**1.**

Пример, приведенный в задании, демонстрирует пример перегрузки функций в языке программирования C++.

В данном случае, определены две функции с одним и тем же именем **myfunc**, но с разными типами параметров: одна функция принимает целочисленный параметр типа **int**, а другая функция принимает параметр типа **double**. Это является примером перегрузки функций.

При выполнении функции **myfunc(10)**, передается целочисленное значение **10**, поэтому вызывается функция **myfunc** с параметром типа **int**. Результатом будет вывод числа **10**.

При выполнении функции **myfunc(5.5)**, передается число с плавающей запятой **5.5**, поэтому вызывается функция **myfunc** с параметром типа **double**. Результатом будет вывод числа **5.5**.

Таким образом, в данном примере перегрузка функций позволяет вызывать функцию с одним и тем же именем, но с разными типами параметров, в зависимости от переданных аргументов. Это позволяет использовать более гибкий и удобный интерфейс функций, а также повышает читабельность и понятность кода.

**2.**

*#include <iostream>*

*#include <cmath>*

*using namespace std;*

*// Функция для нахождения расстояния между двумя точками в двумерном пространстве*

*double distance2D(double x1, double y1, double x2, double y2)*

*{*

*return sqrt(pow((x2 - x1), 2) + pow((y2 - y1), 2));*

*}*

*// Функция для нахождения расстояния между двумя точками в трехмерном пространстве*

*double distance3D(double x1, double y1, double z1, double x2, double y2, double z2)*

*{*

*return sqrt(pow((x2 - x1), 2) + pow((y2 - y1), 2) + pow((z2 - z1), 2));*

*}*

*// Функция для нахождения периметра треугольника в двумерном пространстве*

*double calculatePerimeter2D(double x1, double y1, double x2, double y2, double x3, double y3)*

*{*

*double side1 = distance2D(x1, y1, x2, y2);*

*double side2 = distance2D(x2, y2, x3, y3);*

*double side3 = distance2D(x3, y3, x1, y1);*

*return side1 + side2 + side3;*

*}*

*// Функция для нахождения периметра треугольника в трехмерном пространстве*

*double calculatePerimeter3D(double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3)*

*{*

*double side1 = distance3D(x1, y1, z1, x2, y2, z2);*

*double side2 = distance3D(x2, y2, z2, x3, y3, z3);*

*double side3 = distance3D(x3, y3, z3, x1, y1, z1);*

*return side1 + side2 + side3;*

*}*

*int main()*

*{*

*// Двумерный треугольник*

*double x1\_2D, y1\_2D, x2\_2D, y2\_2D, x3\_2D, y3\_2D;*

*cout << "Введите координаты вершин треугольника (двумерный случай):\n";*

*cout << "Вершина 1 (x y): ";*

*cin >> x1\_2D >> y1\_2D;*

*cout << "Вершина 2 (x y): ";*

*cin >> x2\_2D >> y2\_2D;*

*cout << "Вершина 3 (x y): ";*

*cin >> x3\_2D >> y3\_2D;*

*double perimeter\_2D = calculatePerimeter2D(x1\_2D, y1\_2D, x2\_2D, y2\_2D, x3\_2D, y3\_2D);*

*cout << "Периметр треугольника (двумерный случай): " << perimeter\_2D << endl;*

*// Трехмерный треугольник*

*double x1\_3D, y1\_3D, z1\_3D, x2\_3D, y2\_3D, z2\_3D, x3\_3D, y3\_3D, z3\_3D;*

*cout << "Введите координаты вершин треугольника (трехмерный случай):\n";*

*cout << "Вершина 1 (x y z): ";*

*cin >> x1\_3D >> y1\_3D >> z1\_3D;*

*cout << "Вершина 2 (x y z): ";*

*cin >> x2\_3D >> y2\_3D >> z2\_3D;*

*cout << "Вершина 3 (x y z): ";*

*cin >> x3\_3D >> y3\_3D >> z3\_3D;*

*double perimeter\_3D = calculatePerimeter3D(x1\_3D, y1\_3D, z1\_3D, x2\_3D, y2\_3D, z2\_3D, x3\_3D, y3\_3D, z3\_3D);*

*cout << "Периметр треугольника (трехмерный случай): " << perimeter\_3D << endl;*

*return 0;*

*}*

**Задание**

**1.**

*#include <iostream>*

*using namespace std;*

*class Date {*

*private:*

*int day;*

*int month;*

*int year;*

*int days[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};*

*public:*

*Date(int d, int m, int y) : day(d), month(m), year(y) {}*

*void displayDate() {*

*cout << day << '.' << month << '.' << year << endl;*

*}*

*Date operator+(const Date& other) {*

*int newDay = day + other.day;*

*int newMonth = month + other.month;*

*int newYear = year + other.year;*

*if (newDay > days[month]) {*

*newDay -= days[month];*

*newMonth++;*

*if (newMonth > 12) {*

*newMonth -= 12;*

*newYear++;*

*}*

*}*

*return Date(newDay, newMonth, newYear);*

*}*

*Date operator-(const Date& other) {*

*int newDay = day - other.day;*

*int newMonth = month - other.month;*

*int newYear = year - other.year;*

*if (newDay <= 0) {*

*newMonth--;*

*if (newMonth <= 0) {*

*newMonth += 12;*

*newYear--;*

*}*

*newDay += days[newMonth];*

*}*

*return Date(newDay, newMonth, newYear);*

*}*

*Date operator+(int daysToAdd) const {*

*int newDay = day + daysToAdd;*

*int newMonth = month;*

*int newYear = year;*

*while (newDay > days[newMonth]) {*

*newDay -= days[newMonth];*

*newMonth++;*

*if (newMonth > 12) {*

*newMonth -= 12;*

*newYear++;*

*}*

*}*

*return Date(newDay, newMonth, newYear);*

*}*

*friend Date operator+(int daysToAdd, const Date& date) {*

*return date + daysToAdd;*

*}*

*Date operator-(int daysToSubtract) {*

*int newDay = day - daysToSubtract;*

*int newMonth = month;*

*int newYear = year;*

*while (newDay <= 0) {*

*newMonth--;*

*if (newMonth <= 0) {*

*newMonth += 12;*

*newYear--;*

*}*

*newDay += days[newMonth];*

*}*

*return Date(newDay, newMonth, newYear);*

*}*

*Date operator++() {*

*return \*this + 1;*

*}*

*Date operator--() {*

*return \*this - 1;*

*}*

*};*

*int main() {*

*Date date1(10, 2, 2023);*

*Date date2(15, 2, 2023);*

*Date date3 = date1 + date2;*

*date3.displayDate(); // Output: 25.4.4046*

*Date date4 = date2 - date1;*

*date4.displayDate(); // Output: 5.0.0*

*Date date5 = date1 + 7;*

*date5.displayDate(); // Output: 17.2.2023*

*Date date6 = 7 + date1;*

*date6.displayDate(); // Output: 17.2.2023*

*Date date7 = date2 - 5;*

*date7.displayDate(); // Output: 10.2.2023*

*++date1;*

*date1.displayDate(); // Output: 11.2.2023*

*--date2;*

*date2.displayDate(); // Output: 14.2.2023*

*return 0;*

*}*

**1.**

*#include <iostream>*

*using namespace std;*

*class Date {*

*private:*

*int day;*

*int month;*

*int year;*

*int days[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};*

*public:*

*Date(int d, int m, int y) {*

*day = d;*

*month = m;*

*year = y;*

*}*

*bool isLeapYear(int year) {*

*if (year % 4 == 0) {*

*if (year % 100 == 0) {*

*if (year % 400 == 0)*

*return true;*

*else*

*return false;*

*} else*

*return true;*

*} else*

*return false;*

*}*

*bool isValidDate() {*

*if (month < 1 || month > 12)*

*return false;*

*int maxDays = days[month];*

*if (month == 2 && isLeapYear(year))*

*maxDays++;*

*if (day < 1 || day > maxDays)*

*return false;*

*return true;*

*}*

*};*

*int main() {*

*Date date(29, 2, 2000); // Choose any valid date here.*

*if(date.isValidDate()) {*

*cout << "The date is valid." << endl;*

*} else {*

*cout << "The date is not valid." << endl;*

*}*

*return 0;*

*}*

**2.**

**3.**

**4.**

**5.**

**6.**

**7.**

**8.**