Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Об’єктно – орієнтоване програмування

ЗВІТ ДО

ЛАБОРАТОРНОЇ РОБОТИ №3

ВАРІАНТ №7

Оцінка «\_\_\_\_\_\_\_»

Дата «\_\_» \_\_\_\_\_\_\_ 2017р

Виконав студент ІІ курсу

Гр. ТІ-62

Заїчко Олексій Павлович

Перевірив:\_\_\_\_\_\_\_\_\_\_\_\_

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

Київ 2017

***Склад програми:***

/\*

\* Made by Aleksis\_MVP.

\* All rights reserved 2017 ©

\*/

#include "stdafx.h"

#include "string.h"

#include "windows.h"

class StaffSchedule {

char\* name;

int staffQuantity;

int phd;

int candidates;

int stateWorkers;

int contributors;

void setAmount();

public:

/\* Default constructor\*/

StaffSchedule();

/\*Initialization constructor\*/

StaffSchedule(char\* n, int phd = 0, int candidates = 0, int stateWorkers = 0, int contributors = 0);

/\*Copy constructor\*/

StaffSchedule(const StaffSchedule &p);

/\*Destructor\*/

~StaffSchedule();

/\*Set name method\*/

void setName(const char\* s);

/\*Check and set name\*/

void checkAndSetName(const char\* s);

/\*Sed phd's amount method\*/

void setPhd(int val);

/\*Check and set phd\*/

void checkAndSetPhd(int val);

/\*Set candidate's amount method\*/

void setCandidates(int val);

/\*Check and set candidates\*/

void checkAndSetCandidates(int val);

/\*Set state workers amount method\*/

void setStateWorkers(int val);

/\*Check and set state workers\*/

void checkAndSetStateWorkers(int val);

/\*Set contributor amount method\*/

void setContributors(int val);

/\*Check and set contributors amount\*/

void checkAndSetContributors(int val);

/\*Get name method\*/

char\* getName() { return name; }

/\*Get amount method\*/

int getAmount() { return staffQuantity; }

/\*Get phd's method\*/

int getPhd() { return phd; }

/\*Get candidates method\*/

int getCandidates() { return candidates; }

/\*Get state workers method\*/

int getStateWorkers() { return stateWorkers; }

/\*Get contributors method\*/

int getContributors() { return contributors; }

bool operator> (const StaffSchedule& p);

const StaffSchedule& operator+ (const StaffSchedule& p);

StaffSchedule& operator= (const StaffSchedule& p);

const StaffSchedule& operator++ ();

operator int ();

};

StaffSchedule::StaffSchedule() {

name = new char[12];

strcpy(name, "DefaultName");

staffQuantity = 0;

phd = 0;

candidates = 0;

stateWorkers = 0;

contributors = 0;

printf("Default constructor has been called\n\n");

}

StaffSchedule::StaffSchedule(char\* n, int p, int c, int s, int co)

{

phd = (p >= 0 && p <= 80) ? p : 0;

candidates = (c >= 0 && c <= 80) ? c : 0;

stateWorkers = (s >= 0 && s <= 80) ? s : 0;

contributors = (co >= 0 && co <= 80) ? co : 0;

setAmount();

if (strlen(n) <= 20) {

name = new char[strlen(n) + 1];

strcpy(name, n);

}

else {

name = new char[12];

strcpy(name, "DefaultName");

}

printf("The %s object have been instantiated\n\n", name);

}

StaffSchedule::StaffSchedule(const StaffSchedule &p) {

staffQuantity = p.staffQuantity;

phd = p.phd;

candidates = p.candidates;

stateWorkers = p.stateWorkers;

contributors = p.contributors;

name = new char[strlen(p.name) + 1];

strcpy(name, p.name);

//printf("Copy constructor has been called\n\n");

}

StaffSchedule::~StaffSchedule() {

//printf("Destroying %s object...\n\n", name);

//if (name)

delete[] name;

//system("pause");

}

void StaffSchedule::setAmount() {

staffQuantity = phd + candidates + stateWorkers + contributors;

}

void StaffSchedule::setName(const char\* s) {

if (name) delete[] name;

name = new char[strlen(s) + 1];

strcpy(name, s);

}

void StaffSchedule::checkAndSetName(const char\* s) {

if (name == NULL) return;

if (strlen(s) <= 20) {

setName(s);

}

else {

printf("Entered name is too big.\n\n");

}

}

void StaffSchedule::setCandidates(int val) {

candidates = val;

setAmount();

}

void StaffSchedule::checkAndSetCandidates(int val) {

if (val >= 0 && val <= 80) {

setCandidates(val);

}

else {

printf("Invalid assigmant has been intercepted\n\n");

}

}

void StaffSchedule::setPhd(int val) {

phd = val;

setAmount();

}

void StaffSchedule::checkAndSetPhd(int val) {

if (val >= 0 && val <= 80) {

setPhd(val);

}

else {

printf("Invalid assigmant has been intercepted\n\n");

}

}

void StaffSchedule::setStateWorkers(int val) {

stateWorkers = val;

setAmount();

}

void StaffSchedule::checkAndSetStateWorkers(int val) {

if (val >= 0 && val <= 80) {

setStateWorkers(val);

}

else {

printf("Invalid assigmant has been intercepted\n\n");

}

}

void StaffSchedule::setContributors(int val) {

contributors = val;

setAmount();

}

void StaffSchedule::checkAndSetContributors(int val) {

if (val >= 0 && val <= 80) {

setContributors(val);

}

else {

printf("Invalid assigmant has been intercepted\n\n");

}

}

bool StaffSchedule::operator>(const StaffSchedule& p) {

printf("\"More\" operator called\n\n");

return (staffQuantity > p.staffQuantity);

}

const StaffSchedule& StaffSchedule::operator+ (const StaffSchedule& p) {

StaffSchedule result;

result.phd = (phd + p.phd > 80) ? 80 : (phd + p.phd);

result.candidates = (candidates + p.candidates > 80) ? 80 : (candidates + p.candidates);

result.stateWorkers = (stateWorkers + p.stateWorkers > 80) ? 80 : (stateWorkers + p.stateWorkers);

result.contributors = (contributors + p.contributors > 80) ? 80 : (contributors + p.contributors);

result.setAmount();

if (strlen(name) + strlen(p.name) < 20) {

result.name = new char[strlen(name) + strlen(p.name) + 2];

strcpy(result.name, name);

strcat(result.name, "&");

strcat(result.name, p.name);

}

else {

result.name = new char[4];

result.name[0] = toupper((name)[0]);

result.name[1] = '&';

result.name[2] = toupper((p.name)[0]);

result.name[3] = '\0';

}

printf("Adding operator called\n\n");

return result;

}

StaffSchedule& StaffSchedule::operator= (const StaffSchedule& p) {

if (this == &p) {

return \*this;

}

staffQuantity = p.staffQuantity;

phd = p.phd;

candidates = p.candidates;

stateWorkers = p.stateWorkers;

contributors = p.contributors;

delete[] this->name;

this->name = new char[strlen(p.name) + 1];

strcpy(this->name, p.name);

printf("Assignment operator called\n\n");

return \*this;

}

const StaffSchedule& StaffSchedule::operator++ () {

if (contributors < 80) {

++contributors;

}

else if (stateWorkers < 80) {

++stateWorkers;

}

else if (candidates < 80) {

++candidates;

}

else if (phd < 80) {

++phd;

}

setAmount();

printf("Preincrement operator called\n\n");

return \*this;

}

StaffSchedule::operator int() {

printf("To int operator called\n\n");

return staffQuantity;

}

void instructions();

int readNumber();

char\* readString();

void printTable(int argAmount, const StaffSchedule ...);

inline void printDivider() {

putchar(195);

for (int i = 0; i < 72; i++) {

putchar(45);

if (i == 19 || i == 33 || i == 36 || i == 46 || i == 59) {

putchar(197);

}

}

putchar(180);

putchar('\n');

}

StaffSchedule global("OOO Horns and Hoofs", 1, 3, 10, 5);

int main()

{

StaffSchedule a, copy(global);

StaffSchedule local("State", 10, 6, 50, 24);

if (local > global) {

printf("%s object is greater than %s object\n\n", local.getName(), global.getName());

}

else {

printf("%s object is greater than %s object\n\n", global.getName(), local.getName());

}

++local;

printTable(1, local);

printf("%s object to int value: %d \n\n", local.getName(), (int)local);

int choice;

StaffSchedule test;

test = global;

do {

instructions();

do {

printf("?");

choice = readNumber();

if (choice < 1 || choice > 7) printf("Invalid input! Try again\n? ");

} while (choice < 1 || choice > 7);

switch (choice) {

case 1:

printf("Enter data\n? ");

test.checkAndSetName(readString());

break;

case 2:

printf("Enter data\n? ");

test.checkAndSetPhd(readNumber());

break;

case 3:

printf("Enter data\n? ");

test.checkAndSetCandidates(readNumber());

break;

case 4:

printf("Enter data\n? ");

test.checkAndSetStateWorkers(readNumber());

break;

case 5:

printf("Enter data\n? ");

test.checkAndSetContributors(readNumber());

break;

case 6:

printTable(1, test);

break;

case 7: break;

default:

printf("Invalid option has been chosen!\n\n");

}

} while (choice != 7);

printTable(5, global, a, copy, local, test);

system("pause");

return 0;

}

void instructions() {

printf("\*\*\*MENU\*\*\*\n");

printf("%s\n%s\n%s\n%s\n%s\n%s\n%s\n\n",

"1 - Give your object better name",

"2 - Set doctors amount",

"3 - Set candidates amount",

"4 - Set state workers amount",

"5 - Set contributors amount",

"6 - Print table",

"7 - Exit program");

}

int readNumber() {

int number;

char buffer[256];

do {

scanf("%256s", buffer);

number = atoi(buffer);

if (!number) printf("%s%s", "Invalid input! Try Again:\n", "Enter data\n? ");

} while (!number);

return number;

}

char\* readString() {

char buffer[256];

scanf("%s", buffer);

char\* val = new char[strlen(buffer) + 1];

strcpy(val, buffer);

return val;

}

void printTable(int argAmount, const StaffSchedule ...) {

va\_list arguments;

va\_start(arguments, argAmount);

/\*Head of table\*/

putchar(218);

for (int i = 0; i < 72; i++) {

putchar(45);

if (i == 19 || i == 33 || i == 36 || i == 46 || i == 59) {

putchar(194);

}

}

putchar(191);

putchar('\n');

printf("%c%-20s%c%s%c%s%c%s%c%s%c%s%c\n", 179,

"Object name", 179, "Staff Quantity", 179, "PHD", 179, "Candidates",

179, "State workers", 179, "Contributors", 179);

/\*End head of table\*/

/\*Start body of table\*/

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

StaffSchedule& parametr = va\_arg(arguments, StaffSchedule);

printDivider();

printf("%c%-20s%c%14d%c%3d%c%10d%c%13d%c%12d%c\n", 179,

parametr.getName(), 179, parametr.getAmount(), 179, parametr.getPhd(), 179, parametr.getCandidates(),

179, parametr.getStateWorkers(), 179, parametr.getContributors(), 179);

}

putchar(192);

for (int i = 0; i < 72; i++) {

putchar(45);

if (i == 19 || i == 33 || i == 36 || i == 46 || i == 59) {

putchar(193);

}

}

putchar(217);

putchar('\n');

va\_end(arguments);

}

*Максимальна довжина імені:* 20 символів.

*Максимальна кількість одного виду робітників:* 80.

***Відповіді на теоритичні питання:***

*Властивості полів класу:* Зазвичай представляють дані про об’єкт, можуть бути публічними і приватними, а також захищеними.

*Вибір конструктора компілятором:* Залежно від виду об’яви і переданих даних про новий об’єкт, компілятор вибирає конструктор за замовчуванням, ініціалізації і копіювання.

*Чи можна перевантажити оператор для double?* Перевантажувати операції для вбудованих типів даних заборонено. Можливо перевантажувати операції, якщо один або більше аргументів об’єкт класу користувача.