;

}

Discipline(char \* discipline, int mark, char \* teacher\_last\_name) {

//std::cout << "Create an object !!!\n";

this->discipline = new char[lenght(discipline)];

this->teacher\_last\_name = new char[lenght(teacher\_last\_name)];

copy(discipline, this->discipline);

copy(teacher\_last\_name, this->teacher\_last\_name);

this->mark = mark;

}

};

**2. Розписую клас student\_gradebook :**

class student\_gradebook {

public:

char \* last\_name;

char \* gradebook;

int subjects\_number;

Discipline \* discipline;

void config(char \* last\_name, char \* gradebook) {

//std::cout << "Create an object !!!\n";

this->last\_name = new char[lenght(last\_name)];

this->gradebook = new char[lenght(gradebook)];

copy(last\_name, this->last\_name);

copy(gradebook, this->gradebook);

}

bool config(char \* file\_name) {

char char\_temp[100];

int int\_temp;

FILE \* file = fopen(file\_name, "r");

if (!file){

std::cout << "!!! Wrong file name !!!\n";

return false;

}

fscanf(file, "%i\n", &subjects\_number);

fscanf(file, "%s\n", char\_temp);

last\_name = new char[lenght(char\_temp)];

copy(char\_temp, this->last\_name);

fscanf(file, "%s\n", char\_temp);

gradebook = new char[lenght(char\_temp)];

copy(char\_temp, this->gradebook);

this->discipline = new Discipline[subjects\_number];

// std::cout << "\nSubjects\_number : " << subjects\_number << "\nStudent's last name : " << last\_name << "\nStudent's gradebook : " << gradebook << "\n\n";

for (int i = 0; i < subjects\_number; i++) {

fscanf(file, "%s\n", char\_temp);

// std::cout << "Subject : " << char\_temp << '\n';

discipline[i].discipline = new char[strlen(char\_temp)];

strcpy(discipline[i].discipline, char\_temp);

fscanf(file, "%i\n", &int\_temp);

// std::cout << "Mark : " << int\_temp << '\n';

discipline[i].mark = int\_temp;

fscanf(file, "%s\n", char\_temp);

// std::cout << "Teacher : " << char\_temp << "\n\n";

discipline[i].teacher\_last\_name = new char[strlen(char\_temp)];

strcpy(discipline[i].teacher\_last\_name, char\_temp);

}

fclose(file);

return true;

}

};

**3. Розпусую клас-нащадок child\_class :**

class child\_class : public student\_gradebook {

public:

void show\_by\_marks(int mark) {

std::cout << "Student\'s last name : " << last\_name << "\nGradebook number : " << gradebook << std::endl << '\n';

std::cout << "Mark height than " << mark << " is :\n\n";

for (int i = 0; i < subjects\_number; i++) {

if (discipline[i].mark > mark) {

std::cout << "Subject : " << discipline[i].discipline << '\n';

std::cout << "Mark : " << discipline[i].mark << '\n';

std::cout << "Teacher : " << discipline[i].teacher\_last\_name << "\n\n";

}

}

}

void show\_all() {

std::cout << "Student\'s last name : " << last\_name << "\nGradebook number : " << gradebook << std::endl << '\n';

for (int i = 0; i < subjects\_number; i++) {

std::cout << "Subject : " << discipline[i].discipline << '\n';

std::cout << "Mark : " << discipline[i].mark << '\n';

std::cout << "Teacher : " << discipline[i].teacher\_last\_name << "\n\n";

}

}

float average(child\_class &);

};

**4. Розписуюю функцію-друга для цього класа :**

float average(child\_class &child) {

float average = 0;

for (int i = 0; i < child.subjects\_number; i++) {

average += child.discipline[i].mark;

}

return average / child.subjects\_number;

}

**5. Розписую головну функцію та додаткове меню для навігації по програмі :**

// student\_gradebook gradebook("Ivanov V.V.", "123456");

// student\_gradebook gradebook("zk.txt");

// Discipline discipline("Phisics", 5, "Ivanov");

// child\_class children;

child\_class gradebook;

// gradebook.config("zk.txt");

int user\_choice;

char file\_name[20];

int value;

ups:;

std::cout << "Menu :\n 0. Exit\n 1. Read file\n 2. Show all\n 3. Show by mark\n 4. Show average mark\n>>> ";

std::cin >> user\_choice;

switch (user\_choice) {

case 0:

std::cout << "Good buy !!!\n";

break;

case 1:

up:;

std::cout << "Enter file name to read from : ";

std::cin >> file\_name;

if (!gradebook.config(file\_name)) {

goto up;

}

std::cout << "Data was read successfully\n";

break;

case 2:

gradebook.show\_all();

break;

case 3:

std::cout << "Enter mark to show by it : ";

std::cin >> value;

gradebook.show\_by\_marks(value);

break;

case 4:

std::cout << "Average mark : " << average(gradebook) << std::endl;

break;

default:

std::cout << "!!! Wrong number !!!\n";

}

if (user\_choice != 0) {

goto ups;

}

**6. Розписую додаткові функції :**

void copy(char \* text\_1, char \* text\_2) {

int i = 0;

for ( ; text\_1[i] != '\0'; i++) {

text\_2[i] = text\_1[i];

}

for ( ; text\_2[i] != '\0'; i++) {

text\_2[i] = '\0';

}

}

int lenght(char \* text) {

int i = 0;

for ( ; text[i] != '\0'; i++) {}

return i;

}

**Лістинг коду :**

/\*

Загальні умови:

1. Створити клас-нащадок класу розробленого в попередній лабораторній роботі.

Відповідно до варіанту добавити у клас-нащадок відповідні функції-члени, дані-члени та

дружні функції.

2. Написати програму, яка буде демонструвати роботу з функціями-членами класу

та дружньою функцією. Для демонстрації роботи програма повинна містити меню. В

програмі використати один екземпляр класу для кожного конструктора.

Варіант 1. Клас «Залікова книжка»

Створити константну функцію-член, яка буде виводити на екран інформацію по

дисциплінам з балом вищим чим вказаний в аргументі функції.

Створити дружню функцію, яка буде рахувати і повертати середній бал по всім

дисциплінам.

\*/

#include<iostream>

#include<cstdlib>

#include<cstring>

void copy(char \* text\_1, char \* text\_2);

int lenght(char \* text);

struct Discipline {

char \* discipline;

int mark;

char \* teacher\_last\_name;

const char \* faculty = "FKSA";

void show\_faculty() {

std::cout << "Faculty : " << faculty << std::endl;

}

Discipline() {

discipline = NULL;

mark = 0;

teacher\_last\_name = NULL;

}

Discipline(char \* discipline, int mark, char \* teacher\_last\_name) {

//std::cout << "Create an object !!!\n";

this->discipline = new char[lenght(discipline)];

this->teacher\_last\_name = new char[lenght(teacher\_last\_name)];

copy(discipline, this->discipline);

copy(teacher\_last\_name, this->teacher\_last\_name);

this->mark = mark;

}

};

class student\_gradebook {

public:

char \* last\_name;

char \* gradebook;

int subjects\_number;

Discipline \* discipline;

void config(char \* last\_name, char \* gradebook) {

//std::cout << "Create an object !!!\n";

this->last\_name = new char[lenght(last\_name)];

this->gradebook = new char[lenght(gradebook)];

copy(last\_name, this->last\_name);

copy(gradebook, this->gradebook);

}

bool config(char \* file\_name) {

char char\_temp[100];

int int\_temp;

FILE \* file = fopen(file\_name, "r");

if (!file){

std::cout << "!!! Wrong file name !!!\n";

return false;

}

fscanf(file, "%i\n", &subjects\_number);

fscanf(file, "%s\n", char\_temp);

last\_name = new char[lenght(char\_temp)];

copy(char\_temp, this->last\_name);

fscanf(file, "%s\n", char\_temp);

gradebook = new char[lenght(char\_temp)];

copy(char\_temp, this->gradebook);

this->discipline = new Discipline[subjects\_number];

// std::cout << "\nSubjects\_number : " << subjects\_number << "\nStudent's last name : " << last\_name << "\nStudent's gradebook : " << gradebook << "\n\n";

for (int i = 0; i < subjects\_number; i++) {

fscanf(file, "%s\n", char\_temp);

// std::cout << "Subject : " << char\_temp << '\n';

discipline[i].discipline = new char[strlen(char\_temp)];

strcpy(discipline[i].discipline, char\_temp);

fscanf(file, "%i\n", &int\_temp);

// std::cout << "Mark : " << int\_temp << '\n';

discipline[i].mark = int\_temp;

fscanf(file, "%s\n", char\_temp);

// std::cout << "Teacher : " << char\_temp << "\n\n";

discipline[i].teacher\_last\_name = new char[strlen(char\_temp)];

strcpy(discipline[i].teacher\_last\_name, char\_temp);

}

fclose(file);

return true;

}

};

class child\_class : public student\_gradebook {

public:

void show\_by\_marks(int mark) {

std::cout << "Student\'s last name : " << last\_name << "\nGradebook number : " << gradebook << std::endl << '\n';

std::cout << "Mark height than " << mark << " is :\n\n";

for (int i = 0; i < subjects\_number; i++) {

if (discipline[i].mark > mark) {

std::cout << "Subject : " << discipline[i].discipline << '\n';

std::cout << "Mark : " << discipline[i].mark << '\n';

std::cout << "Teacher : " << discipline[i].teacher\_last\_name << "\n\n";

}

}

}

void show\_all() {

std::cout << "Student\'s last name : " << last\_name << "\nGradebook number : " << gradebook << std::endl << '\n';

for (int i = 0; i < subjects\_number; i++) {

std::cout << "Subject : " << discipline[i].discipline << '\n';

std::cout << "Mark : " << discipline[i].mark << '\n';

std::cout << "Teacher : " << discipline[i].teacher\_last\_name << "\n\n";

}

}

float average(child\_class &);

};

float average(child\_class &child) {

float average = 0;

for (int i = 0; i < child.subjects\_number; i++) {

average += child.discipline[i].mark;

}

return average / child.subjects\_number;

}

int main() {

// student\_gradebook gradebook("Ivanov V.V.", "123456");

// student\_gradebook gradebook("zk.txt");

// Discipline discipline("Phisics", 5, "Ivanov");

// child\_class children;

child\_class gradebook;

// gradebook.config("zk.txt");

int user\_choice;

char file\_name[20];

int value;

ups:;

std::cout << "Menu :\n 0. Exit\n 1. Read file\n 2. Show all\n 3. Show by mark\n 4. Show average mark\n>>> ";

std::cin >> user\_choice;

switch (user\_choice) {

case 0:

std::cout << "Good buy !!!\n";

break;

case 1:

up:;

std::cout << "Enter file name to read from : ";

std::cin >> file\_name;

if (!gradebook.config(file\_name)) {

goto up;

}

std::cout << "Data was read successfully\n";

break;

case 2:

gradebook.show\_all();

break;

case 3:

std::cout << "Enter mark to show by it : ";

std::cin >> value;

gradebook.show\_by\_marks(value);

break;

case 4:

std::cout << "Average mark : " << average(gradebook) << std::endl;

break;

default:

std::cout << "!!! Wrong number !!!\n";

}

if (user\_choice != 0) {

goto ups;

}

}

void copy(char \* text\_1, char \* text\_2) {

int i = 0;

for ( ; text\_1[i] != '\0'; i++) {

text\_2[i] = text\_1[i];

}

for ( ; text\_2[i] != '\0'; i++) {

text\_2[i] = '\0';

}

}

int lenght(char \* text) {

int i = 0;

for ( ; text[i] != '\0'; i++) {}

return i;

}

**Висновок :** виконуючи дану лабораторно роботу, я навчився працювати з класами, нащадками та дружніми функціями.