МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе

по дисциплине   
«Языки программирования высокого уровня»

Тема: «Исключения»

Вариант № 13

Выполнил:   
 ст. гр. УИС-111:

Мензульский Д.Д.

Проверил:

Савушкин С.А.

Москва 2022 г.

# Постановка задачи

## Общие требования

Во всех заданиях реализуемые функции должны генерировать подходящие исключения. Обработку исключений нужно выполнять главной функцией, которая должна демонстрировать обработку всех перехватываемых исключений.

Функции, реализуемые в заданиях, обязаны выполнять проверку передаваемых параметров и генерировать исключения в случае ошибочных. Все функции реализуются в четырех вариантах:

- без спецификации исключений;

- со спецификацией throw();

- с конкретной спецификацией с подходящим стандартным исключением;

- спецификация с собственным реализованным исключением.

Собственное исключение должно быть реализовано в трех вариантах: как пустой класс, как независимый класс с полями-параметрами функции, как наследник от стандартного исключения с полями.

## Вариант задания

Функция вычисляет положительность телефонного разговора в минутах, принимая время начала и окончания. Время предоставлено структурой с тремя полями: час, минута, секунда. Неполная минута считается за полную.

# Авторское понимание и формализация задачи

Создать класс времени. Сделать функцию подсчёта разницы времени.

# Текст программы

#### - без спецификации исключений;

## #include <iostream>

#include <string>

using namespace std;

struct Time{

private:

int hours, minute, second;

public:

int time\_to\_second(){

return (hours\*60\*60)+minute\*60+second;

}

void set\_time(int &hours, int &minute, int &second){

if((hours<24&&hours>=0)&&(second<60&&second>=0)&&(minute<60&&minute>=0)){

this->hours=hours;

this->minute=minute;

this->second=second;

}

else{

cout<<"ERROR"<<endl;

}

}

void set\_hours(int hours){

this->hours=hours;

}

void set\_minute(int minute){

this->minute=minute;

}

void set\_sec#include <iostream>

#include <string>

using namespace std;

enum ERROR{fail\_time};

struct Time{

private:

int hours, minute, second;

public:

Time(){};

Time(int &hours, int &minute, int &second){

this->hours=hours;

this->minute=minute;

this->second=second;

}

int time\_to\_second(){

return (hours\*60\*60)+minute\*60+second;

}

void set\_time(int hours, int minute, int second){

this->hours=hours;

this->minute=minute;

this->second=second;

}

void set\_hours(int hours){

this->hours=hours;

}

void set\_minute(int minute){

this->minute=minute;

}

void set\_second(int second){

this->second=second;

}

int get\_hours(){

return hours;

}

int get\_minute(){

return minute;

}

int get\_second(){

return second;

}

};

Time TimeCall;

void timeOfCall(Time &start\_call, Time &end\_call){

int call\_second = end\_call.time\_to\_second()-start\_call.time\_to\_second();

TimeCall.set\_hours(call\_second/3600);

call\_second = call\_second%3600;

TimeCall.set\_minute(call\_second/60);

TimeCall.set\_second(call\_second%60);

};

int main(){

int time\_start\_call[]={12,23,54};

int time\_end\_call[]={12,32,32};

Time start\_call(time\_start\_call[0],time\_start\_call[1],time\_start\_call[2]);

Time end\_call(time\_end\_call[0],time\_end\_call[1],time\_end\_call[2]);

timeOfCall(start\_call, end\_call);

cout<<TimeCall.get\_hours()<<":"<<TimeCall.get\_minute()<<":"<<TimeCall.get\_second()<<endl;

} ond(int second){

this->second=second;

}

int get\_hours()const{

return hours;

}

int get\_minute()const{

return minute;

}

int get\_second()const{

return second;

}

};

Time TimeCall;

void timeOfCall(Time &start\_call, Time &end\_call){

int call\_second = end\_call.time\_to\_second()-start\_call.time\_to\_second();

TimeCall.set\_hours(call\_second/3600);

call\_second = call\_second%3600;

TimeCall.set\_minute(call\_second/60);

TimeCall.set\_second(call\_second%60);

};

void print(){

cout<<TimeCall.get\_hours()<<":"<<TimeCall.get\_minute()<<":"<<TimeCall.get\_second()<<endl;

};

int main(){

Time start\_call;

Time end\_call;

int time\_start\_call[2];

int time\_end\_call[2];

cout<<"example"<<endl;

cout<<"12:12:12"<<endl;

string call\_start;

cout<<"Начало звонка"<<endl;

cin>>call\_start;

time\_start\_call[0]=((call\_start[0]-'0')\*10)+call\_start[1]-'0';

time\_start\_call[1]=((call\_start[3]-'0')\*10)+call\_start[4]-'0';

time\_start\_call[2]=((call\_start[6]-'0')\*10)+call\_start[7]-'0';

cout<<"Конец звонка"<<endl;

string call\_end;

cin>>call\_end;

time\_end\_call[0]=((call\_end[0]-'0')\*10)+(call\_end[1]-'0');

time\_end\_call[1]=((call\_end[3]-'0')\*10)+(call\_end[4]-'0');

time\_end\_call[2]=((call\_end[6]-'0')\*10)+(call\_end[7]-'0');

start\_call.set\_time(time\_start\_call[0],time\_start\_call[1],time\_start\_call[2]);

end\_call.set\_time(time\_end\_call[0],time\_end\_call[1],time\_end\_call[2]);

timeOfCall(start\_call, end\_call);

print();

}

#### - со спецификацией throw();

#include <iostream>

#include <string>

using namespace std;

struct Time{

private:

int hours, minute, second;

public:

int time\_to\_second(){

return (hours\*60\*60)+minute\*60+second;

}

void set\_time(int &hours, int &minute, int &second){

try{

set\_hours(hours);

}

catch(int test){

cout<<"Not right hours: "<<test<<endl;

}

try{

set\_minute(minute);

}

catch(int test){

cout<<"Not right minute: "<<test<<endl;

}

try{

set\_second(second);

}

catch(int test){

cout<<"Not right second: "<<test<<endl;

}

}

void set\_hours(int hours){

if(hours<24&&hours>=0){

this->hours=hours;

}

else{

throw(hours);

}

}

void set\_minute(int minute){

if(minute<60&&minute>=0){

this->minute=minute;

}

else{

throw(minute);

}

}

void set\_second(int second){

if(second<60&&second>=0){

this->second=second;

}

else{

throw(second);

}

}

int get\_hours(){

return hours;

}

int get\_minute(){

return minute;

}

int get\_second(){

return second;

}

};

class TimeCall:public Time{

public:

TimeCall(Time &start\_call, Time &end\_call){

int call\_second = end\_call.time\_to\_second()-start\_call.time\_to\_second();

if(call\_second<0){

call\_second=call\_second+(60\*60\*24);

}

set\_hours(call\_second/3600);

call\_second = call\_second%3600;

set\_minute(call\_second/60);

set\_second(call\_second%60);

}

};

int main(){

Time start\_call;

Time end\_call;

int time\_start\_call[2];

int time\_end\_call[2];

cout<<"example"<<endl;

cout<<"12:12:12"<<endl;

char call\_start[8];

cout<<"Начало звонка"<<endl;

cin>>call\_start;

time\_start\_call[0]=((call\_start[0]-'0')\*10)+call\_start[1]-'0';

time\_start\_call[1]=((call\_start[3]-'0')\*10)+call\_start[4]-'0';

time\_start\_call[2]=((call\_start[6]-'0')\*10)+call\_start[7]-'0';

start\_call.set\_time(time\_start\_call[0],time\_start\_call[1],time\_start\_call[2]);

cout<<"Конец звонка"<<endl;

char call\_end[8];

cin>>call\_end;

time\_end\_call[0]=((call\_end[0]-'0')\*10)+(call\_end[1]-'0');

time\_end\_call[1]=((call\_end[3]-'0')\*10)+(call\_end[4]-'0');

time\_end\_call[2]=((call\_end[6]-'0')\*10)+(call\_end[7]-'0');

end\_call.set\_time(time\_end\_call[0],time\_end\_call[1],time\_end\_call[2]);

TimeCall x(start\_call, end\_call);

cout<<x.get\_hours()<<":"<<x.get\_minute()<<":"<<x.get\_second()<<endl;

}

#### - с конкретной спецификацией с подходящим стандартным исключением;

#include <iostream>

#include <string>

using namespace std;

struct Time{

private:

int hours, minute, second;

public:

int time\_to\_second(){

return (hours\*60\*60)+minute\*60+second;

}

void set\_time(int &hours, int &minute, int &second){

try{

set\_hours(hours);

set\_minute(minute);

set\_second(second);

}

catch(std::range\_error err){

cout<<"Not right: "<<err.what()<<endl;

}

}

void set\_hours(int hours){

if(hours<24&&hours>=0){

this->hours=hours;

}

else{

throw std::range\_error("hours");

}

}

void set\_minute(int minute){

if(minute<60&&minute>=0){

this->minute=minute;

}

else{

throw std::range\_error("minute");

}

}

void set\_second(int second){

if(second<60&&second>=0){

this->second=second;

}

else{

throw std::range\_error("second");

}

}

int get\_hours(){

return hours;

}

int get\_minute(){

return minute;

}

int get\_second(){

return second;

}

};

class TimeCall:public Time{

public:

TimeCall(Time &start\_call, Time &end\_call){

int call\_second = end\_call.time\_to\_second()-start\_call.time\_to\_second();

if(call\_second<0){

call\_second=call\_second+(60\*60\*24);

}

set\_hours(call\_second/3600);

call\_second = call\_second%3600;

set\_minute(call\_second/60);

set\_second(call\_second%60);

}

};

int main(){

Time start\_call;

Time end\_call;

int time\_start\_call[2];

int time\_end\_call[2];

cout<<"example"<<endl;

cout<<"12:12:12"<<endl;

char call\_start[8];

cout<<"Начало звонка"<<endl;

cin>>call\_start;

time\_start\_call[0]=((call\_start[0]-'0')\*10)+call\_start[1]-'0';

time\_start\_call[1]=((call\_start[3]-'0')\*10)+call\_start[4]-'0';

time\_start\_call[2]=((call\_start[6]-'0')\*10)+call\_start[7]-'0';

start\_call.set\_time(time\_start\_call[0],time\_start\_call[1],time\_start\_call[2]);

cout<<"Конец звонка"<<endl;

char call\_end[8];

cin>>call\_end;

time\_end\_call[0]=((call\_end[0]-'0')\*10)+(call\_end[1]-'0');

time\_end\_call[1]=((call\_end[3]-'0')\*10)+(call\_end[4]-'0');

time\_end\_call[2]=((call\_end[6]-'0')\*10)+(call\_end[7]-'0');

end\_call.set\_time(time\_end\_call[0],time\_end\_call[1],time\_end\_call[2]);

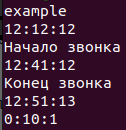
TimeCall x(start\_call, end\_call);

cout<<x.get\_hours()<<":"<<x.get\_minute()<<":"<<x.get\_second()<<endl;

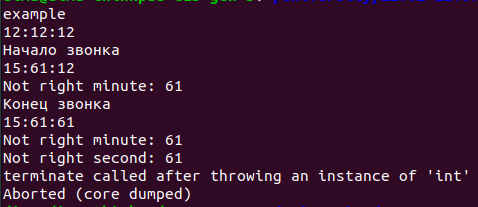
}

# Результаты тестирования

#### - без спецификации исключений



#### - со спецификацией throw();



#### - с конкретной спецификацией с подходящим стандартным исключением;

