



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)

Факультет

О

Естественнонаучный

шифр

наименование

Кафедра

О7

Информационные системы и программная инженерия

шифр

наименование

Дисциплина

Информационные технологии и программирование

ПРАКТИЧЕСКАЯ РАБОТА №2

Классы

Вариант №6

Выполнил студент группы

О722Б

Вяткин Н.А.

Фамилия И.О.

ПРЕПОДАВАТЕЛЬ

Гладевич А.А.

Фамилия И.О.

Подпись

« 20 »

марта

2023 г.

Цель работы

Изучить одну из базовых концепций ООП — наследование классов с++, заключающуюся в построении цепочек классов, связанных иерархически. Получить практические навыки работы с наследованием и виртуальными функциями.

Постановка задачи

В рамках работы необходимо разработать иерархию классов на указанную в варианте тему. В иерархии должен быть корневой класс (базовый), который должен быть абстрактным и содержать общие для остальных поля и методы. У каждого из классов должен быть хотя бы один собственный метод. Также должны быть продемонстрированы виртуальные и переопределённые методы. Переопределённые методы должны также вызывать методы базового класса если это оправдано.

Вариант 6

Вариант 6: Наземная техника.

Варианты полей: марка, объём двигателя, максимальная скорость, тип кузова.

Варианты методов: завести, поехать на определённое расстояние, заправить.

Возможные классы иерархии: наземная техника (базовый), автомобиль, танк, электросамокат.

Текст программы

Файл main.cpp:

```
#include <iostream>
#include "class.h"
#include <vector>
#define n 10

using namespace std;

int main(void)
{
    int menu, variant, i = 0, k, count; // переменные для меню, i
    индекс для заполнения массива, count количество совпадений, k индекс для
    вывода и поисков в массиве
    string yes; // переменная для возврата
    в меню
    groundEquip **arr; // массив указателей на
    объекты базового класса
    arr = new groundEquip *[n]; // выделяем память под
    массив
    *arr = NULL;
    do
    {
        system("cls");
        cout << "1. Add ground equipment" << endl
        << "2. Show all objects" << endl
        << "3. Select and execute a specific class method for
all objects" << endl
        << "4. Select type of objects and execute his special
method" << endl
        << "5. Exit" << endl;
        cin >> menu;
        switch (menu)
        {
            case 1: // добавление объекта в массив
            {
                if (i == n-1) // если массив полон
                {
                    cout << "Array is full" << endl;
                    cout << "For return to the menu, press e or E" <<
endl;

                    do
                    {
                        cin >> yes;
                    } while (yes != "e" && yes != "E");

                    break;
                }
                do
                {
                    if (i == n-1) // если массив полон
                    {
                        cout << "Array is full" << endl;
                        cout << "For return to the menu, press e or E" <<
endl;

                        do
                        {

```

```

        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}

system("cls"); // меню выбора для пользователя
cout << "1. Add auto" << endl
    << "2. Add tank" << endl
    << "3. Add electric scooter" << endl
    << "4. Return to the menu" << endl;
cin >> variant;
switch (variant)
{
case 1:
{
    system("cls");
    cout << "Input information about auto" << endl;
    arr[i] = new Auto();
    cin >> *static_cast<Auto *>(arr[i++]);
    cout << "For return to the menu, press e or E"
    << endl;

    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
case 2:
{
    system("cls");
    cout << "Input information about tank" << endl;
    arr[i] = new Tank();
    cin >> *static_cast<Tank *>(arr[i++]);
    cout << "For return to the menu, press e or E"
    << endl;

    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
case 3:
{
    system("cls");
    cout << "Input information about electric
    scooter" << endl;

    arr[i] = new electricScooter();
    cin >> *static_cast<electricScooter
    *>(arr[i++]);

    cout << "For return to the menu, press e or E"
    << endl;

    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
}
}

```

```

        } while (variant != 4);
        break;
    }
    case 2: // вывод всех объектов массива
    {
        system("cls");
        if (i == 0) // если пуст
        {
            cout << "The array is still empty, add objects" <<
endl;
            cout << "For return to the menu, press e or E" <<
endl;

            do
            {
                cin >> yes;
            } while (yes != "e" && yes != "E");
            break;
        }
        cout << "Outputing information about objects" << endl;
        for (k = 0; k < i; k++)
            arr[k]->printInformation();
        cout << "For return to the menu, press e or E" << endl;
        do
        {
            cin >> yes;
        } while (yes != "e" && yes != "E");

        break;
    }
    case 3: // выполнения общих методов, пользователь выбирает
какой общий метод выполнить
    {
        system("cls"); // если пуст
        if (i == 0)
        {
            cout << "The array is still empty, add objects" <<
endl;
            cout << "For return to the menu, press e or E" <<
endl;

            do
            {
                cin >> yes;
            } while (yes != "e" && yes != "E");
            break;
        }
        do
        {
            system("cls"); // пользователь выбирает какой
общий метод выполнить
            cout << "1. Execute the start method for all
objects" << endl
                << "2. Execute the drive distance method for
all objects" << endl
                << "3. Execute the fill method for all
objects" << endl
                << "4. Return to the menu" << endl;
            cin >> variant;

```

```

switch (variant)
{
case 1:
{
    system("cls");
    for (k = 0; k < i; k++)
    {
        cout << arr[k]->GetMake() << endl;
        arr[k]->toStart();
    }
    cout << "For return to the menu, press e or E"
<< endl;

    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
case 2:
{
    system("cls");
    for (k = 0; k < i; k++)
    {
        cout << arr[k]->GetMake() << endl;
        arr[k]->toRide();
    }
    cout << "For return to the menu, press e or E"
<< endl;

    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
case 3:
{
    system("cls");
    for (k = 0; k < i; k++)
    {
        cout << arr[k]->GetMake() << endl;
        arr[k]->toFill();
    }
    cout << "For return to the menu, press e or E"
<< endl;

    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
}
} while (variant != 4);
break;
}
case 4:
{
    system("cls");

```

```

        if (i == 0) // если пуст
        {
            cout << "The array is still empty, add objects" <<
endl;
            cout << "For return to the menu, press e or E" <<
endl;
            do
            {
                cin >> yes;
            } while (yes != "e" && yes != "E");
            break;
        }
        do
        {
            count = 0;
            system("cls"); // пользователь выбирает для какого
            класса наземной техники выполнить его конкретный специальный метод
            cout << "1. Execute the violate traffic method for
all objects type Auto(violate traffic rules)" << endl
                << "2. Execute the shoot method for all
objects type Tank(shoot)" << endl
                << "3. Execute the crash method for all
objects type electricScooter(crash)" << endl
                << "4. Return to the menu" << endl;
            cin >> variant;
            switch (variant)
            {
            case 1:
            {
                system("cls");
                for (k = 0; k < i; k++)
                {
                    if (typeid(*arr[k]) == typeid(Auto)) //
поиск данного класса
                    {
                        count++;
                        cout << arr[k]->GetMake() << endl;
                        static_cast<Auto*>(arr[k])-
>breakRule();
                    }
                }
                if (count == 0)
                    cout << "Objects of this class not found"
<< endl;
                cout << "For return to the menu, press e or E"
<< endl;
                do
                {
                    cin >> yes;
                } while (yes != "e" && yes != "E");
                break;
            }
            case 2:
            {
                system("cls");
                for (k = 0; k < i; k++)
                {

```

```

        if (typeid(*arr[k]) == typeid(Tank)) //
поиск данного класса
        {
            count++;
            cout << arr[k]->GetMake() << endl;
            static_cast<Tank
*>(arr[k])-
>toShoot();
        }
    }
    if (count == 0)
        cout << "Objects of this class not found"
<< endl;
    cout << "For return to the menu, press e or E"
<< endl;
    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
case 3:
{
    system("cls");
    for (k = 0; k < i; k++)
    {
        if (typeid(*arr[k]) ==
typeid(electricScooter)) // поиск данного класса
        {
            count++;
            cout << arr[k]->GetMake() << endl;
            static_cast<electricScooter
*>(arr[k])->toCrash();
        }
    }
    if (count == 0)
        cout << "Objects of this class not found"
<< endl;
    cout << "For return to the menu, press e or E"
<< endl;
    do
    {
        cin >> yes;
    } while (yes != "e" && yes != "E");
    break;
}
}
} while (variant != 4);
break;
}
}
} while (menu != 5);
delete[] arr;
return 0;
}

```

Файл class.h:

```

#ifndef CLASS_H
#define CLASS_H

```



```

#include "iostream"

using namespace std;
// базовый класс
class groundEquip
{
    string make;           // марка
    double engineCapacity; // объем двигателя
    double maxSpeed;       // максимальная скорость

public:
    groundEquip();          // конструктор без
параметров
    groundEquip(string, double, double); // конструктор с
параметрами для базового класс
    ~groundEquip();        // деструктор

    string GetMake();       // получить марку
    double GetCapacity();   // получить объем двигателя
    double GetSpeed();      // получить скорость

    virtual void printName(); // вывести имя класса
    virtual void printInformation(); // вывести всю информацию об
объекте

    virtual void toStart() = 0; // метод завести
    virtual void toRide() = 0;  // метод поехать
    virtual void toFill() = 0;  // метод заправить

    friend istream &operator>>(istream &, groundEquip &); //
перегрузка оператора ввода
};

class Auto : public groundEquip // класс наследник
{
    string typeBodie; // тип кузова

public:
    Auto();          // конструктор
    Auto(string, double, double, string); // конструктор с
параметрами
    ~Auto();         // деструктор

    void printName() override; // вывод имени класса
    void printInformation() override; // вывод информации об
объекте этого класса

    void toStart() override;
// завести
    void toRide() override;
// поехать
    void toFill() override;
// заправить
    void breakRule() { cout << "The driver violated the traffic
rules" << endl; }; // нарушить правила

```

```

        friend istream &operator>>(istream &, Auto &); // перегрузка
оператора ввода
    };

    class Tank : public groundEquip // класс наследник танк
    {
        string typeTank; // тип танка

    public:
        Tank(); // конструктор без
параметров
        Tank(string, double, double, string); // конструктор с
параметрами
        ~Tank(); // деструктор

        void printName() override; // вывод имени класса
        void printInformation() override; // вывод всей информации об
танке

        void toStart() override; //
завести
        void toRide() override; //
проехать
        void toFill() override; //
заправить
        void toShoot() { cout << "Tank is shooting" << endl; }; //
стрелять

        friend istream &operator>>(istream &, Tank &); // перегрузка
оператора ввода
    };

    class electricScooter : public groundEquip // класс наследник
электросамокат
    {
        double sizeBattery; // размер батареи

    public:
        electricScooter(); // конструктор
        electricScooter(string, double, double, double); //
конструктор с параметрами
        ~electricScooter(); // деструктор

        void printName() override; // вывод имени класса
        void printInformation() override; // вывод всей информации об
электро самокате

        void toStart() override; //
завести
        void toRide() override; //
поехать
        void toFill() override; //
зарядить
        void toCrash() { cout << "A scooter crashed into a man" <<
endl; }; // врезаться

        friend istream &operator>>(istream &, electricScooter &);

```

```
};  
#endif
```

Файл methods.cpp:

```
#include "class.h"  
#include <cmath>  
#include <limits>  
// конструктор без параметров  
groundEquip::groundEquip()  
{  
    cout << "Constructor without parametrs" << endl;  
    make = "No make";  
    engineCapacity = 0;  
    maxSpeed = 0;  
}  
// конструктор с параметрами автомобиля  
Auto::Auto(string a, double b, double c, string d) : groundEquip(a,  
b, c)  
{  
    cout << "Constructor for auto" << endl;  
    typeBodie = d;  
}  
// конструктор с параметрами танка  
Tank::Tank(string a, double b, double c, string d) : groundEquip(a,  
b, c)  
{  
    cout << "Constructor for tank" << endl;  
    typeTank = d;  
}  
// конструктор с параметрами электросамоката  
electricScooter::electricScooter(string a, double b, double c,  
double d) : groundEquip(a, b, c)  
{  
    cout << "Constructor for electric scooter" << endl;  
    sizeBattery = d;  
}  
// конструктор без параметров для автомобиля  
Auto::Auto() : groundEquip()  
{  
    cout << "Constructor for auto" << endl;  
    typeBodie = "No type bodie";  
}  
// конструктор без параметров для танка  
Tank::Tank() : groundEquip()  
{  
    cout << "Constructor for tank" << endl;  
    typeTank = "No type tank";  
}  
// конструктор без параметров для электро самоката  
electricScooter::electricScooter() : groundEquip()  
{  
    cout << "Constructor for electric scooter" << endl;  
    sizeBattery = 1;  
}  
// деструктор  
groundEquip::~~groundEquip()  
{  
    cout << "Destructor" << endl;
```

```

    }
    // деструктор автомобиля
    Auto::~~Auto()
    {
        cout << "Destructor Auto" << endl;
    }
    // деструктор танка
    Tank::~~Tank()
    {
        cout << "Destructor Tank" << endl;
    }
    // деструктор электросамоката
    electricScooter::~electricScooter()
    {
        cout << "Destructor electricScooter" << endl;
    }
    // конструктор с параметрами для базового класса
    groundEquip::groundEquip(string a, double b, double c)
    {
        cout << "Constructor with parametrs" << endl;
        make = a;
        engineCapacity = b;
        maxSpeed = c;
    }
    // получение имен классов и получение значений полей базового
    класса
    string groundEquip::GetMake() { return make; }
    double groundEquip::GetCapacity() { return engineCapacity; }
    double groundEquip::GetSpeed() { return maxSpeed; }
    void groundEquip::printName() { cout << "Ground equip" << endl; }
    void Auto::printName() { cout << "Auto" << endl; }
    void Tank::printName() { cout << "Tank" << endl; }
    void electricScooter::printName() { cout << "Electric scooter" <<
endl; }
    // перегрузка оператора ввода для базового класса
    istream &operator>>(istream &in, groundEquip &machine)
    {
        cout << "Entering values " << endl;
        cout << "Input make" << endl;
        in >> machine.make;
        cout << "Input engine capacity" << endl;
        while (1)
        {
            in >> machine.engineCapacity;
            if (machine.engineCapacity > 0)
                break;
            else
            {
                cout << "Input correct value of engine capacity" <<
endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
            }
        }
        cout << "Input max speed" << endl;
        while (1)
        {

```

```

        in >> machine.maxSpeed;
        if (machine.maxSpeed > 0)
            break;
        else
        {
            cout << "Input correct value of max speed" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }
    }
    return in;
}
// перегрузка оператора ввода для класса авто наследника класса
наземная техника
istream &operator>>(istream &in, Auto &car)
{
    in >> static_cast<groundEquip &>(car);
    cout << "Input type bodie of auto" << endl;
    in >> car.typeBodie;
    return in;
}
// перегрузка оператора присваивания для класса танк наследника
класса наземная техника
istream &operator>>(istream &in, Tank &t)
{
    in >> static_cast<groundEquip &>(t);
    cout << "Input type of tank" << endl;
    in >> t.typeTank;
    return in;
}
// перегрузка оператора присваивания для класса электро самокат
наследника класса наземная техника
istream &operator>>(istream &in, electricScooter &scooter)
{
    in >> static_cast<groundEquip &>(scooter);
    cout << "Input size of battery" << endl;
    while (1)
    {
        in >> scooter.sizeBattery;
        if (scooter.sizeBattery > 0)
            break;
        else
        {
            cout << "Input corect size of battery" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(),
(' \n'));
        }
    }
    return in;
}
// вывод информации для базового класса
void groundEquip ::printInformation()
{
    const type_info &capacity = typeid(GetCapacity());
    const type_info &speed = typeid(GetSpeed());
    cout << "Outputing information about ground equipment" << endl;
}

```

```

        cout << "Type of ground equipment: "
        << "groundEquip;"
        << " Data type of make: "
        << "string"
        << "; Make: " << GetMake()
        << "; Data type of engine capacity : " << capacity.name()
<< "; Engine capacity: " << GetCapacity()
        << " Data type of max speed: " << speed.name() << "; Max
speed: " << GetSpeed() << endl;
    }
    // вывод информации об авто
    void Auto ::printInformation()
    {
        const type_info &capacity = typeid(GetCapacity());
        const type_info &speed = typeid(GetSpeed());
        cout << "Outputing information about auto" << endl;
        cout << "Type of ground equipment: "
        << "Auto;"
        << " Data type of make: "
        << "string;"
        << " Make: " << GetMake()
        << "; Data type of engine capacity : " << capacity.name()
<< "; Engine capacity: " << GetCapacity()
        << "; Data type of max speed: " << speed.name() << "; Max
speed: " << GetSpeed() << " Data type of typeBodie: string; "
        << " Type of Bodie: " << typeBodie << endl;
    }
    // вывод информации об танке
    void Tank ::printInformation()
    {
        const type_info &capacity = typeid(GetCapacity());
        const type_info &speed = typeid(GetSpeed());
        cout << "Outputing information about tank" << endl;
        cout << "Type of ground equipment: "
        << "Tank;"
        << " Data type of make: "
        << "string;"
        << " Make: " << GetMake()
        << "; Data type of engine capacity : " << capacity.name()
<< "; Engine capacity: " << GetCapacity()
        << "; Data type of max speed: " << speed.name() << "; Max
speed: " << GetSpeed() << "Data type of typeTank: string; "
        << "Type of tank: " << typeTank << endl;
    }
    // вывод информации об электросамокате
    void electricScooter ::printInformation()
    {
        cout << "Outputing information about electric scooter" << endl;
        cout << "Type of ground equipment: "
        << "electricScooter"
        << " Data type of make: "
        << "string"
        << " Make: " << GetMake()
        << " Data type of engine capacity : " <<
typeid(GetCapacity()).name() << " Engine capacity: " << GetCapacity()

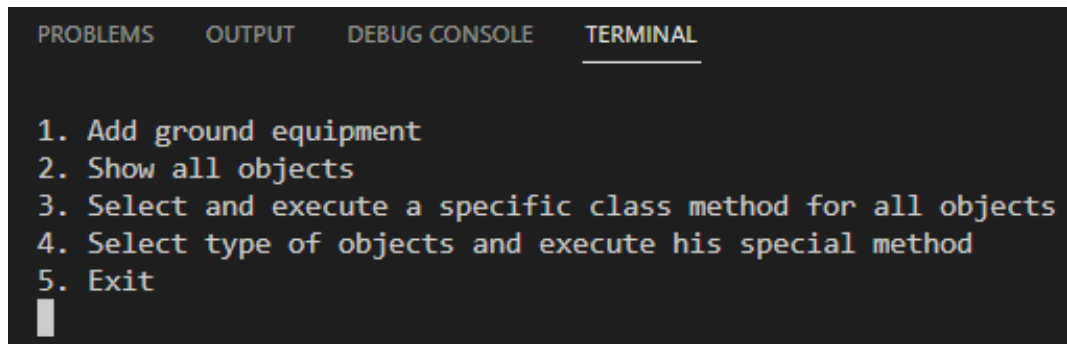
```

```

        << "      Data      type      of      max      speed:      "      <<
typeid(GetSpeed()).name() << " Max speed: " << GetSpeed() << "Data type
of sizeBattery: " << typeid(sizeBattery).name()
        << ";Size battery: " << sizeBattery << endl;
    }
    // завести авто
    void Auto::toStart()
    {
        cout << "The car starts" << endl;
    }
    // завести танк
    void Tank::toStart()
    {
        cout << "The Tank starts" << endl;
    }
    // завести самокат
    void electricScooter::toStart()
    {
        cout << "The electric scooter starts" << endl;
    }
    // поехать на авто
    void Auto::toRide()
    {
        cout << "The car drove 120km " << endl;
    }
    // поехать на танке
    void Tank::toRide()
    {
        cout << "The tank drove 50km " << endl;
    }
    // поехать на электросамокате
    void electricScooter::toRide()
    {
        cout << "The electric scooter drove 20km " << endl;
    }
    // заправить авто
    void Auto::toFill()
    {
        cout << "Petrol is poured into the car" << endl;
    }
    // заправить танк
    void Tank::toFill()
    {
        cout << "Diesel is poured into the tank" << endl;
    }
    // зарядить электросамокат
    void electricScooter::toFill()
    {
        cout << "The scooter is charging" << endl;
    }
}

```

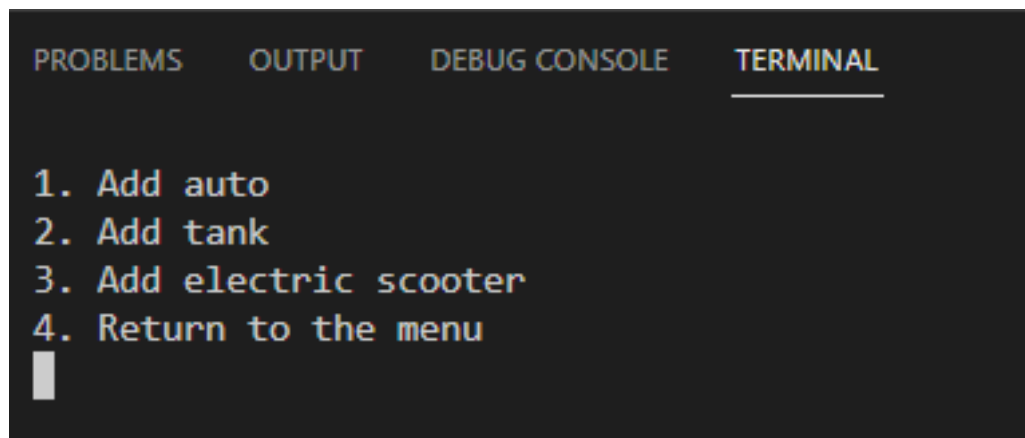
Скриншоты



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

1. Add ground equipment
2. Show all objects
3. Select and execute a specific class method for all objects
4. Select type of objects and execute his special method
5. Exit
█
```

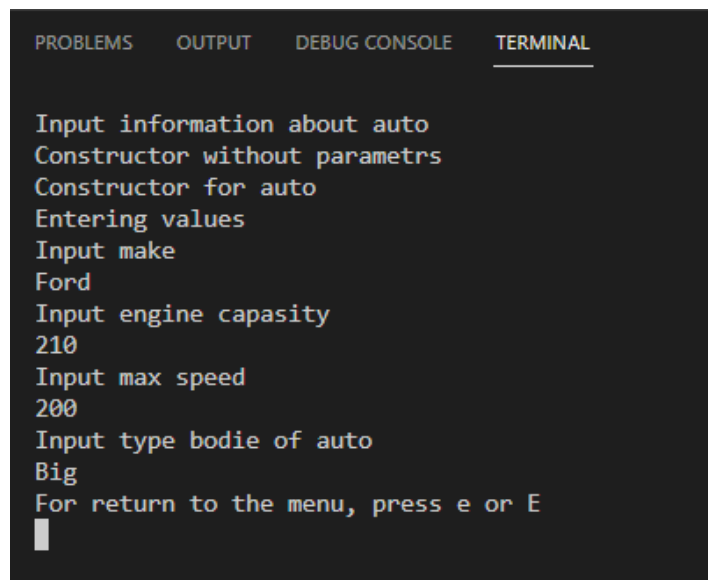
Рисунок 1 – Основное меню



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

1. Add auto
2. Add tank
3. Add electric scooter
4. Return to the menu
█
```

Рисунок 2 – Меню добавления записи в массив



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Input information about auto
Constructor without parametrs
Constructor for auto
Entering values
Input make
Ford
Input engine capacity
210
Input max speed
200
Input type bodie of auto
Big
For return to the menu, press e or E
█
```

Рисунок 3 – Добавление автомобиля в массив


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Input information about tank
Constructor without parametrs
Constructor for tank
Entering values
Input make
Russian
Input engine capacity
450
Input max speed
60
Input type of tank
Light
For return to the menu, press e or E
█
```

Рисунок 4 – Добавление танка в массив

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Input information about electric scooter
Constructor without parametrs
Constructor for electric scooter
Entering values
Input make
Xiaomi
Input engine capacity
120
Input max speed
40
Input size of battery
2000
For return to the menu, press e or E
█
```

Рисунок 5 – Добавление электросамоката в массив

```
Constructor for tank
Outputing information about objects
Outputing information about auto
Type of ground equipment: Auto; Data type of make: string; Make: Ford; Data type of engine capacity :d; Engine capacity: 210; Data ty
pe of max speed: d; Max speed: 200 Data type of typeBodie: string; Type of Bodie: Big
Outputing information about tank
Type of ground equipment: Tank; Data type of make: string; Make: Russian; Data type of engine capacity :d; Engine capacity: 450; Data
type of max speed: d; Max speed: 60Data type of typeTank: string; Type of tank: Light
Outputing information about electric scooter
Type of ground equipment: electricScooter Data type of make: string Make: Xiaomi Data type of engine capacity :d Engine capacity: 120
Data type of max speed: d Max speed: 40Data type of sizeBattery: d;Size battery: 2000
Outputing information about auto
Type of ground equipment: Auto; Data type of make: string; Make: Toyota; Data type of engine capacity :d; Engine capacity: 220; Data
type of max speed: d; Max speed: 200 Data type of typeBodie: string; Type of Bodie: small
For return to the menu, press e or E
█
```

Рисунок 6 – Вывод информации о всех объектах массива

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

1. Execute the start method for all objects
2. Execute the drive distance method for all objects
3. Execute the fill method for all objects
4. Return to the menu
█
```

Рисунок 7 – Меню выбора методов базового класса

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Ford
The car starts
Russian
The Tank starts
Xiaomi
The electric scooter starts
Toyota
The car starts
For return to the menu, press e or E
█
```

Рисунок 8 – Выполнение метода toStart() всеми объектами, которые есть в массиве

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Ford
The car drove 120km
Russian
The tank drove 50km
Xiaomi
The electric scooter drove 20km
Toyota
The car drove 120km
For return to the menu, press e or E
█
```

Рисунок 9 – Выполнение метода toRide() всеми объектами, которые есть в массиве

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ford
Petrol is poured into the car
Russian
Diesel is poured into the tank
Xiaomi
The scooter is charging
Toyota
Petrol is poured into the car
For return to the menu, press e or E
█
```

Рисунок 10 – Выполнение метода toFill() всеми объектами, которые есть в массиве

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1. Execute the violate traffic method for all objects type Auto(violate traffic rules)
2. Execute the shoot method for all objects type Tank(shoot)
3. Execute the crash method for all objects type electricScooter(crash)
4. Return to the menu
█
```

Рисунок 11 – Меню выбора уникального метода для каждого класса

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

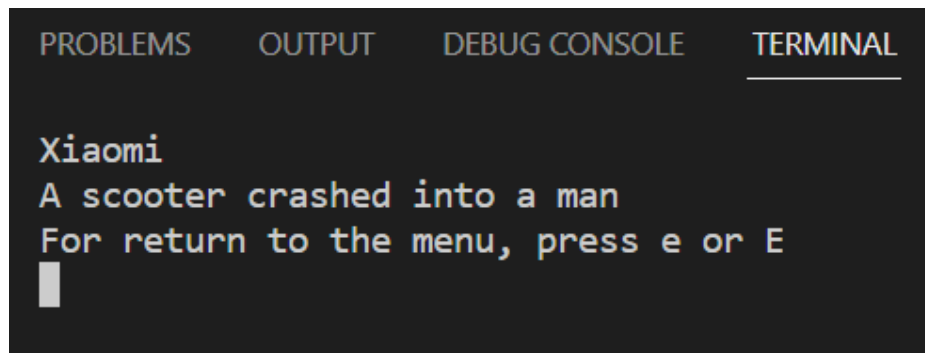
Ford
The driver violated the traffic rules
Toyota
The driver violated the traffic rules
For return to the menu, press e or E
█
```

Рисунок 12 – Выполнение уникального метода класса авто

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Russian
Tank is shooting
For return to the menu, press e or E
█
```

Рисунок 13 – Выполнение уникального метода класса танк

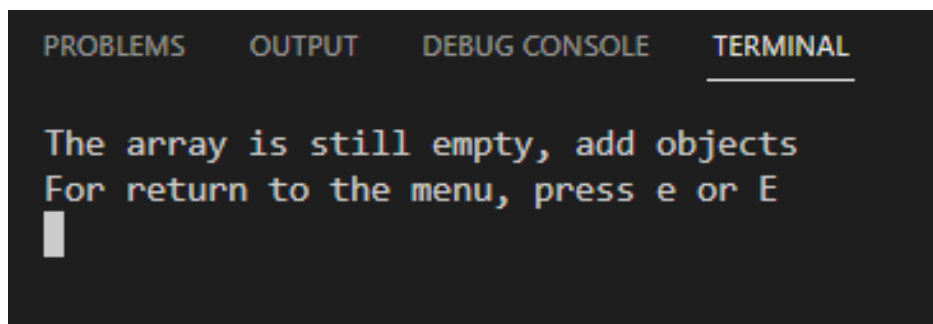


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Xiaomi
A scooter crashed into a man
For return to the menu, press e or E
█
```

Рисунок 14 – Выполнение уникального метода класса электросамокат

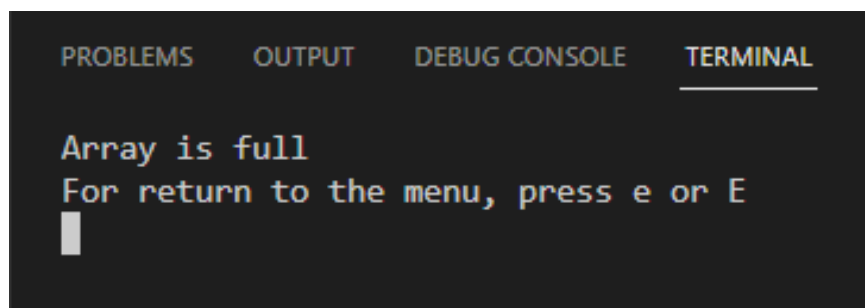
Если пользователь выбирает любое действие отличное от добавления записи, когда массив пуст, программа выведет сообщение, представленное на рисунке 15.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

The array is still empty, add objects
For return to the menu, press e or E
█
```

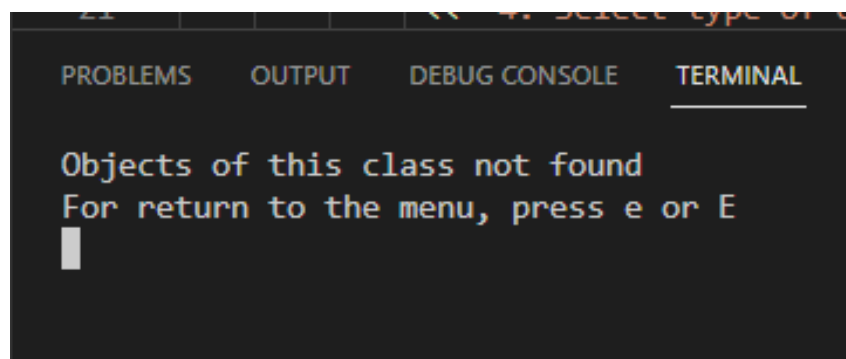
Рисунок 15 – Сообщение, когда массив пуст



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Array is full
For return to the menu, press e or E
█
```

Рисунок 16 – Сообщение, когда массив заполнен



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Objects of this class not found
For return to the menu, press e or E
█
```

Рисунок 17 – Сообщение, когда при выполнении уникального метода, объект данного класса не был найден