**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>*

*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) {}*

*Date() : day(1), month(1), year(1900) {}*

*void showDate() {*

*std::cout << day << "/" << month << "/" << year << std::endl;*

*}*

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

*return date + numDays;*

*}*

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

*int newDay = day + other.day;*

*int newMonth = month + other.month;*

*int newYear = year + other.year;*

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

*newMonth++;*

*newDay -= days[month];*

*}*

*if (newMonth > 12) {*

*newYear++;*

*newMonth -= 12;*

*}*

*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 < 1) {*

*newMonth--;*

*newDay += days[month - 1];*

*}*

*if (newMonth < 1) {*

*newYear--;*

*newMonth += 12;*

*}*

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

*}*

*Date operator+(int numDays) {*

*int newDay = day + numDays;*

*int newMonth = month;*

*int newYear = year;*

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

*newDay -= days[newMonth];*

*newMonth++;*

*if (newMonth > 12) {*

*newMonth = 1;*

*newYear++;*

*}*

*}*

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

*}*

*Date operator-(int numDays) {*

*int newDay = day - numDays;*

*int newMonth = month;*

*int newYear = year;*

*while (newDay < 1) {*

*newMonth--;*

*if (newMonth < 1) {*

*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(24, 10, 2023);*

*Date date2;*

*Date date3 = date1 + date2;*

*Date date4 = date1 - date2;*

*Date date5 = date1 + 10;*

*Date date6 = 5 + date2;*

*Date date7 = date1 - 5;*

*Date date8 = ++date1;*

*Date date9 = --date2;*

*date3.showDate();*

*date4.showDate();*

*date5.showDate();*

*date6.showDate();*

*date7.showDate();*

*date8.showDate();*

*date9.showDate();*

*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.**

Класс ***Date***, который вы использовали, содержит два конструктора:

1. Первый конструктор принимает три параметра (день, месяц и год), чтобы включить пользователю возможность задавать специфические даты при создании объекта:

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

1. Второй конструктор - это конструктор по умолчанию. Этот конструктор не принимает никаких параметров, а просто устанавливает дату на 1 января 1900 года. Если пользователь создает объект даты, не указывая дату, конструктор по умолчанию будет использоваться.

*Date() : day(1), month(1), year(1900) {}*

Таким образом, в классе ***Date*** нужно два конструктора. Один из них позволяет пользователю задать конкретную дату, а другой устанавливает дефолтную дату, если никакая другая дата не указана.

**3.**

Для перегрузки оператора вставки, вам нужно создать немодифицирующий friend-функцию, которая будет принимать ссылку на std::ostream и дату

*#include<iostream>*

*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) {}*

*Date() : day(1), month(1), year(1900) {}*

*void showDate() {*

*std::cout << day << "/" << month << "/" << year << std::endl;*

*}*

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

*return date + numDays;*

*}*

*//перегрузка операции <<*

*friend std::ostream& operator<<(std::ostream& os, const Date& date) {*

*os << date.day << "/" << date.month << "/" << date.year ;*

*return os;*

*}*

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

*int newDay = day + other.day;*

*int newMonth = month + other.month;*

*int newYear = year + other.year;*

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

*newMonth++;*

*newDay -= days[month];*

*}*

*if (newMonth > 12) {*

*newYear++;*

*newMonth -= 12;*

*}*

*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 < 1) {*

*newMonth--;*

*newDay += days[month - 1];*

*}*

*if (newMonth < 1) {*

*newYear--;*

*newMonth += 12;*

*}*

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

*}*

*Date operator+(int numDays) {*

*int newDay = day + numDays;*

*int newMonth = month;*

*int newYear = year;*

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

*newDay -= days[newMonth];*

*newMonth++;*

*if (newMonth > 12) {*

*newMonth = 1;*

*newYear++;*

*}*

*}*

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

*}*

*Date operator-(int numDays) {*

*int newDay = day - numDays;*

*int newMonth = month;*

*int newYear = year;*

*while (newDay < 1) {*

*newMonth--;*

*if (newMonth < 1) {*

*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(24, 10, 2023);*

*Date date2;*

*Date date3 = date1 + date2;*

*Date date4 = date1 - date2;*

*Date date5 = date1 + 10;*

*Date date6 = 5 + date2;*

*Date date7 = date1 - 5;*

*Date date8 = ++date1;*

*Date date9 = --date2;*

*date3.showDate();*

*date4.showDate();*

*date5.showDate();*

*date6.showDate();*

*date7.showDate();*

*date8.showDate();*

*date9.showDate();*

*std::cout << date1 << std::endl;*

*return 0;*

*}*

Заметьте, что сейчас вы можете выводить дату на печать, используя оператор << напрямую, без необходимости вызова функции showDate().

**4.**

Ваш код уже содержит перегруженные операторы "+" и "-", которые позволяют добавлять и вычитать дни к текущей дате. Для обычных дат это работает, но в случае с "периодом строительства", который содержит года, месяцы и дни, нам придется слегка изменить ваш код.

Я предложу два новых класса, один "Period", который будет содержать периоды времени в годах, месяцах и днях, и класс "Building", который будет использовать и класс "Date" и класс "Period" для определения даты начала и конца строительства

*#include<iostream>*

*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) {}*

*Date() : day(1), month(1), year(1900) {}*

*void showDate() {*

*std::cout << day << "/" << month << "/" << year << std::endl;*

*}*

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

*return date + numDays;*

*}*

*friend std::ostream& operator<<(std::ostream& os, const Date& date) {*

*os << date.day << "/" << date.month << "/" << date.year ;*

*return os;*

*}*

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

*int newDay = day + other.day;*

*int newMonth = month + other.month;*

*int newYear = year + other.year;*

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

*newMonth++;*

*newDay -= days[month];*

*}*

*if (newMonth > 12) {*

*newYear++;*

*newMonth -= 12;*

*}*

*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 < 1) {*

*newMonth--;*

*newDay += days[month - 1];*

*}*

*if (newMonth < 1) {*

*newYear--;*

*newMonth += 12;*

*}*

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

*}*

*Date operator+(int numDays) {*

*int newDay = day + numDays;*

*int newMonth = month;*

*int newYear = year;*

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

*newDay -= days[newMonth];*

*newMonth++;*

*if (newMonth > 12) {*

*newMonth = 1;*

*newYear++;*

*}*

*}*

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

*}*

*Date operator-(int numDays) {*

*int newDay = day - numDays;*

*int newMonth = month;*

*int newYear = year;*

*while (newDay < 1) {*

*newMonth--;*

*if (newMonth < 1) {*

*newMonth = 12;*

*newYear--;*

*}*

*newDay += days[newMonth];*

*}*

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

*}*

*Date operator++() {*

*return \*this + 1;*

*}*

*Date operator--() {*

*return \*this - 1;*

*}*

*};*

*class Period {*

*public:*

*int days;*

*int months;*

*int years;*

*Period(int d, int m, int y) : days(d), months(m), years(y) {}*

*};*

*class Building {*

*Date startDate;*

*Period period;*

*public:*

*Building(const Date& d, const Period& p) : startDate(d), period(p) {}*

*Date getEndDate() {*

*Date endDate = startDate + period.days + (period.months \* 30) + (period.years \* 365);*

*return endDate;*

*}*

*Date getStartDate() {*

*Date startDate = getEndDate() - period.days - (period.months \* 30) - (period.years \* 365);*

*return startDate;*

*}*

*};*

*int main() {*

*Date date1(24, 10, 2023);*

*Date date2;*

*Date date3 = date1 + date2;*

*Date date4 = date1 - date2;*

*Date date5 = date1 + 10;*

*Date date6 = 5 + date2;*

*Date date7 = date1 - 5;*

*Date date8 = ++date1;*

*Date date9 = --date2;*

*date3.showDate();*

*date4.showDate();*

*date5.showDate();*

*date6.showDate();*

*date7.showDate();*

*date8.showDate();*

*date9.showDate();*

*std::cout << date1 << std::endl;*

*// объявление объекта Building с датой начала и периодом строительства*

*Building building(Date(1, 4, 2010), Period(10, 3, 2));*

*// вывод завершающей даты*

*std::cout << "Дата сдачи объекта: ";*

*building.getEndDate().showDate();*

*// вывод начальной даты*

*std::cout << "Дата начала строительства: ";*

*building.getStartDate().showDate();*

*return 0;*

*}*

Обратите внимание, что в этом коде для простоты считается, что каждый месяц содержит 30 дней, а каждый год — 365 дней. Конечно, это не совсем так на самом деле из-за високосных годов и месяцев с различным количеством дней, поэтому можно внести дополнительные усовершенствования, чтобы более точно учесть эти переменные.

**5.**

Ваш код уже содержит перегруженные операции для сложения даты и целого числа: «дата» + «int» и «int» + «дата». Эти операторы используются для добавления определенного количества дней к дате, что соответствует описанию в вашей задаче.

1. «дата» + «int»: Ваш класс **Date**уже содержит перегруженный оператор сложения для даты и целого числа. Оператор **Date operator+(int numDays)** добавляет **numDays**дней к объекту **Date**, с которым он вызывается, создавая новый объект **Date**.
2. «int» + «дата»: Этот тип оператора перегрузки реализован с помощью дружественной функции **Date operator+(int numDays, Date& date)**. В данном случае, порядок операндов меняется, но результат остается таким же, как в предыдущем пункте.

Соответственно, в вашем коде вы уже имеете возможность считать дату доставки строительных материалов, добавляя к некоей изначальной дате (например, дате начала строительства) необходимое количество дней.

**6.**

*#include<iostream>*

*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) {}*

*Date() : day(1), month(1), year(1900) {}*

*void showDate() {*

*std::cout << day << "/" << month << "/" << year << std::endl;*

*}*

*// int + дата*

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

*return date + numDays;*

*}*

*//перегрузка оператора <<*

*friend std::ostream& operator<<(std::ostream& os, const Date& date) {*

*os << date.day << "/" << date.month << "/" << date.year ;*

*return os;*

*}*

*//дата + дата*

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

*int newDay = day + other.day;*

*int newMonth = month + other.month;*

*int newYear = year + other.year;*

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

*newMonth++;*

*newDay -= days[month];*

*}*

*if (newMonth > 12) {*

*newYear++;*

*newMonth -= 12;*

*}*

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

*}*

*// дата - дата*

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

*int newDay = day - other.day;*

*int newMonth = month - other.month;*

*int newYear = year - other.year;*

*if (newDay < 1) {*

*newMonth--;*

*newDay += days[month - 1];*

*}*

*if (newMonth < 1) {*

*newYear--;*

*newMonth += 12;*

*}*

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

*}*

*// дата + int*

*Date operator+(int numDays) {*

*int newDay = day + numDays;*

*int newMonth = month;*

*int newYear = year;*

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

*newDay -= days[newMonth];*

*newMonth++;*

*if (newMonth > 12) {*

*newMonth = 1;*

*newYear++;*

*}*

*}*

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

*}*

*//вычитания числа дней из даты*

*Date operator-(int numDays) {*

*int newDay = day - numDays;*

*int newMonth = month;*

*int newYear = year;*

*while (newDay < 1) {*

*newMonth--;*

*if (newMonth < 1) {*

*newMonth = 12;*

*newYear--;*

*}*

*newDay += days[newMonth];*

*}*

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

*}*

*Date operator++() {*

*return \*this + 1;*

*}*

*Date operator--() {*

*return \*this - 1;*

*}*

*};*

*class Period {*

*public:*

*int days;*

*int months;*

*int years;*

*Period(int d, int m, int y) : days(d), months(m), years(y) {}*

*};*

*class Building {*

*Date startDate;*

*Period period;*

*public:*

*Building(const Date& d, const Period& p) : startDate(d), period(p) {}*

*Date getEndDate() {*

*Date endDate = startDate + period.days + (period.months \* 30) + (period.years \* 365);*

*return endDate;*

*}*

*Date getStartDate() {*

*Date startDate = getEndDate() - period.days - (period.months \* 30) - (period.years \* 365);*

*return startDate;*

*}*

*};*

*int main() {*

*Date date1(24, 10, 2023);*

*Date date2;*

*Date date3 = date1 + date2;*

*Date date4 = date1 - date2;*

*Date date5 = date1 + 10;*

*Date date6 = 5 + date2;*

*Date date7 = date1 - 5;*

*Date date8 = ++date1;*

*Date date9 = --date2;*

*date3.showDate();*

*date4.showDate();*

*date5.showDate();*

*date6.showDate();*

*date7.showDate();*

*date8.showDate();*

*date9.showDate();*

*std::cout << date1 << std::endl;*

*// объявление объекта Building с датой начала и периодом строительства*

*Building building(Date(1, 4, 2010), Period(10, 3, 2));*

*// вывод завершающей даты*

*std::cout << "Дата сдачи объекта: ";*

*building.getEndDate().showDate();*

*// вывод начальной даты*

*std::cout << "Дата начала строительства: ";*

*building.getStartDate().showDate();*

*return 0;*

*}*

Ваша операторная функция "-" для вычитания числа дней (int) из даты уже реализована в конце вашего класса Date. Она получает число дней, которые нужно вычесть из даты, и затем вычисляет новую дату

Учитывая этот код, мы можем использовать этот оператор для определения даты начала заработной платы. Предположим, у нас есть дата начисления зарплаты и количество отработанных дней. Мы можем вычесть количество отработанных дней из даты начисления зарплаты, чтобы получить дату начала заработной платы

*// предположим, у нас есть следующая дата начисления зарплаты*

*Date salaryDate(30, 8, 2023);*

*// предположим, что количество отработанных дней = 10*

*int workedDays = 10;*

*// находим дату начала начисления зарплаты*

*Date salaryStartDate = salaryDate - workedDays;*

*// выводим дату начала начисления зарплаты*

*std::cout << "Дата начала начисления зарплаты: " << salaryStartDate << std::endl;*

**7.**

*#include<iostream>*

*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) {}*

*Date() : day(1), month(1), year(1900) {}*

*void showDate() {*

*std::cout << day << "/" << month << "/" << year << std::endl;*

*}*

*// int + дата*

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

*return date + numDays;*

*}*

*friend std::ostream& operator<<(std::ostream& os, const Date& date) {*

*os << date.day << "/" << date.month << "/" << date.year ;*

*return os;*

*}*

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

*int newDay = day + other.day;*

*int newMonth = month + other.month;*

*int newYear = year + other.year;*

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

*newMonth++;*

*newDay -= days[month];*

*}*

*if (newMonth > 12) {*

*newYear++;*

*newMonth -= 12;*

*}*

*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 < 1) {*

*newMonth--;*

*newDay += days[month - 1];*

*}*

*if (newMonth < 1) {*

*newYear--;*

*newMonth += 12;*

*}*

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

*}*

*// дата + int*

*Date operator+(int numDays) {*

*int newDay = day + numDays;*

*int newMonth = month;*

*int newYear = year;*

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

*newDay -= days[newMonth];*

*newMonth++;*

*if (newMonth > 12) {*

*newMonth = 1;*

*newYear++;*

*}*

*}*

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

*}*

*//вычитания числа дней из даты*

*Date operator-(int numDays) {*

*int newDay = day - numDays;*

*int newMonth = month;*

*int newYear = year;*

*while (newDay < 1) {*

*newMonth--;*

*if (newMonth < 1) {*

*newMonth = 12;*

*newYear--;*

*}*

*newDay += days[newMonth];*

*}*

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

*}*

*//оператор ++, это +1 день*

*Date operator++() {*

*return \*this + 1;*

*}*

*//оператор --, это -1 день*

*Date operator--() {*

*return \*this - 1;*

*}*

*};*

*class Period {*

*public:*

*int days;*

*int months;*

*int years;*

*Period(int d, int m, int y) : days(d), months(m), years(y) {}*

*};*

*class Building {*

*Date startDate;*

*Period period;*

*public:*

*Building(const Date& d, const Period& p) : startDate(d), period(p) {}*

*Date getEndDate() {*

*Date endDate = startDate + period.days + (period.months \* 30) + (period.years \* 365);*

*return endDate;*

*}*

*Date getStartDate() {*

*Date startDate = getEndDate() - period.days - (period.months \* 30) - (period.years \* 365);*

*return startDate;*

*}*

*};*

*int main() {*

*Date date1(24, 10, 2023);*

*Date date2;*

*Date date3 = date1 + date2;*

*Date date4 = date1 - date2;*

*Date date5 = date1 + 10;*

*Date date6 = 5 + date2;*

*Date date7 = date1 - 5;*

*Date date8 = ++date1;*

*Date date9 = --date2;*

*date3.showDate();*

*date4.showDate();*

*date5.showDate();*

*date6.showDate();*

*date7.showDate();*

*date8.showDate();*

*date9.showDate();*

*std::cout << date1 << std::endl;*

*// объявление объекта Building с датой начала и периодом строительства*

*Building building(Date(1, 4, 2010), Period(10, 3, 2));*

*// вывод завершающей даты*

*std::cout << "Дата сдачи объекта: ";*

*building.getEndDate().showDate();*

*// вывод начальной даты*

*std::cout << "Дата начала строительства: ";*

*building.getStartDate().showDate();*

*return 0;*

*}*

**8.**

Код, представленный вам, определяет два пользовательских класса - **Date** и **Building**. Класс **Date** представляет собой конкретную дату, представленную в виде дня, месяца и года.

Сначала отметим перегружаемые операторы, которые были определены в классе **Date**:

1. **operator+** и **operator-** определены для добавления или вычитания количества дней или другого объекта Date из текущего объекта **Date**.
2. **operator++** и **operator--** представляют собой инкремент и декремент, т. е. добавление или вычитание одного дня от текущей даты соответственно.
3. **operator<<** — она позволяет напечатать объект **Date** напрямую через потоковый оператор вывода, так что, когда вы делаете **std::cout << someDate;**, он автоматически печатает разделенные слешем день, месяц и год.
4. Также определен непосредственно метод **showDate()**, который напечатает дату с помощью **std::cout**.

Второй класс, **Building**, представляет собой здание со связанной датой начала работ и периодом строительства. Он определяет методы **getEndDate()** и **getStartDate()**, которые вычисляют конечную или начальную даты на основе периода строительства.

Работа перегрузки операторов в функции **main()**, показана следующим образом:

1. Создаются два объекта **Date** — **date1** и **date2**. **date1** устанавливается на 24/10/2023, в то время как **date2** использует конструктор по умолчанию, который устанавливает дату на 01/01/1900.
2. **operator+** и operato**r-** используются для суммирования **date1** и **date2** и записи результатов в **date3** и **date4** соответственно. Эти даты затем печатаются.
3. Происходит добавление и вычитание константного числа дней с помощью **operator+** и **operator-** и хранение результатов в **date5** и **date7**. Эти даты также выводятся.
4. **operator++** и **operator--** используются для инкремента и декремента **date1** и **date2** и сохранения результатов в **date8** и **date9** соответственно. Эти даты печатаются.
5. Для вывода **date1** используется **operator<<.**
6. Потом создается экземпляр здания, и демонстрируется вычисление даты начала и окончания строительства, используя методы **getStartDate()** и **getEndDate()** соответственно.