МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №2**

по дисциплине: «Объектно-ориентированные технологии

программирования и стандарты проектирования»

на тему: «Конструкторы и деструкторы»

Вариант №8 «Отдел кадров»

|  |  |
| --- | --- |
| Выполнил | Студент группы 21-ИТ-1  Макеёнок Д.И. |
|  |  |
| Проверил | Ассистент  Хирьянов И.Д. |

Полоцк 2023

**Цель работы:** научиться описывать конструкторы и деструкторы для класса, понимать различия между конструкторами. Понимать, в каких случаях могут быть вызваны конструкторы копирования, понимать время жизни объекта.

**Ход работы**

**Задание:**

1. Дополнить и при необходимости модифицировать приложение,
2. разработанное согласно варианта лабораторной работы №1.
3. Определить в классах следующие конструкторы: без параметров, с параметрами, копирования, перемещения.
4. Определить в классе деструктор.
5. В каждом конструкторе и деструкторе выдавать сообщение, показывающее, какой именно конструктор или деструктор был вызван.
6. Дополнить основной класс методом с параметрами, внутри которого должен создаваться объект дополнительного класса с помощью конструктора с параметрами и добавляться в массив объектов дополнительного класса.
7. Дополнить основной класс методом с двумя параметрами (ссылка на объект дополнительного класса и количество), внутри которого должны создаваться объекты дополнительного класса с помощью конструктора копирования (создается необходимое количество копий указанного объекта) и заноситься в массив объектов дополнительного класса.
8. Дополнить/модифицировать основное тело программы таким образом, чтобы продемонстрировать использование всех конструкторов.
9. Сделать выводы.

**Вариант 8**

Тема проекта: приложение «Отдел кадров».

Спроектировать ПО, предназначенное для создания и просмотра информации

по штатным работникам предприятия. Функции, которые должны быть

реализованы в приложении: добавление, удаление, редактирование и

просмотр работников, структуризация работников по отделам, детальная

информация о работнике.

Листинг 1.1 - реализация меню программы:

namespace lab1;

class Program

{

static void Main()

{

int dep = 0;

bool flag;

Department department = new Department();

Human Eployment = new Human();

while (true)

{

Console.WriteLine("Что делаем?\n" +

"1 - Добавить сотрудника\n" +

"2 - Удалить сотрудника\n" +

"3 - Просмотреть сотрудников\n" +

"4 - Редактировать сотрудника\n" +

"5 - Добавить сотрудника через конструктор с параметрами\n" +

"6 - Копировать сотрудника");

int.TryParse(Console.ReadLine(), out int n);

switch (n)

{

case 1:

Console.WriteLine("В какой отдел добавить сотрудника?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out dep) && dep <= 3 && dep >= 1))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

Eployment.NewEployment();

department.AddNewEployee(dep, Eployment);

Console.WriteLine("Успешно");

// Console.ReadKey();

Console.Clear();

break;

}

break;

case 2:

Console.WriteLine("Из какого отдела удалить сотрудника?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out dep) && dep <= 3 && dep >= 1))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

flag = department.DeleteEployee(dep);

if (flag)

{

Console.WriteLine("Успешно");

break;

}

else

{

break;

}

}

Console.ReadKey();

Console.Clear();

break;

case 3:

Console.WriteLine("Какой отдел вывести?(1-3) / all - вывести всех");

while (true)

{

string tem = Console.ReadLine();

if (int.TryParse(tem, out dep))

{

if (!(dep <= 3 && dep >= 1))

{

Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

department.ShowEployee(dep);

}

else

{

if (tem != "all")

{

Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

continue;

}

department.ShowEployee();

break;

}

}

//Console.ReadKey();

//Console.Clear();

break;

case 4:

Console.WriteLine("Из какого отдела редактируемый сотрудник?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out dep) && dep <= 3 && dep >= 1))

{

Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

continue ;

}

dep--;

flag = department.EditingEployee(dep);

if (flag)

{

Console.WriteLine("Успешно");

break;

}

else

{

break;

}

}

Console.ReadKey();

Console.Clear();

break;

case 5:

Console.WriteLine("В какой отдел добавить сотрудника?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out dep) && dep <= 3 && dep >= 1))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

department.AddEmployeeByConstructor(dep);

Console.WriteLine("Успешно");

// Console.ReadKey();

Console.Clear();

break;

}

break;

case 6:

Console.WriteLine("Из какого отдела копируемый сотрудник?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && dep <= 3 && dep >= 1)

{

dep--;

if (department.CheckEmployee(dep))

{

Console.WriteLine("Введите номер копируемого сотрудника ");

if (int.TryParse(Console.ReadLine(), out int number)) { }

else break;

Console.WriteLine("Введите количество копий ");

if (!int.TryParse(Console.ReadLine(), out int quantity))

break;

department.Copy(dep, number, quantity);

Console.WriteLine("Успешно");

Console.ReadKey();

Console.Clear();

break;

}

else

{

Console.WriteLine("В этом отделе никого нет.");

break;

}

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

}

}

}

}

Листинг 1.2 - реализация главного класса:

using System.Data;

using System.Runtime.Intrinsics.Arm;

using System.Security.Cryptography.X509Certificates;

using System.Xml.Linq;

namespace lab1

{

class Department

{

List<Human>[] depart = new List<Human>[3];

public void AddNewEployee(int num, Human hum)

{

depart[num].Add(new Human(hum));

}

public Department()

{

depart[0] = new List<Human>();

depart[1] = new List<Human>();

depart[2] = new List<Human>();

}

public bool DeleteEployee(int num)

{

int count = 0;

string name = "";

bool flag = false, MoreOne = false;

if (depart[num].Count == 0) {

Console.WriteLine("В этом отделе никого нет.");

return false;

}

Console.WriteLine("Фамилия удаляемого сотрудника:");

name = Console.ReadLine();

for (int i = 0; i < depart[num].Count; i++)

{

if (depart[num][i].GetSurename().IndexOf(name) != -1)

{

Console.Write(i + ".");

depart[num][i].Show();

count = i;

if (flag)

{

MoreOne = true;

}

flag = true;

}

}

if (MoreOne)

{

Console.Write("Сотрудника под каким номером нужно удалить?\n");

while (true)

{

if (int.TryParse(Console.ReadLine(), out int i))

{

depart[num].RemoveAt(i);

GC.Collect();

return true;

}

else Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

}

}

else

{

if (flag)

{

depart[num].RemoveAt(count);

GC.Collect();

return true;

}

else {

Console.WriteLine("Неверная фамилия, или такой нет");

return false;

}

}

}

public void ShowEployee()

{

int num = 0;

for (int i = 0; i < 3; i++)

{

Console.WriteLine(++num + "-------------------------------");

for (int j = 0; j < depart[i].Count; j++)

{

Console.WriteLine(j + ".");

depart[i][j].Show();

}

if (depart[i].Count == 0) Console.WriteLine("Тут пусто\n");

}

}

public void ShowEployee(int num)

{

for (int i = 0; i < depart[num].Count; i++)

{

Console.WriteLine(i + ".");

depart[num][i].Show();

}

if (depart[num].Count == 0) Console.WriteLine("Тут пусто\n");

}

public bool EditingEployee(int num)

{

int count = 0;

string name = "";

bool flag = false, MoreOne = false;

if (depart[num].Count == 0)

{

Console.WriteLine("В этом отделе никого нет.");

return false;

}

Console.Write("Фамилия редактируемого сотрудника:");

name = Console.ReadLine();

for (int i = 0; i < depart[num].Count; i++)

{

if (depart[num][i].GetSurename() == name)

{

if (depart[num][i].GetSurename().IndexOf(name) != -1)

{

Console.Write(i + ".");

depart[num][i].Show();

count = i;

if (flag)

{

MoreOne = true;

}

flag = true;

}

}

}

if (MoreOne)

{

Console.Write("Сотрудника под каким номером нужно редактировать?\n");

while (true)

{

if (int.TryParse(Console.ReadLine(), out int i))

{

count = i;

break;

}

else Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

}

}

else

{

if (!flag)

{

Console.WriteLine("Неверная фамилия, или такой нет");

return false;

}

}

Console.WriteLine("Что нужно изменить?\n" +

"1 - Имя сотрудника\n" +

"2 - Фамилию сотрудника\n" +

"3 - День рождения сотрудника\n" +

"4 - Год начала работы сотрудника\n");

int edit = Convert.ToInt32(Console.ReadLine());

switch (edit)

{

case 1:

depart[num][count].AddHumanName();

return true;

case 2:

depart[num][count].AddHumanSurename();

return true;

case 3:

depart[num][count].AddHumanBirthday();

return true;

case 4:

depart[num][count].AddHumanStartYear();

return true;

}

return false;

}

public void AddEmployeeByConstructor(int dep)

{

int startyear = 0, day = 0, year = 0, month = 0;

string name = "", surename = "";

DateTime birthday;

bool flag = true;

Console.Write("Введите имя сотрудника: ");

while (true)

{

name = Console.ReadLine();

flag = true;

foreach (char c in name)

{

if (!((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= 'а' && c <= 'я') || (c >= 'А' && c <= 'Я')))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

flag = false;

break;

}

}

if (flag) break;

}

Console.Write("Введите фамилию сотрудника: ");

while (true)

{

surename = Console.ReadLine();

flag = true;

foreach (char c in surename)

{

if (!((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= 'а' && c <= 'я') || (c >= 'А' && c <= 'Я')))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

flag = false;

break;

}

}

if (flag) break;

}

Console.WriteLine("Ввод Данных дня рождения сотрудника");

Console.Write("Введите день(1 - 31): ");

while (true)

{

if (int.TryParse(Console.ReadLine(), out day) && day <= 12 && day > 0)

{

break;

}

else

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

}

Console.Write("Введите месяц(1 - 12): ");

while (true)

{

if (int.TryParse(Console.ReadLine(), out month) && day <= 31 && day > 0)

{

break;

}

else

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

}

Console.Write($"Введите год({DateTime.Now.Year - 60} - {DateTime.Now.Year - 18}): ");

while (true)

{

if (int.TryParse(Console.ReadLine(), out year) && year <= DateTime.Now.Year - 18 && year >= DateTime.Now.Year - 60)

{

break;

}

else

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

}

birthday = new DateTime(year, month, day);

Console.Write("Введите год начала работы(2000 - 2023): ");

while (true)

{

if (int.TryParse(Console.ReadLine(), out startyear) && startyear <= 2023 && startyear >= 2000 && (startyear - 18) >= year)

{

break;

}

else

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

}

Human hum = new Human(startyear, name, birthday,surename, day, month, year);

depart[dep].Add(hum);

}

public void Copy(int dep, int number, int quantity)

{

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

{

Human human = new Human(depart[dep][number]);

depart[dep].Add(human);

}

}

public bool CheckEmployee(int dep)

{

if (depart[dep].Count > 0)

{

return true;

}

else return false;

}

}

}

Листинг 1.3 – реализация дополнительного класса

using System.Linq.Expressions;

using System.Security.Cryptography;

namespace lab1

{

class Human

{

private string \_name;

private string \_surename;

private DateTime \_birthday;

private int \_startYear;

public Human()

{

\_startYear = 0;

\_name = "пусто";

\_surename = "пусто";

\_birthday = DateTime.Now;

Console.WriteLine("Вызван конструкотр без параметров");

}

public Human(int startYear,string name, DateTime birthday, string surename,int day,int month,int year)

{

\_startYear = startYear;

\_name = name;

\_surename = surename;

\_birthday = birthday;

Console.WriteLine("Вызван конструкотр с параметрами");

}

public Human(Human human)

{

\_startYear = human.\_startYear;

\_name = human.\_name;

\_surename = human.\_surename;

\_birthday = human.\_birthday;

Console.WriteLine("Вызван конструкотр копирования");

}

~Human()

{

Console.WriteLine("Вызван деструктор");

}

public string GetName()

{

return \_name;

}

public DateTime Getbirthday()

{

return \_birthday;

}

public int GetStartYear()

{

return \_startYear;

}

public string GetSurename()

{

return \_surename;

}

public void AddHumanName()

{

Console.Write("Введите имя сотрудника: ");

while (true) {

bool flag = true;

string str = Console.ReadLine();

foreach (char c in str)

{

if (!((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= 'а' && c <= 'я') || (c >= 'А' && c <= 'Я')))

{

flag = false;

Console.Write("Некоректный ввод, попробуй ещё раз: ");

break;

}

}

if (flag)

{

\_name = str;

break;

}

}

}

public void AddHumanSurename()

{

Console.Write("Введите фамилию сотрудника: ");

while (true)

{

bool flag = true;

string str = Console.ReadLine();

foreach (char c in str)

{

if (!((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= 'а' && c <= 'я') || (c >= 'А' && c <= 'Я')))

{

flag = false;

Console.Write("Некоректный ввод, попробуй ещё раз: ");

break;

}

}

if (flag)

{

\_surename = str;

break;

}

}

}

public void AddHumanBirthday()

{

int day, month, year;

Console.WriteLine("Ввод Данных дня рождения сотрудника\nВведите день(1 - 31): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out day) && day <= 31 && day > 0;

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

break;

}

Console.Write("Введите месяц(1 - 12): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out month) && month <= 12 && month > 0;

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

break;

}

Console.Write($"Введите год({DateTime.Now.Year - 60} - {DateTime.Now.Year - 18}): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out year) && year <= DateTime.Now.Year - 18 && year >= DateTime.Now.Year - 60;

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

break;

}

\_birthday = new DateTime(year, month, day);

}

public void AddHumanStartYear()

{

Console.Write($"Введите год начала работы(2000 - {DateTime.Now.Year}): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out int year) && year <= DateTime.Now.Year && year >= 2000 && (year - 18) >= \_birthday.Year;

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

\_startYear = year;

break;

}

}

public void NewEployment()

{

AddHumanName();

AddHumanSurename();

AddHumanBirthday();

AddHumanStartYear();

}

public void Show()

{

Console.Write(

"Имя и Фамилия - " + \_name + " " + \_surename + "\n"+

$"День рождения - {\_birthday.ToString("D")}\n"+

"Год начала работы - " + \_startYear + "\n\n");

}

}

}

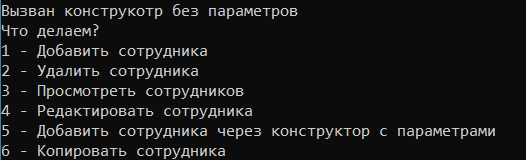


Рисунок 1. Начальное меню и конструктор без параметров

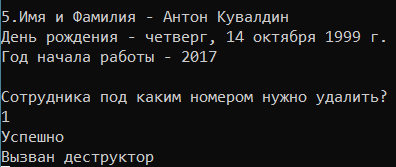


Рисунок 2. Вызван деструктор

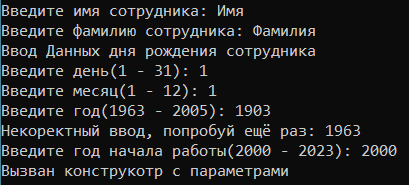


Рисунок 3. Вызван конструктор с параметрами

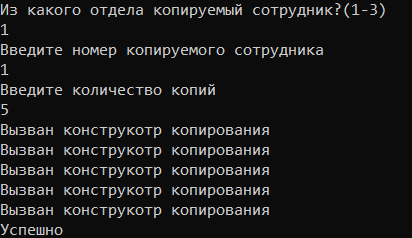


Рисунок 4. Вызван конструктор копирования

**Вывод:** В ходе данный лабораторной работе было изучено как описывать конструкторы и деструкторы для класса, понимать различия между конструкторами. Понимать, в каких случаях могут быть вызваны конструкторы копирования, понимать время жизни объекта.