ФГБОУ ВО Хакасский государственный университет им. Н.Ф. Катанова

Инженерно-технологический институт

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

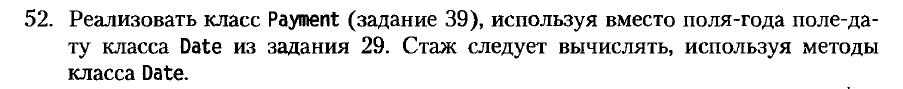
Проверил

доцент кафедры ПОВТиАС Санников Е.В.

Выполнил

студент группы 120-1 Хвостов Д.А.

Абакан 2022



#include <iostream>

#include <cmath>

using namespace std;

class Date {

private:

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

public:

unsigned int year;

unsigned int month;

int day;

void Init(const unsigned int year,

const unsigned int month,

const unsigned int day) {

this->year = year;

this->day = day;

this->month = month;

}

string toString() {

return to\_string(year) + "." + to\_string(month) + "." + to\_string(day);

}

bool IsLeapYear(int year) {

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

return true;

}

return false;

}

bool IsThisLeapYear() {

return IsLeapYear(year);

}

void operator+=(const int &rhs) {

this->day += rhs;

Normalize();

}

void operator-=(const int &rhs) {

this->day -= rhs;

Normalize();

}

unsigned int Day() {

return day;

}

unsigned int Mount() {

return month;

}

unsigned int Year() {

return year;

}

Date &operator=(Date &date) {

this->year = date.year;

this->day = date.day;

this->month = date.month;

return \*this;

}

private:

void Normalize() {

while (day > YearsOfMonth(year, month) || day <= 0) {

if (day <= 0) {

day += YearsOfMonth(year, month - 1);

if (month == 1) {

month = 12;

year--;

} else {

month--;

}

} else {

day -= YearsOfMonth(year, month);

if (month == 12) {

year++;

month = 1;

} else {

month++;

}

}

} }

int YearsOfMonth(const unsigned int y, unsigned int m) {

int dayInMonth;

dayInMonth = days[m];

if (month == 2 && IsLeapYear(y)) {

dayInMonth += 1;

}

return dayInMonth;

}

};

#pragma region iostream overloading date

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

out << date.year << "." << date.month << "." << date.day << endl;

return out;

}

istream &operator>>(istream &inp, Date &date) {

cout << "Введите год: " << endl;

inp >> date.year;

cout << "Введите месяц: " << endl;

inp >> date.month;

cout << "Введите день: " << endl;

inp >> date.day;

return inp;

}

#pragma endregion

class Payment {

public:

string name;

double salary;

Date date;

double perks\_in\_procent;

double tax;

double count\_days\_in\_mount;

double count\_worker\_days;

double sum\_accrued;

double sum\_withheld;

void Init(string name, double salary, Date &date,

double perks\_in\_procent, double tax, double count\_days\_in\_mount,

double count\_worker\_days, double sum\_accrued, double sum\_withheld) {

this-> count\_worker\_days = count\_worker\_days;

this->name = name;

this->salary = salary;

this->perks\_in\_procent = perks\_in\_procent;

this->tax = tax;

this->count\_days\_in\_mount = count\_days\_in\_mount;

this-> count\_worker\_days = count\_worker\_days;

this->sum\_accrued = sum\_accrued;

this->sum\_withheld = sum\_withheld;

this->date = date; }

double get\_withheld\_sum(){

sum\_withheld = get\_accrued\_sum() / 100;

return sum\_withheld; }

double get\_accrued\_sum(){

sum\_accrued = salary \* (count\_days\_in\_mount - count\_worker\_days);

return sum\_accrued; }

double get\_sum\_on\_hand(){

return get\_accrued\_sum() - get\_withheld\_sum() - get\_accrued\_sum() \*13 / 100; }

int get\_experience(const Date& current\_date) const{

cout<<date << endl;

cout<< current\_date << endl;

int year = current\_date.year - date.year;

if (date.month > current\_date.month){

return year - 1; }

if (date.month == current\_date.month && date.day > current\_date.day){

return year - 1;}

return year;}};

ostream &operator<<(ostream &out, const Payment &payment) {

out << "Name: " << payment.name << "|salary: " << payment.salary <<

"|Date: " << payment.date << "|perks\_in\_procent: " <<

payment.perks\_in\_procent << "|tax: " << payment.tax <<

"|count\_days\_in\_mount: " << "|sum\_accrued: " << payment.sum\_accrued <<

"| sum\_withheld: " << payment.sum\_withheld;

return out;}

int main() {

Date date;

date.Init(2012, 2, 12);

cout << date;

date += 100;

cout << date;

date -= 300;

cout << date;

Payment payment;

payment.Init("Cat", 12, date,12, 12, 12, 12, 23, 4);

Date date1;

date1.Init(2022, 1, 11);

cout << payment.get\_experience(date1) << endl;

cout << payment;

}