МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Вятский государственный университет»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

ЛАБОРАТОРНАЯ РАБОТА №6

«Разработка приложение на основе ОО парадигмы»

Отчёт по лабораторной работе дисциплины

«Технологии программирования»

Выполнил студент группы ИВТб-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Жеребцов К. А./

Проверил преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М. Л./

Киров 2022

**Задание:** в выбранной предметной области создать иерархию классов, состоящую минимум из одного родительского и двух дочерних классов. В каждом классе определить минимум два собственных член данных, две собственных, две унаследованных и две перекрытых член функции. Разработать приложение, демонстрирующее принципы полиморфизма, наследования и инкапсуляции.

**Экранные формы:**

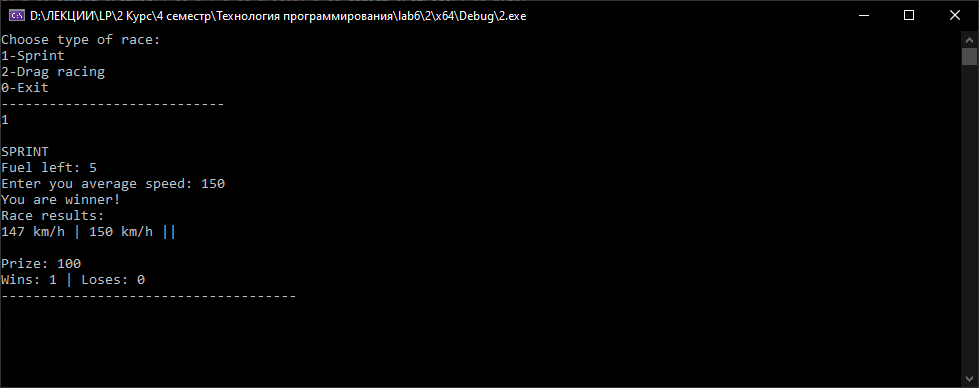


Рисунок 1 – Результат выполнения программы.



Рисунок 2 – Результат выполнения программы.

**Листинг:**

#include <iostream>

#include <ctime>

#include <Windows.h>

#include <stdio.h>

#include <conio.h>

using namespace std;

class Race

{

private:

int gas = 0;

int prize = 0;

protected:

int wins = 0;

int loses = 0;

public:

//полиморфизм перекрытая #1

virtual void Info()

{

cout << "Choose type of race: \n";

cout << "1-Sprint\n";

cout << "2-Drag racing\n";

cout << "0-Exit\n";

cout << "----------------------------\n";

}

//полиморфизм перекрытая #2

virtual int CheckEnter()

{

int number;

cin >> number;

while ((!cin.good()) || (cin.peek() != '\n') || (number != 1) && (number != 2) && (number != 0))

{

cin.clear();

cin.ignore(255, '\n');

cin >> number;

}

return number;

}

//собственная #1 нследуемая #1

void SetGas(int GasValue)

{

gas = GasValue;

}

//собственная #2 нследуемая #2

int GetGas()

{

return gas;

}

int getPrize()

{

return prize;

}

void setPrize(int a)

{

prize = a;

}

int GetWin()

{

return wins;

}

int GetLose()

{

return loses;

}

};

class Sprint : public Race

{

private:

int count = 1;

int results[10][2];

public:

/\*

~Sprint()

{

count = 1;

delete results;

}\*/

//Собственная #1

void ShowStat(int op, int you)

{

cout << "Race results: \n";

results[count - 1][0] = op;

results[count - 1][1] = you;

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

{

cout << results[i][0] << " km/h | " << results[i][1] << " km/h || \n";

}

count++;

}

//Собственная #2

int RaceResult(int num)

{

int pr = getPrize();

int OpponentRes = num + rand() % 41 - 20;

if (num > OpponentRes)

{

cout << "You are winner!\n";

pr = pr + 100;

setPrize(pr);

wins++;

}

else

{

cout << "You are loser!\n";

if (pr >= 100)

{

pr = pr - 100;

setPrize(pr);

}

loses++;

}

return OpponentRes;

}

//Перекрытая #1

void Info()

{

cout << "Sprint complete!\n";

}

//Перекрытая #2

int CheckEnter()

{

int number;

cout << "Enter you average speed: ";

cin >> number;

while ((!cin.good()) || (cin.peek() != '\n') || (number <100) || (number > 300))

{

cin.clear();

cin.ignore(255, '\n');

cout << "Wrong input!\n";

cin >> number;

}

return number;

}

};

class Drag : public Race

{

private:

int count = 1;

int results[10][2];

public:

//Собственная #1

void ShowStat(int op, int you)

{

cout << "Race results: \n";

results[count - 1][0] = op;

results[count - 1][1] = you;

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

{

cout << results[i][0] << " s | " << results[i][1] << " s || \n";

}

count++;

}

//Собственная #2

int RaceResult(int num)

{

int pr = getPrize();

int OpponentRes = num + rand() % 6 - 5;

if (num < OpponentRes)

{

cout << "You are winner!\n";

pr = pr + 100;

setPrize(pr);

wins++;

}

else

{

cout << "You are loser!\n";

if (pr >= 100)

{

pr = pr - 100;

setPrize(pr);

}

loses++;

}

return OpponentRes;

}

//Перекрытая #1

void Info()

{

cout << "Drag complete!\n";

}

//Перекрытая #2

int CheckEnter()//полиморфизм

{

int number;

cout << "Enter you time: ";

cin >> number;

while ((!cin.good()) || (cin.peek() != '\n') || (number < 8) || (number > 25))

{

cin.clear();

cin.ignore(255, '\n');

cout << "Wrong input!\n";

cin >> number;

}

return number;

}

};

int main()

{

srand(time(NULL));

char c;

cout << "Enter the amount of fuel (from 1 to 10): ";

int Fuel;

cin >> Fuel;

while ((!cin.good()) || (cin.peek() != '\n') || (Fuel <= 0) || (Fuel > 10))

{

cin.clear();

cin.ignore(255, '\n');

cout << "Wrong input!\n";

cin >> Fuel;

}

Race\* Racing = new Race();

Racing->SetGas(Fuel);

int key;

do

{

system("cls");

Racing->Info();

key = Racing->CheckEnter();

switch (key)

{

case 1:

{

cout << "\nSPRINT\n";

Race\* sprint = new Sprint();

do

{

int youSpeed;

cout << "Fuel left: " << Fuel << '\n';

youSpeed = sprint->CheckEnter();

int opSpeed = static\_cast<Sprint\*>(sprint)->RaceResult(youSpeed);

static\_cast<Sprint\*>(sprint)->ShowStat(opSpeed, youSpeed);

Fuel--;

cout << "\nPrize: " << sprint->getPrize() << "\n";

cout << "Wins: " << sprint->GetWin() << " | " << "Loses: " << sprint->GetLose() << "\n";

cout << "-------------------------------------\n";

c = \_getch();

} while (Fuel > 0);

system("cls");

cout << "Total prize: " << sprint->getPrize() << "\n";

cout << "Wins: " << sprint->GetWin() << " | " << "Loses: " << sprint->GetLose() << "\n";

delete sprint;

key = 0;

}

break;

case 2:

{

cout << "\nDRAG\n";

Race\* drag = new Drag();

do

{

int youTime;

cout << "Fuel left: " << Fuel << '\n';

youTime = drag->CheckEnter();

int opTime = static\_cast<Drag\*>(drag)->RaceResult(youTime);

static\_cast<Drag\*>(drag)->ShowStat(opTime, youTime);

Fuel--;

cout << "\nPrize: " << drag->getPrize() << "\n";

cout << "Wins: " << drag->GetWin() << " | " << "Loses: " << drag->GetLose() << "\n";

cout << "-------------------------------------\n";

c = \_getch();

} while (Fuel > 0);

system("cls");

cout << "Total prize: " << drag->getPrize() << "\n";

cout << "Wins: " << drag->GetWin() << " | " << "Loses: " << drag->GetLose() << "\n";

delete drag;

key = 0;

}

break;

}

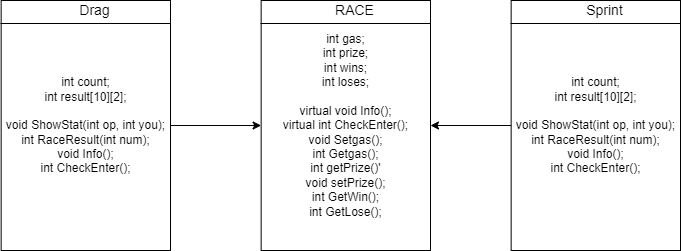
} while (key != 0);

delete Racing;

return 0;

}

**Иерархия классов:**

****

**Вывод:**

В ходе выполнения лабораторной работы была написана программа-игра, в которой можно выбрать тип гонки и совершить серию заездов. Для реализации необходимого функционала были разработаны три класса: один родительский и два дочерних. Родительский класс имеет методы, часть которых наследуется в дочерних, а часть перекрывается. Для того чтобы функция считалась перекрытой, необходимо указать ее как виртуальную. Перекрытые функции показывают принципы полиморфизма, а наследуемые функции – принципы наследования. Размещение член данных классов в разделе private запрещает изменять их вне класса без специальных методов: сеттеров и геттеров. Эти методы класса позволяют получить доступ к скрытым член данным или изменять их вне этого класса. Данные функции демонстрируют принцип инкапсуляции.