Государственный Университет Молдовы
Факультет Математики и Информатики
Департамент Информатики

Лабораторная работа №1

по курсу “Объектно ориентированное программирование”
тема: Классы

Выполнил студент группы I2302:
Ciobanu
Stanislav.,
Проверил преподаватель:
G.Latul

Кишинэу, 2023

Условия работы

Вариант 1 : Дан класс Рабочий с приватными (private) полями: Имя, дата, дата приема на работу, должность, зарплата и публичные поля: конструктор без параметров, конструктор с параметрами, деструктор и вспомогательные функции. Создать программу, в кот. Объявляется массив объектов этого класса и меню (каждый пункт меню – отдельная функция):

1) Ввести элементы массива с клавиатуры;

2) Вывести на экран элементы массива (объекты);

3) Сортировать в алфавитном порядке по полю Имя;

4) Вывести на экран элементы, для кот. зарплата меньше чем значение X (вводится с клавиатуры);

5) Добавить на позицию К новый элемент (К вводится с клавиатуры);

6) Удалить первый элемент, для которого мы знаем должность (вводим должность и по должности удаляем первого попавшегося);

7) Записать элементы массива в файл;

8) Вывести на экран первые N элементов из файла (N с клавиатуры).

Задние 0

Создаём класс Worker в отдельном header файле Worker.h. Объявляем приватные поля класса – Имя, дата (_birthDate), дата приёма на работу (_hiring date), зарплата (_salary). Т.к нам необходимо создать поле “должность”, создаём enum (JobTitle), где определяем разные должности. После этого создаём приватное поле _job типа JobTitle.

```
enum JobTitle
{
	Programmer,
	Tester,
	HR,
	TeamLead,
	GameDesigner,
	Other
};

class Worker
{
private:

	// Fields

	string _name;
	tm _birthDate;
	tm _hiringDate;
	JobTitle _job;
	int _salary;
}
```

Создаём конструкторы, деструкторы, геттеры и сеттеры.

```
// Constructors

Worker();

Worker(string name, tm birthDate, tm hiringDate, JobTitle job, int salary);

Worker(string str);

// Getters

string GetName();
tm GetBirthDate();
tm GetHiringDate();
JobTitle GetJobTitle();
int GetSalary();

// Setters

void SetSalary(int salary);
void SetName(string name);
void SetHiringDate(tm date);
void SetBirthDate(tm date);
void SetJobTitle(JobTitle job);

// Destructor

~Worker();
```

Имплементируем методы в новосозданном файле (Worker.cpp).

Задние 1

В основном cpp файле программы создадим функцию, где попросим пользователя ввести все данные, а после создадим экземпляр Worker с этими полями и вернём его.

```
Worker* EnterWorkerData()
{
	string name;
	cout << "Enter name" << endl;
	cin >> name;

	. . .

	return  new Worker(name, birthDate, hiringDate, (JobTitle)j, salary);

}
```

Задние 2

Создадим функцию (PrintWorkerData), которая будет выводить на экран данные об одном рабочем. Далее создадим функцию (PrintData), которая выведет данные обо всех пользователях из массива.

```
void PrintWorkerData(Worker* worker)
{
	. . .
}


void PrintData(Worker** workers)
{

	int size = _msize(workers) / sizeof(workers[0]);
	for (int i = 0; i < size; i++)
	{
		PrintWorkerData(workers[i]);
	}

}
```

Задние 3

Создадим функцию (SortByName), которая будет сортировать массив рабочих. Для этого подключаем библиотеку vector. Создаём вектор, в который копируем имена рабочих и сортируем их функцией sort. После этого выводим имена на экран.

```
#include <vector>

. . .

void SortByName(Worker** workers)
{
	int size = _msize(workers) / sizeof(workers[0]);

	vector<string> names;

	for (int i = 0; i < size; i++)
	{
		names.push_back(workers[i]->GetName());
	}

	sort(names.begin(), names.end());

	for (int i = 0; i < size; i++)
	{
		cout << names[i] << endl;
	}

}
```

Задние 4

Создадим функцию (SortByName). В ней попросим пользователя ввести зарплату в цикле найдём всех рабочих, у которых зарплата <= введённой и выведем информацию о них на экран.

```
void FindBySalary(Worker** workers)
{

	. . .

	cin >> sal;

	for (int i = 0; i < size; i++)
	{
		if (workers[i]->GetSalary() >= sal)
		{
			cout << . . .;
		}
	}

	. . .
}
```

Задние 5

Создадим функцию (AddNewWorker), которая будет возвращать новый массив, с добавленным внутрь рабочим, введённым с клавиатуры.

```
Worker** AddNewWorker(Worker** workers, int position)
{
	. . .

	if (position < size && position >= 0)
	{

		Worker** workers2 = new Worker * [size + 1];

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

			workers2[i] = workers[i];

		}

		workers2[position] = EnterWorkerData();

		if (position < size - 1)
		{
			for (int i = position + 1; i <= size; i++)
			{
				workers2[i] = workers[i - 1];
			}
		}

		return workers2;
	}
	else
	{
		return workers;
	}
}
```

Задние 6

Создадим функцию (DeleteElement), в которой попросим пользователя ввести индекс должности, по которому ищем и удаляем первого попавшегося рабочего. Записываем в новый массив и возвращаем его.

```
Worker** DeleteElement(Worker** workers)
{
	cout << endl << " Enter Job title" << endl;
	cout << "   0 - Programmer" << endl;

	. . .

	cout << "   5 - Other" << endl;

	. . .

	int size = _msize(workers) / sizeof(workers[0]);

	Worker** workers2 = new Worker * [size - 1];

	int j = 0;
	for (int i = 0; i < size; i++)
	{
		if (workers[i]->GetJobTitle() == job && i == j)
		{
			delete workers[i];
			i++;
		}

		workers2[j] = workers[i];

		j++;
	}

	. . .

	return workers2;

}
```

Задние 7

Создадим функцию (SaveToFile), в которую передадим массив рабочих и путь к файлу. Открываем поток вывода, в цикле выводим необходимые поля каждого из рабочих в файл и закрываем поток вывода.

```
void SaveToFile(string fileName, Worker** workers)
{
	ofstream outFile;
	outFile.open(fileName);

	if (outFile.is_open())
	{

		. . .

		for (int i = 0; i < size; i++)
		{
			outFile << workers[i]->GetName() << " " << . . .
 << " " << workers[i]->GetSalary() << endl;
		}

		cout << "File has been written" << std::endl;
	}
	else
	{
		cout << "File hasn't been written" << std::endl;
	}

	outFile.close();
	cout << endl << "Press any key to continue";
}
```

Задние 8

Создадим функцию (GetDataFromFile), в которую передадим и путь к файлу. Открываем поток ввода и записываем всё в новосозданный массив рабочих. Возвращаем массив.

```
Worker** GetDataFromFile(string filename)
{
	ifstream inFile(filename);

	if (inFile.is_open())
	{

		string str;

		int i = 0;
		while (getline(inFile, str))
		{
			i++;
		}

		. . .

		while (getline(inFile, str))
		{
			workers[i] = new Worker(str);
			i++;
		}

		inFile.close();

		return workers;
	}
	else
	{
		cerr << "Unable to open the file.\n";
		return NULL;
	}
}
```

Используемые библиотеки

```
<iostream>   // Console
<string>     // String
"Worker.h"
<conio.h>    // getch
<time.h>     // time
<stdio.h>    // msize
<algorithm>  // sort
<vector>
<fstream>    // working with files
```