**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.**

Вы определили класс **Date**, который представляет дату, содержащую три закрытые поля типа **int**: **day** (день), **month** (месяц), **year** (год). Это класс, определенный для удобства управления датами.

Также включена внутренняя структура данных, представляющая количество дней в каждом месяце, **days[13]={0,31,28,31,30,31,30,31,31,30,31,30,31}**. Это используется при операциях с добавлением и вычитанием, чтобы обеспечить корректность даты.

Операторы **+,** **-, ++, --** перегружены для облегчения работы с датами:

Операторы **+** и **-** могут использоваться либо с двумя датами, либо с датой и числом дней. При сложении или вычитании двух дат результатом будет новая дата. При сложении даты и числа будет получена новая дата через заданное количество дней после изначальной даты. При вычитании числа из даты будет получена новая дата, которая на заданное количество дней раньше изначальной даты.

Операторы **++** и **--** используются для продвижения даты на один день вперед или назад.

Итак, вы создали мощный инструмент для работы с датами, который будет особенно полезен для строительной фирмы для управления сроками и графиками.

**2.**

Ваш класс **Date** уже содержит один конструктор:

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

Это параметризованный конструктор, который позволяет создать объект **Date**, указывая день, месяц и год. Это основной и единственный конструктор, который требуется для данного класса, исходя из показанной реализации и требований к операциям над датами.

Однако в зависимости от требований к программе вы можете реализовать дополнительные конструкторы. Например, вы можете добавить конструктор по умолчанию (который не принимает никаких аргументов), конструктор копирования или конструктор перемещения.

Конструктор по умолчанию может выглядеть так:

*Date() : day(1), month(1), year(2000) {} // Date set to a default value*

Конструктор копирования может выглядеть так:

*Date(const Date &d) : day(d.day), month(d.month), year(d.year) {}*

Конструктор перемещения может выглядеть так:

*Date(Date &&d) : day(d.day), month(d.month), year(d.year) {*

*d.day = 0;*

*d.month = 0;*

*d.year = 0;*

*}*

Обратите внимание, что конструктор перемещения используется для перехвата временных (rvalue) объектов, чтобы избежать ненужное копирование. Это более продвинутый аспект языка C++, и он может не быть необходим для вашего класса **Date**, если вы не планируете использовать функциональность перемещения.

Также важно отметить, что этот класс не учитывает високосные годы в своих расчетах. Если вам это нужно, вы должны будете добавить дополнительную логику для обработки високосных годов.

**3.**

*#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) {}*

*friend ostream& operator<< (ostream &out, const Date &date);*

*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++() {*

*\*this = \*this + 1;*

*return \*this;*

*}*

*Date operator--() {*

*\*this = \*this - 1;*

*return \*this;*

*}*

*};*

*ostream& operator<< (ostream &out, const Date &date)*

*{*

*out << date.day << '.' << date.month << '.' << date.year;*

*return out;*

*}*

*int main() {*

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

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

*Date date3 = date1 + date2;*

*cout << date3 << endl; // Output: 25.4.4046*

*Date date4 = date2 - date1;*

*cout << date4 << endl; // Output: 5.0.0*

*Date date5 = date1 + 7;*

*cout << date5 << endl; // Output: 17.2.2023*

*Date date6 = 7 + date1;*

*cout << date6 << endl; // Output: 17.2.2023*

*Date date7 = date2 - 5;*

*cout << date7 << endl; // Output: 10.2.2023*

*++date1;*

*cout << date1 << endl; // Output: 11.2.2023*

*--date2;*

*cout << date2 << endl; // Output: 14.2.2023*

*return 0;*

*}*

**4.**

Чтобы реализовать всё, что вы описали, нужно немного изменить класс **Date**. Новый класс будет называться **ConstructionProject**, и он будет использовать класс **Date** для хранения даты начала и дата конца строительства.

Также, мне нужно исправить вашу функцию добавления и вычитания дат, чтобы он учитывал високосные годы.

*#include<iostream>*

*#include<string>*

*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) {}*

*friend ostream& operator<< (ostream &out, const Date &date);*

*int calculateLeapYears(Date d);*

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

*int newDay = day + other.day;*

*int newMonth = month + other.month;*

*int newYear = year + other.year;*

*// account for leap year*

*if(newMonth == 2 && newDay > 28){*

*newYear += calculateLeapYears(Date(newDay, newMonth, newYear));*

*newDay -= days[newMonth];*

*}*

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

*newDay -= days[newMonth];*

*newMonth++;*

*}*

*if(newMonth > 12) {*

*newYear++;*

*newMonth = 1;*

*}*

*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;*

*// Account for negative days*

*if(newDay <= 0) {*

*newMonth--;*

*if(newMonth < 1){*

*newYear--;*

*newMonth = 12;*

*}*

*newDay += days[newMonth];*

*}*

*// Account for negative months*

*if(newMonth <= 0) {*

*newYear--;*

*newMonth += 12;*

*}*

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

*}*

*int getDay() const { return day; }*

*int getMonth() const { return month; }*

*int getYear() const { return year; }*

*bool isLeapYear(int y) {*

*if ((y % 4 == 0 && y % 100 != 0) || (y % 400 == 0)) {*

*return true;*

*}*

*return false;*

*}*

*void incrementDate(int d, int m, int y) {*

*day += d;*

*month += m;*

*year += y;*

*if (month > 12) {*

*year += month / 12;*

*month = month % 12;*

*}*

*if (month == 2 && day > 28) {*

*if (isLeapYear(year)) {*

*if (day > 29) {*

*day -= 29;*

*month++;*

*}*

*} else {*

*day -= 28;*

*month++;*

*}*

*} else {*

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

*day -= days[month];*

*month++;*

*}*

*}*

*if (month > 12) {*

*year += month / 12;*

*month = month % 12;*

*}*

*}*

*void decrementDate(int d, int m, int y) {*

*day -= d;*

*month -= m;*

*year -= y;*

*while(day <= 0){*

*month--;*

*if(month == 0){*

*year--;*

*month = 12;*

*}*

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

*day += 29;*

*} else {*

*day += days[month];*

*}*

*}*

*while(month <= 0){*

*year--;*

*month += 12;*

*}*

*}*

*};*

*ostream& operator<< (ostream &out, const Date &date)*

*{*

*out << date.getDay() << '.' << date.getMonth() << '.' << date.getYear();*

*return out;*

*}*

*class ConstructionProject {*

*string name;*

*Date startDate;*

*Date endDate;*

*public:*

*ConstructionProject(string name, Date s, Date e) : name(name), startDate(s), endDate(e) {}*

*Date getStartDate() const { return startDate; }*

*Date getEndDate() const { return endDate; }*

*void setConstructionTime(int days, int months, int years) {*

*Date newDate = startDate;*

*newDate.incrementDate(days, months, years);*

*endDate = newDate;*

*}*

*void calculateStartDate(int days, int months, int years) {*

*Date newDate = endDate;*

*newDate.decrementDate(days, months, years);*

*startDate = newDate;*

*}*

*};*

*int main() {*

*ConstructionProject project1("Office Building", Date(1, 4, 2010), Date(1, 4, 2010));*

*project1.setConstructionTime(10, 3, 2);*

*cout << "End date of construction: " << project1.getEndDate() << endl; // Output: 11.7.2012*

*ConstructionProject project2("Apartment Complex", Date(11, 7, 2012), Date(11, 7, 2012));*

*project2.calculateStartDate(10, 3, 2);*

*cout << "Start date of construction: " << project2.getStartDate() << endl; // Output: 1.4.2010*

*return 0;*

*}*

**5.**

**6.**

**7.**

**8.**