**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №1

«Разработка класса для базы данных»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-22 |  | преподаватель каф. ИУ5 |
| Барышников Михаил |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

**Задача**

1. Разработать класс «Планета» для планет солнечной системы. Класс должен иметь 4 характеристики:
   1. Имя планеты.
   2. Диаметр планеты.
   3. Количество спутников.
   4. Наличие жизни.
2. Создать методы (get, set) соответственно для доступа к этим параметрам и изменения их изменения, также проверять правильность ввода данных.
3. Создать 3 конструктора:
   1. По умолчанию
   2. Параметрический
   3. Копирования
4. Организовать интерфейс пользователя с программой в виде меню, позволяющего выполнять следующие действия:

- сортировка БД;

- редактирование БД;

- вывод БД на экран.

1. Сортировка должна быть реализована:
   1. По именам
   2. По размеру диаметра
   3. По количеству спутников

Радиуса <= 0 не может быть – введены некорректные данные

Спутников < 0 не может быть – введены некорректные данные

Одинакового имени у планет не может быть – введены некорректные данные

**Текст программы**

* class\_planet.h

#pragma once

#include <string>

class Planet

{

std::string name;

int diam;

int count\_of\_satellites;

bool is\_life;

public:

Planet() {}

Planet(const std::string&, const int&, const int&, const bool&);

Planet(const Planet&);

~Planet() {}

void set\_name(const std::string&);

void set\_diam(const int&);

void set\_count\_of\_satellites(const int&);

void set\_is\_life(const bool&);

std::string get\_name();

int get\_diam();

int get\_count\_of\_satellites();

bool get\_is\_life();

friend std::ostream &operator<<(std::ostream&, const Planet&);

static bool check\_name\_planet(std::string n, Planet \*pl, int count);

static void sorting(Planet \*pl, int choise, int count);

static void swap(Planet&, Planet&);

};

* class\_planet.cpp

#include "class\_planet.h"

Planet::Planet(const std::string& n, const int& d, const int& count\_sat, const bool& is\_l)

{

name = n;

set\_diam(d);

set\_count\_of\_satellites(count\_sat);

is\_life = is\_l;

}

Planet::Planet(const Planet& pl)

{

name = pl.name;

diam = pl.diam;

count\_of\_satellites = pl.count\_of\_satellites;

is\_life = pl.is\_life;

}

void Planet::set\_name(const std::string &n)

{

name = n;

}

void Planet::set\_diam(const int &d)

{

diam = d > 0 ? d : throw std::exception("Invalid diameter!");

}

void Planet::set\_count\_of\_satellites(const int &count)

{

count\_of\_satellites = count >= 0 ? count : throw std::exception("Invalid count!");

}

void Planet::set\_is\_life(const bool &is)

{

is\_life = is;

}

std::string Planet::get\_name()

{

return name;

}

int Planet::get\_diam()

{

return diam;

}

int Planet::get\_count\_of\_satellites()

{

return count\_of\_satellites;

}

bool Planet::get\_is\_life()

{

return is\_life;

}

std::ostream& operator<<(std::ostream &stream, const Planet &pl)

{

stream << "Name of the planet: " << pl.name << "\n";

stream << "Diameter of the planet: " << pl.diam << "\n";

stream << "Count of satellites: " << pl.count\_of\_satellites << "\n";

stream << "Life on the planet: " << (pl.is\_life ? "exist" : "not exist") << "\n";

return stream;

}

bool Planet::check\_name\_planet(std::string n, Planet \*pl, int count)

{

for (int i = 0; i != count; ++i)

if (n == pl[i].get\_name())

return false;

return true;

}

void Planet::sorting(Planet \*pl, int choise, int count)

{

switch (choise)

{

case 1:

for (int i = 0; i != count; ++i)

for (int j = i; j != count; ++j)

if (pl[i].get\_name() > pl[j].get\_name())

Planet::swap(pl[i], pl[j]);

break;

case 2:

for (int i = 0; i != count; ++i)

for (int j = i; j != count; ++j)

if (pl[i].get\_diam() < pl[j].get\_diam())

Planet::swap(pl[i], pl[j]);

break;

case 3:

for (int i = 0; i != count; ++i)

for (int j = i; j != count; ++j)

if (pl[i].get\_count\_of\_satellites() < pl[j].get\_count\_of\_satellites())

Planet::swap(pl[i], pl[j]);

break;

}

}

void Planet::swap(Planet &a, Planet &b)

{

Planet c = a;

a = b;

b = c;

}

* main.cpp

#include <iostream>

#include "class\_planet.h"

Planet set\_planet(Planet \*pl, int count)

{

using std::string;

using std::cout;

using std::cin;

string name;

int diam, count\_sat, is;

bool exist = false;

while (!exist)

{

std::cout << "\nEnter name: ";

std::cin >> name;

exist = Planet::check\_name\_planet(name, pl, count);

if (!exist)

cout << "\nError! This name is already is used!\n";

}

std::cout << "\nEnter diameter: ";

std::cin >> diam;

cout << "\nEnter count of satellites: ";

cin >> count\_sat;

cout << "\nEnter life factor, 1 - exist, 0 - not exist: ";

cin >> is;

return Planet(name, diam, count\_sat, is);

}

int main()

{

using namespace std;

char what\_to\_do = 1;

int count;

cout << "Count of planets: ";

cin >> count;

Planet \*mas = new Planet[count];

for (int i = 0; i != count; ++i)

try

{

mas[i] = set\_planet(mas, count);

}

catch (exception &ex)

{

--i;

cout << "\n" << ex.what() << " Try again!";

}

while (what\_to\_do != '0')

{

try

{

cout << "\nEnter:\n\t";

cout << "0 - exit\n\t";

cout << "1 - change all params\n\t";

cout << "2 - change specific param\n\t";

cout << "3 - show all information about planets\n\t";

cout << "4 - sorting planets\n";

cin >> what\_to\_do;

switch (what\_to\_do)

{

case '1':

{

int index;

cout << "\n\tEnter number of planet: ";

cin >> index;

--index;

if (index < 0 || index >= count)

throw exception("Invalid number of planet!");

else

mas[index] = set\_planet(mas, count);

}

break;

case '2':

{

int index;

cout << "\n\tEnter number of planet: ";

cin >> index;

--index;

if (index < 0 || index >= count)

{

throw exception("Invalid number of planet!");

}

else

{

int choise;

cout << "\n\t1 - change name: ";

cout << "\n\t2 - change diameter: ";

cout << "\n\t3 - change count of satellites: ";

cout << "\n\t4 - change - life factor, 1 - exist, 0 - not exist: \n";

cin >> choise;

switch (choise)

{

case 1:

{

string new\_name;

bool exist = false;

while (!exist)

{

std::cout << "\nEnter name: ";

std::cin >> new\_name;

exist = Planet::check\_name\_planet(new\_name, mas, count);

if (!exist)

cout << "\nError! This name is already is used!\n";

}

mas[index].set\_name(new\_name);

}

break;

case 2:

{

int diam;

cout << "\nEnter diameter: ";

cin >> diam;

mas[index].set\_diam(diam);

}

break;

case 3:

{

int count\_s;

cout << "\nEnter count of satellites: ";

cin >> count\_s;

mas[index].set\_count\_of\_satellites(count\_s);

}

break;

case 4:

{

bool is;

cout << "\nEnter life factor, 1 - exist, 0 - not exist: ";

cin >> is;

mas[index].set\_is\_life(is);

}

break;

}

}

}

break;

case '3':

{

for (int i = 0; i != count; ++i)

{

cout << "\nPlanet number is " << i + 1 << "\n";

cout << mas[i];

}

}

break;

case '4':

{

int choise;

cout << "\n\tEnter: ";

cout << "\n\t1 - sorting name: ";

cout << "\n\t2 - sorting diameter: ";

cout << "\n\t3 - sorting count of satellites: \n";

cin >> choise;

Planet::sorting(mas, choise, count);

}

break;

}

}

catch (exception &ex)

{

cout << "\n" << ex.what();

}

}

system("pause");

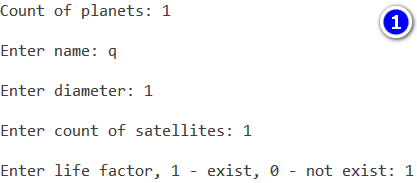
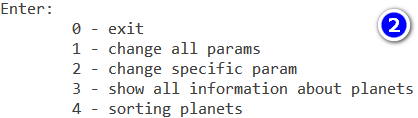
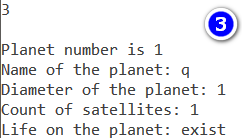
return 0;

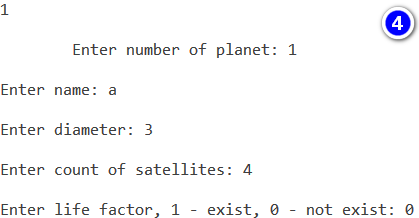
}

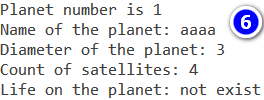
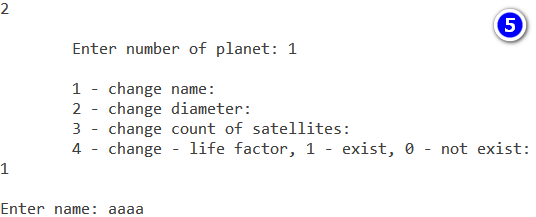
**Анализ результатов**

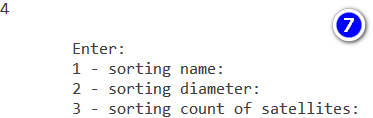
Важное замечание, нумеровать объекты не требовалось по задания, поэтому номера планет не фиксированы, а созданы только для взаимодействия с пользователем!

Интерфейс программы:







Тетировалась написанная программа для работы с БД на следующих влодных данных:

Планета №1

1. Имя – aaab
2. Диаметр – 100
3. Количество спутников – 10
4. Фактор жизни – нет

Планета №2

1. Имя – aaac
2. Диаметр – 110
3. Количество спутников – 5
4. Фактор жизни – есть

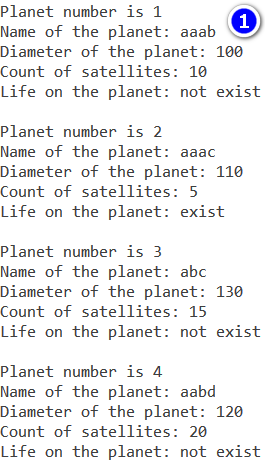
Планета №3

1. Имя – abc
2. Диаметр – 130
3. Количество спутников – 15
4. Фактор жизни – нет

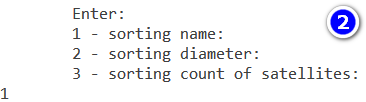
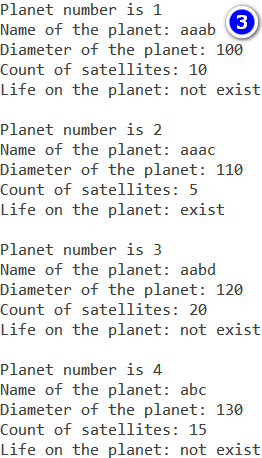
Планета №4

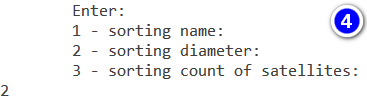
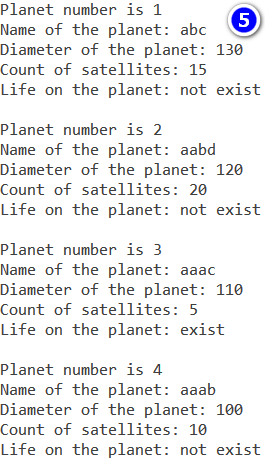
1. Имя – aabd
2. Диаметр – 120
3. Количество спутников – 20
4. Фактор жизни – нет

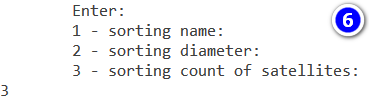
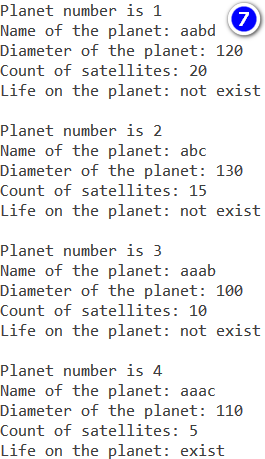
Введенные данные:



Полученные результаты сортировки:

Программа сортирует по убыванию. Результаты сортировки верны.

**Вывод**

В результате написания программы был получен опыт работы с классами (ООП). Стало понятно зачем необходим конструктор и деструктор. Были созданы методы класса для доступа к переменным. Также программа способна обрабатывать исключения, которые выбрасывают методы класса, например, если ввести радиус планеты <= 0, будет выброшено исключение. Еще был переопределен оператор вывода <<, для этого создана friend – функция класса.

Впервые были использованы статические функции класса.

Получен опыт разработки интерфейса для взаимодействия пользователя с БД.