|  |  |  |
| --- | --- | --- |
| для прик эмбл | |  |
| МИНОБРНАУКИ РОССИИ | |  |
| Федеральное государственное бюджетное образовательное учреждение  высшего профессионального образования  **"Московский технологический университет"**  **МИРЭА** | |  |
|  | Институт информационных технологий (ИТ) | |
|  | **Кафедра инструментального и прикладного программного обеспечения (ИППО)** | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ** | |
| **по дисциплине** | |
| **“Объектно-ориентированное программирование”** | |
|  | |
| Выполнил студент группы \_ИКБО-13-17\_ | Хитров Н.О. |
| Принял Ассистент кафедры ИиППО | **Синицын А.В** |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | «\_\_»\_\_\_\_\_\_\_2018г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2018г. |  |

Москва 2018

Содержание

1. **Практическая работа №1**
2. **Практическая работа №2**
3. **Практическая работа №3**
4. **Практическая работа №4**
5. **Практическая работа №5**
6. **Практическая работа №6**
7. **Практическая работа №7**
8. **Практическая работа №8**

**Практическая работа №1**

Цель работы:

Целью данной практической работы является знакомство с синтаксисом, основными алгоритмическими конструкциями языке программирования С++, а также приобретение практических навыков разработки объектно-ориентированных программ на языке программирования С++.

Вариант 1.

Реализация:

#include <iostream>

using namespace std;

int main() {

string name;

cout << "Enter your name: ";

cin >> name;

cout << "Hello, " << name << endl;

return 0;

}Вывод на экран:

Enter your name: Nikolay

Hello, Nikolay

Вариант 2.

Реализация:

#include <iostream>

#include <stdio.h>

using namespace std;

int main() {

double number1, number2, result;

char operation;

bool repeat = true;

while (repeat) {

cout << "Enter expression with numbers in following format: {number} {operation} {number}\nExample: 12 + 31" << endl;

scanf("%lg %c %lg", &number1, &operation, &number2);

switch(operation) {

case '+':

result = number1 + number2;

repeat = false;

break;

case '-':

result = number1 - number2;

repeat = false;

break;

case '\*':

result = number1 \* number2;

repeat = false;

break;

case '/':

result = number1 / number2;

repeat = false;

break;

default:

cout << "Unsupported operation " << operation << " .\nTry on of supported: +, -, /, \*\n";

break;

}

}

printf("%lg %c %lg = %lg\n", number1, operation, number2, result);

return 0;

}

Вывод на экран:

Enter expression with numbers in following format: {number} {operation} {number}

Example: 12 + 31

8\*6

8 \* 6 = 48

Вывод:

В ходе выполнения данной парктической работы: ознакомился с синтаксисом языка с++ , получили практические навыки написания кода на этом языке.

**Практическая работа №2**

Цель работы:

Целью данной практической работы является получение практических навыков по разработке нелинейных алгоритмов с использованием конструкции «if-else»

Вариант 1.

Реализация:

#include <stdio.h>

#include <cmath>

using namespace std;

int main() {

double a = 4.8, b = -7.9, n, m;

if (a >= b) {

n = pow(a-b, 1./3);

} else {

n = a\*a + (a-b)/sin(a\*b);

}

if (n < b) {

m = (n+a)/-b + sqrt(pow(sin(a), 2) - cos(n));

} else if (n == b) {

m = b\*b + tan(n\*a);

} else {

m = pow(b, 3) + n\*a\*a;

}

printf("a: %lg\nb: %lg\nn: %lg\nm: %lg\n", a, b, n, m);

return 0;

}Вывод на экран:

a: 4.8

b: -7.9

n: 2.33311

m: -439.284

Вариант 2.

Реализация:

#include <stdio.h>

#include <cmath>

using namespace std;

int main() {

double a = -0.6, b = 5.3, z, t;

if (a < b) {

z = sqrt(a\*a - b\*b);

} else {

z = 1 - 2\*cos(a)\*sin(b);

}

if (z < b) {

t = pow(z +a\*a\*b, 1./3);

} else if (z == b) {

t = 1 - log10(z) + cos(a\*a\*b);

} else {

t = 1/(cos(z\*a));

}

printf("a: %lg\nb: %lg\nz: %lg\nt: %lg\n", a, b, z, t);

return 0;

}Вывод на экран:

a: 4.8

b: -7.9

n: 2.33311

m: -439.284

Вариант 3.

Реализация:

#include <stdio.h>

#include <cmath>

using namespace std;

int main() {

double a = 4.6, b = 1.5, y, t;

if (a <= b) {

y = ((a-b)/(a+b))\*((a+b)/(a\*a-a\*b+b\*2));

} else {

y = a + log2(b\*b);

}

if (y == b) {

t = (2\*y + sqrt(y\*y - a));

} else if (y < b) {

t = pow(sin(y), 2) + 1/tan(a-b);

} else {

t = pow(y\*sin(a), 1./3) + 1/(sqrt(y\*cos(b)));

}

printf("a: %lg\nb: %lg\ny: %lg\nt: %lg\n", a, b, y, t);

return 0;

}Вывод на экран:

a: 4.6

b: 1.5

y: 5.76993

t: nan

Вариант 4.

#include <stdio.h>

#include <cmath>

using namespace std;

int main() {

double a = 3.5, x = -0.7, y, t;

if (a <= x) {

y = a + log2(x+a);

} else {

y = sqrt(sin(a\*x));

}

if (a > y) {

t = y/(a-x);

} else if (a == y) {

t = y/(a-x) + (a+x)/(y\*y);

} else {

t = tan(a\*x) + cos(2\*a\*y);

}

printf("a: %lg\nx: %lg\ny: %lg\nt: %lg\n", a, x, y, t);

return 0;

}

Вывод на экран:

a: 3.5

x: -0.7

y: nan

t: nan

Вывод:

Приобрёл навыки по применению конструкции “if-else”.

**Практическая работа №3**

Цель работы:

Целью данной практической работы является приобретение практических навыков при разработке классов на языке программирования С++.

Вариант 1.

Реализация:

#include <iostream>

using namespace std;

class Book {

private:

string \_title;

string \_author;

int \_edition;

int \_year;

public:

Book(string title, string author, int edition, int year):

\_title{title}, \_author{author}, \_edition{edition}, \_year{year} {}

};

Вариант 2.

Реализация:

#include <iostream>

#include <vector>

using namespace std;

class GroupOfStudents {

public:

int course;

int group\_size;

string starosta;

string group\_name;

vector<string> students\_names;

};

Вариант 3.

Реализация:

#include <iostream>

using namespace std;

class Car {

public:

int rpm;

int count\_of\_fuel;

int max\_speed;

int count\_of\_wheels;

string model;

};

Вариант 4.

Реализация:

#include <iostream>

class Vector {

public:

int a, b, c;

Vector(int a, int b, int c): a{a}, b{b}. c{c} {}

Vector operator+(const Vector& a, const Vector& b) {

return Vector(a.a + b.a, a.b + b.b, a.c + b.c);

}

Vector operator-(const Vector& a, const Vector& b) {

return Vector(a.a - b.a, a.b - b.a, a.c - b.c);

}

};

Вариант 5.

Реализация:

#include <iostream>

#include <vector>

using namespace std;

class Library {

public:

int books\_count;

int count\_of\_readers;

int count\_of\_librarians;

vector<string> books\_titels;

};

Вывод:

Ознакомился с разработкой классов на языке с++.

**Практическая работа №4**

Цель работы:

Целью данной практической работы является изучение динамического выделения и удаления памяти, а также работа с указателями в языке C++.

Вариант 1.

Реализация:

#include <iostream>

#include <algorithm>

using namespace std;

double \*create\_array(size\_t size) {

return new double[size];

}

void fill\_array(double \*array, size\_t size) {

for (size\_t i = 0; i < size; i++) {

cin >> array[i];

}

}

void print\_array(double \*array, size\_t size) {

for (size\_t i = 0; i < size; i++) {

cout << array[i] << " ";

}

}

void free\_array(double \*array) {

delete[] array;

}

int main() {

double \*array;

size\_t size;

while (true) {

cout << "Enter array size: ";

cin >> size;

array = create\_array(size);

cout << "Fill array for " << size << " elements: ";

fill\_array(array, size);

cout << "Array:";

print\_array(array, size);

free\_array(array);

cout << "continue?[Y/n] ";

string next;

cin >> next;

transform(next.begin(), next.end(), next.begin(), ::tolower);

if (next == "y") {

continue;

}

break;

}

return 0;

}

Вывод на экран:

Enter array size: 2

Fill array for 2 elements: 1

3

Array:1 3 continue?[Y/n] y

Enter array size: 1

Fill array for 1 elements: 4

Array:4 continue?[Y/n] n

Вариант 2.

Реализация:

#include <iostream>

using namespace std;

void process(int \*array, size\_t size) {

for (size\_t i = 0; i < size/2; i++) {

int tmp = array[i\*2];

array[i\*2] = array[i\*2+1];

array[i\*2] = tmp;

}

}

void fill\_array(int \*array, size\_t size) {

for (size\_t i = 0; i < size; i++) {

cin >> array[i];

}

}

void print\_array(int \*array, size\_t size) {

for (size\_t i = 0; i < size; i++) {

cout << array[i] << " ";

}

}

int main() {

int \*array = new int[12];

cout << "Fill array for 12 elements: ";

fill\_array(array, 12);

cout << "Array: ";

process(array, 12);

cout << "Array: ";

print\_array(array, 12);

delete[] array;

return 0;

}

Вывод на экран:

Fill array for 12 elements: 3 4 5 12 0 56 3 123 5 7 1 2

Array: 3 4 5 12 0 56 3 123 5 7 1 2

Вариант 3.

Реализация:

#include <random>

#include <iostream>

using namespace std;

int \*\*create\_matrix(size\_t rows\_num, size\_t cols\_num) {

int \*\*matrix = new int\*[rows\_num];

for (size\_t i = 0; i < rows\_num; i++) {

matrix[i] = new int[cols\_num];

}

return matrix;

}

void free\_matrix(int \*\*matrix, size\_t rows\_num) {

for (size\_t i = 0; i < rows\_num; i++) {

delete[] matrix[i];

}

delete[] matrix;

}

void fill\_matrix(int \*\*matrix, size\_t rows\_num, size\_t cols\_num) {

random\_device rd;

mt19937 gen(rd());

for (size\_t i = 0; i < rows\_num; i++) {

for (size\_t j = 0; j < cols\_num; j++) {

matrix[i][j] = gen() % 1500 - 750;

}

}

}

void print\_matrix(int \*\*matrix, size\_t rows\_num, size\_t cols\_num) {

for (size\_t i = 0; i < rows\_num; i++) {

for (size\_t j = 0; j < cols\_num; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

}

int main() {

int rows, cols;

cout << "Enter rows and cols: ";

cin >> rows >> cols;

int \*\*matrix = create\_matrix(rows, cols);

fill\_matrix(matrix, rows, cols);

print\_matrix(matrix, rows, cols);

free\_matrix(matrix, rows);

return 0;

}

Вывод на экран:

Enter rows and cols: 4 5

536 480 -70 -578 267

-720 504 98 -710 -673

-723 168 675 288 -584

677 607 -651 363 -340

Вывод:

Ознакомился с динамическим выделением, очисткой памяти и указателями на языке с++.

**Практическая работа №5**

Цель работы:

Целью данной практической работы является приобретение практических навыков по программированию перегрузки функций на языке программирования С++.

Вариант 1.

Реализация:

void bubble(int\* a, int n) {

for (int i = n - 1; i >= 0; i--) {

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

if (a[j] > a[j + 1]) {

int tmp = a[j];

a[j] = a[j + 1];

a[j + 1] = tmp;

}

}

}

}

void bubble(double \*a, int n) {

for (int i = n - 1; i >= 0; i--) {

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

if (a[j] > a[j + 1]) {

double tmp = a[j];

a[j] = a[j + 1];

a[j + 1] = tmp;

}

}

}

}

Вариант 2.

Реализация:

void select(int \*a, int n) {

for (int i = 0; i < n - 1; i++) {

int min\_i = i;

for (int j = i + 1; j < n; j++) {

if (a[j] < a[min\_i]) {

min\_i = j;

}

}

int temp = a[i];

a[i] = a[min\_i];

a[min\_i] = temp;

}

}

void select(double \*a, int n) {

for (int i = 0; i < n - 1; i++) {

int min\_i = i;

for (int j = i + 1; j < n; j++) {

if (a[j] < a[min\_i]) {

min\_i = j;

}

}

double temp = a[i];

a[i] = a[min\_i];

a[min\_i] = temp;

}

}

Вариант 3.

Реализация:

void insert(int \*a, int n) {

for (int i = 1; i < n; i++) {

int cur = a[i];

int j = i;

while (j > 0 && cur < a[j - 1]) {

a[j] = a[j - 1];

j--;

}

a[j] = cur;

}

}

void insert(double \*a, int n) {

for (int i = 1; i < n; i++) {

double cur = a[i];

int j = i;

while (j > 0 && cur < a[j - 1]) {

a[j] = a[j - 1];

j--;

}

a[j] = cur;

}

}

Вариант 4.

Реализация:

#include <iostream>

#include <stdio.h>

using namespace std;

int main() {

double number1, number2, result;

char operation;

bool repeat = true;

while (repeat) {

cout << "Enter expression with numbers in following format: {number} {operation} {number}\nExample: 12 + 31" << endl;

scanf("%lg %c %lg", &number1, &operation, &number2);

switch(operation) {

case '+':

result = number1 + number2;

repeat = false;

break;

case '-':

result = number1 - number2;

repeat = false;

break;

case '\*':

result = number1 \* number2;

repeat = false;

break;

case '/':

result = number1 / number2;

repeat = false;

break;

default:

cout << "Unsupported operation " << operation << " .\nTry on of supported: +, -, /, \*\n";

break;

}

}

printf("%lg %c %lg = %lg\n", number1, operation, number2, result);

return 0;

}

Вывод на экран:

Enter expression with numbers in following format: {number} {operation} {number}

Example: 12 + 31

90/2

90 / 2 = 45

Вывод:

Ознакомился с перегрузкой функций на языке с++.

**Практическая работа №6**

Цель работы:

Целью данной практической работы является приобретение практических навыков использования принципа ООП – наследования для разработки программ на языке программирования C++.

Вариант 1.

Реализация:

#include <iostream>

using namespace std;

class Property {

public:

double worth;

Property(double worth): worth{worth} {}

virtual ~Property() {}

virtual double process() = 0;

};

class Appartment: public Property {

public:

Appartment(double worth): Property(worth) {}

~Appartment() {}

double process() {

return worth/1000;

}

};

class Car: public Property {

public:

Car(double worth): Property(worth) {}

~Car() {}

double process() {

return worth/200;

}

};

class CountryHouse: public Property {

public:

CountryHouse(double worth): Property(worth) {}

~CountryHouse() {}

double process() {

return worth/500;

}

};

int main() {

Property \*properties[7] = {

new Appartment(1000),

new Appartment(2000),

new Appartment(3000),

new Car(1000),

new Car(2000),

new CountryHouse(1000),

new CountryHouse(2000)

};

for (Property \*property: properties) {

cout << property->process() << endl;

}

for (Property \*property: properties) {

delete property;

}

return 0;

}

Вывод на экран:

**1**

**2**

**3**

**5**

**10**

**2**

**4**

Вариант 2.

Реализация:

#include <iostream>

#include <stdio.h>

#include <cmath>

using namespace std;

class Expression {

public:

double value;

Expression() {}

~ Expression() {}

virtual Expression\* clone() const {

return new Expression(\*this);

}

virtual double evaluate() {

return value;

}

};

class Variable : public Expression {

public:

Variable(double value) {

this->value = value;

}

~ Variable() {}

Variable\* clone() const override {

return new Variable(\*this);

}

};

class Add : public Expression {

public:

Add(Expression\* left, Expression\* right) {

value = left->evaluate() + right->evaluate();

}

~ Add() {}

Add\* clone() const override {

return new Add(\*this);

}

};

class Subtract : public Expression {

public:

Subtract(Expression\* left, Expression\* right) {

value = left -> evaluate() - right -> evaluate();

}

~ Subtract() {}

Subtract\* clone() const override {

return new Subtract(\*this);

}

};

class Multiplay : public Expression {

public:

Multiplay(Expression\* left, Expression\* right) {

value = left->evaluate() \* right->evaluate();

}

~ Multiplay() {}

Multiplay\* clone() const override {

return new Multiplay(\*this);

}

};

class Divide : public Expression {

public:

Divide(Expression\* left, Expression\* right) {

value = left->evaluate() / right->evaluate();

}

~ Divide() {}

Divide\* clone() const override {

return new Divide(\*this);

}

};

class Sin : public Expression {

public:

Sin(double value) {

this->value = value;

}

~ Sin() {}

double evaluate() override {

return sin(value);

}

Sin\* clone() const override {

return new Sin(\*this);

}

};

class Cos : public Expression {

public:

Cos(double value) {

this->value = value;

}

~Cos() {}

double evaluate() override {

return cos(value);

}

Cos\* clone() const override {

return new Cos(\*this);

}

};

class Ln : public Expression {

public:

Ln(double value) {

this->value = value;

}

~ Ln();

double evaluate() override {

return log(value);

}

Ln\* clone() const override {

return new Ln(\*this);

}

};

class Exp : public Expression {

public:

Exp(double value) {

this->value = value;

}

~ Exp();

double evaluate() override {

return exp(value);

}

Exp\* clone() const override {

return new Exp(\*this);

}

};

int main() {

// create fisrt expression

Add expr1(new Variable(90), new Variable(5));

cout << "90 + 5 = " << expr1.evaluate() << endl;

// copy first expression value

Expression\* ex1 = &expr1;

Expression\* ex2 = ex1->clone();

// use last expression value

Subtract expr2(ex2, new Variable(2));

cout << "95 - 2 = " << expr2.evaluate() << endl;

return 0;

}

Вывод на экран:

**90 + 5 = 95**

**95 - 2 = 93**

Вывод:

Ознакомился с наследованием классов на языке с++.

**Практическая работа №7**

Цель работы:

Целью данной практической работы является знакомство с абстрактными классами и механизмом виртуальных функций, а также множественным наследованием в языке программирования С++.

Вариант 1.

Реализация:

#include <iostream>

#include <stdio.h>

using namespace std;

class Animal {

protected:

string name;

string owner;

public:

Animal(string name, string owner) {

this->name = name;

this->owner = owner;

}

~ Animal() {}

void changeOwner(string newOwnerName) {

owner = newOwnerName;

}

void displayAnimalOwner() {

cout << name << "'s owner is " << owner << "." << endl;

}

virtual void animalSound() {

cout << "Animals sounds different" << endl;

}

};

class Cat : public Animal {

private:

string favoriteToy;

public:

Cat(string name, string owner): Animal(name,owner) {}

~ Cat() {}

void setFavoriteToy(string toy) {

favoriteToy = toy;

displayFavoriteToy();

}

void displayFavoriteToy() {

cout << name << "'s favorite toy is " << favoriteToy << "." << endl;

}

void animalSound() override {

cout << "meow-meow" << endl;

}

};

class Dog : public Animal {

private:

string favoritePlace;

public:

Dog(string name, string owner): Animal(name,owner) {}

~ Dog() {}

void setFavoritePlace(string place) {

favoritePlace = place;

displayFavoritePlace();

}

void displayFavoritePlace() {

cout << name << "'s favorite place to walk is " << favoritePlace << "." << endl;

}

void animalSound() override {

cout << "woof-woof" << endl;

}

};

class Parrot : public Animal {

private:

double flyingHeight;

public:

Parrot(string name, string owner): Animal(name,owner) {}

~ Parrot() {}

void setFlyingHeight(double height) {

flyingHeight = height;

displayFlyingHeight();

}

void displayFlyingHeight() {

cout << name << "'s max fly hight is " << flyingHeight << "." << endl;

}

void animalSound() override {

cout << "Prrrrivettt!" << endl;

}

};

class Human {

public:

void beckonAnimal(Animal \*sound) {

cout << "Human beckon an animal: ";

sound->animalSound();

}

};

int main() {

Parrot parrot("Kesha", "Nikolay");

parrot.changeOwner("Gleb");

parrot.displayAnimalOwner();

parrot.setFlyingHeight(100);

Human owner;

owner.beckonAnimal(&parrot);

return 0;

}

Вывод на экран:

**Kesha's owner is Gleb.**

**Kesha's max fly hight is 100.**

**Human beckon an animal: Prrrrivettt!**

Вариант 2.

Реализация:

#include <iostream>

#include <stdio.h>

#include <vector>

#include <math.h>

using namespace std;

class Figure {

protected:

double s;

double p;

double a;

double b;

double c;

double d;

Figure() {}

~ Figure() {}

void setFigure(double side\_a,double side\_b, double side\_c,double side\_d) {

a = side\_a;

b = side\_b;

c = side\_c;

d = side\_d;

}

virtual void findSquare(){}

void findPerimetr() {

p = a + b + c + d;

}

};

class Quadrangle: protected Figure {

public:

Quadrangle(){}

~ Quadrangle() {}

void findSquare() override {

findPerimetr();

s = sqrt((p/2-a)\*(p/2-b)\*(p/2-c)\*(p/2-d));

cout << "Square of Quadrangle is "<< s << endl;

}

};

class Rhombus: protected Figure {

private:

double h;

public:

Rhombus(){}

~ Rhombus() {}

void setFigure(double side\_a, double h) {

a = side\_a;

this->h = h;

}

void findSquare() override {

s = a\*h;

cout << "Square of Rhombus is " << s << endl;

}

};

class Rectangle: protected Figure {

public:

Rectangle(){}

~ Rectangle() {}

void setFigure(double side\_a,double side\_b) {

a = side\_a;

b = side\_b;

}

void findSquare() override {

s = a\*b;

cout << "Square of Rectangle is " << s << endl;

}

};

int main() {

Rectangle figure1;

figure1.setFigure(4, 2);

figure1.findSquare();

return 0;

}

Вывод на экран:

**Square of Rectangle is 8**

Вариант 3.

Реализация:

#include <iostream>

#include <stdio.h>

#include <string>

using namespace std;

class Transport {

protected:

double speed;

string color;

// points

int distance = 0;

virtual void move(int meters\_to\_go) {

unsigned int lastDistance = distance - meters\_to\_go;

cout << "I drove " << lastDistance << "!" << endl;

distance = distance + meters\_to\_go;

}

};

class Car: protected Transport {

private:

string mark;

public:

Car(string color, double speed) {

this->color = color;

this->speed = speed;

}

~ Car() {

cout << "objects class Car no more work" << endl;

}

void setCarMark (string mark) {

this->mark = mark;

cout << "This is " << mark << "." << endl;

}

void displayCarSpeed() {

cout << "The car speed is " << speed << " km/h." << endl;

}

void displayCarColor() {

cout << "This car is " << color << " color." << endl;

}

};

class Bus: protected Transport {

private:

int passengers\_count;

public:

Bus(string color) {

this->color = color;

}

~ Bus() {

cout << "objects class Bus no more work" << endl;

}

void addPassengers(int passengers) {

passengers\_count+= passengers;

}

void came\_out\_Passengers(int passengers) {

passengers\_count-=passengers;

}

void displayPassengersCount() {

cout << "In a bus " << passengers\_count << " passengers." << endl;

}

void displayBusColor() {

cout << "This bus is " << color << " color." << endl;

}

};

class Bike: protected Transport {

private:

string things\_in\_buscket[10];

int i = 0;

public:

~ Bike() {

cout << "objects class Bike no more work" << endl;

}

void add\_thing\_to\_buscket(string thing) {

things\_in\_buscket[i] = thing;

i++;

}

void display\_things\_in\_busket() {

cout << "Things in the bike busket:" << endl;

for (int c = 0; c < 10; c++) {

cout << c << "." << things\_in\_buscket[c] << endl;

}

}

};

int main() {

Car mayCar("gary",288.28);

mayCar.setCarMark("BMW");

mayCar.displayCarColor();

mayCar.displayCarSpeed();

mayCar.~Car();

cout << "\n";

Bus number8\_Bus("blue");

number8\_Bus.displayBusColor();

number8\_Bus.addPassengers(8);

number8\_Bus.came\_out\_Passengers(2);

number8\_Bus.displayPassengersCount();

number8\_Bus.~Bus();

cout << "\n";

Bike myBike;

myBike.add\_thing\_to\_buscket("can of water");

myBike.add\_thing\_to\_buscket("teachbook");

myBike.add\_thing\_to\_buscket("sandwich");

myBike.add\_thing\_to\_buscket("ear pods");

myBike.add\_thing\_to\_buscket("nupkin");

myBike.display\_things\_in\_busket();

cout << "\n";

return 0;

}

Вывод на экран:

**This is BMW.**

**This car is gary color.**

**The car speed is 288.28 km/h.**

**objects class Car no more work**

**This bus is blue color.**

**In a bus 6 passengers.**

**objects class Bus no more work**

**Things in the bike busket:**

**0.can of water**

**1.teachbook**

**2.sandwich**

**3.ear pods**

**4.nupkin**

**5.**

**6.**

**7.**

**8.**

**9.**

**objects class Bike no more work**

**objects class Bus no more work**

**objects class Car no more work**

Вариант 4.

Реализация:

#include <iostream>

#include <stdio.h>

#include <string>

using namespace std;

class Display {

protected:

string screen\_resolution;

double screen\_diagonal;

void display\_parameters() {

cout << "Display parameters: resolution - " << screen\_resolution << ", diagonal - " << screen\_diagonal << endl;

}

};

class KeyBoard {

protected:

int keys;

string language;

void keyboard\_parameters() {

cout << "Keyboard parameters: keys count - " << keys << ", main keyboard language - " << language << endl;

}

};

class Notebook: protected Display, protected KeyBoard {

private:

string processor;

int RAM;

public:

Notebook(string processor, int ram, string screen\_resolution, double screen\_diagonal, int keyboard\_keys, string keyboard\_language) {

this->processor = processor;

RAM = ram;

this->screen\_resolution = screen\_resolution;

this->screen\_diagonal = screen\_diagonal;

keys = keyboard\_keys;

language = keyboard\_language;

}

void display\_information() {

cout << "Notebook technical parameters: processor - " << processor << ", RAM - " << RAM << " GB" << endl;

display\_parameters();

keyboard\_parameters();

cout << endl;

}

};

class Smartphone: protected Display, protected KeyBoard {

private:

string OS\_version;

int camera\_parameters;

public:

Smartphone(string OS\_version, int camera\_parameters, string screen\_resolution, double screen\_diagonal, int keyboard\_keys, string keyboard\_language) {

this->OS\_version = OS\_version;

this->camera\_parameters = camera\_parameters;

this->screen\_resolution = screen\_resolution;

this->screen\_diagonal = screen\_diagonal;

keys = keyboard\_keys;

language = keyboard\_language;

}

void display\_information() {

cout << "Smartphone technical parameters: OS version - " << OS\_version << ", camera - " << camera\_parameters << " MP" << endl;

display\_parameters();

keyboard\_parameters();

cout << endl;

}

};

class Computer: protected Display, protected KeyBoard {

private:

string OS;

double weight;

public:

Computer(string OS, double weight, string screen\_resolution, double screen\_diagonal, int keyboard\_keys, string keyboard\_language) {

this->OS = OS;

this->weight = weight;

this->screen\_resolution = screen\_resolution;

this->screen\_diagonal = screen\_diagonal;

keys = keyboard\_keys;

language = keyboard\_language;

}

void display\_information() {

cout << "Computer technical parameters: OS - " << OS << ", weight - " << weight << " kg" << endl;

display\_parameters();

keyboard\_parameters();

cout << endl;

}

};

int main() {

Notebook notebook("Intel Core i7", 32, "1920x1080", 13, 73, "english");

notebook.display\_information();

Smartphone samsung("Android 7.0", 12, "2960x1440", 5.8, 38, "russian");

samsung.display\_information();

Computer Dell("Linux", 4.3, "3840x2160", 27, 73, "english");

Dell.display\_information();

return 0;

}

Вывод на экран:

**Notebook technical parameters: processor - Intel Core i7, RAM - 32 GB**

**Display parameters: resolution - 1920x1080, diagonal - 13**

**Keyboard parameters: keys count - 73, main keyboard language - english**

**Smartphone technical parameters: OS version - Android 7.0, camera - 12 MP**

**Display parameters: resolution - 2960x1440, diagonal - 5.8**

**Keyboard parameters: keys count - 38, main keyboard language - russian**

**Computer technical parameters: OS - Linux, weight - 4.3 kg**

**Display parameters: resolution - 3840x2160, diagonal - 27**

**Keyboard parameters: keys count - 73, main keyboard language - english**

Вывод:

Ознакомился с понятием абстрактного класса, виртуальными функциями и множественным наследованием на языке с++.

**Практическая работа №8**

Цель работы:

Целью данной практической работы является приобретение практических навыков по работе с файлами на языке программирования языке C++.

Вариант 1.

Реализация:

#include <iostream>

#include <fstream>

using namespace std;

int main(int argc, char \*\*argv) {

if (argc != 2) {

cout << "usage: program filename" << endl;

return 1;

}

ifstream f{argv[1]};

if (!f) {

cout << "error" << endl;

return 1;

}

string word;

while (f >> word) {

cout << word << endl;

}

return 0;

}

Вариант 2.

Реализация:

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream f{"tmp.txt"};

string line = " ";

while (line != "") {

getline(cin, line);

f << line << endl;

}

return 0;

}

Вывод:

Ознакомился с работой с файлами на языке с++.