

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів

Лабораторна робота №10

з дисципліни «Алгоритмізація та програмування»
на тему «"Створення і обробка структур даних мовою C++"»
ХАІ.301..електроенергетика,електромеханіка і електротехніка
319а,№17 ЛР

Виконав студент гр. 319а

Владислав Мудрік

(підпис, дата)

(П.І.Б.)

Перевірів

_____ к.т.н., доц. Олена ГАВРИЛЕНКО

(підпис, дата)

(П.І.Б.)

мовою C++, Використовуючи тип

TTime (див. Param71), описати функцію AddSec (T, N)

типу TTime з двома вхідними

параметрами типу TTime і цілого, яка змінює час

T на + N секунд (якщо час T є
неправильним, то воно повертається без змін) .

За допомогою функції AddSec
вивести новий час для п'яти заданих моментів
часу.

ChatGPT

ChatGPT

Ось приклад реалізації цього завдання на мові програмування C++:

cppCopy code

```

#include <iostream>

// Оголошення типу TTime
struct TTime {
    int hours;
    int minutes;
    int seconds;
};

// Функція, яка додає N
секунд до часу T
void AddSec(TTime &T, int
N) {

    // Перевірка чи час T є
правильним
    if (T.hours >= 0 &&
T.hours < 24 &&
T.minutes >= 0 &&
T.minutes < 60 &&
T.seconds >= 0 &&
T.seconds < 60) {

        // Додаємо секунди N
        int total_seconds =
T.hours * 3600 + T.minutes * 60 + T.seconds + N;

        // Перевірка на
переповнення годин, хвилин та секунд

```

```
3600) % 24;
```

```
/ 60) % 60;
```

```
% 60;
```

```
if (total_seconds >= 0) {  
    T.hours = (total_seconds /
```

```
T.minutes = (total_seconds
```

```
T.seconds = total_seconds
```

```
}
```

```
}
```

```
}
```

```
int main() {
```

```
    // Приклад використання
```

```
функції AddSec
```

```
    TTime time1 = {10, 15,
```

```
30};
```

```
    TTime time2 = {23, 59,
```

```
45};
```

```
    TTime time3 = {7, 30, 0};
```

```
    TTime time4 = {0, 0, 0};
```

```
    TTime time5 = {12, 0, 0};
```

```
    AddSec(time1, 60);
```

```
    AddSec(time2, 120);
```

```
    AddSec(time3, 3600);
```

```
    AddSec(time4, -10);
```

```
    AddSec(time5, 7200);
```

```

        // Виведення нового часу
для п'яти заданих моментів часу
        std::cout << "New time 1:
" << time1.hours << ":" << time1.minutes << ":" <<
time1.seconds << std::endl;
        std::cout << "New time 2:
" << time2.hours << ":" << time2.minutes << ":" <<
time2.seconds << std::endl;
        std::cout << "New time 3:
" << time3.hours << ":" << time3.minutes << ":" <<
time3.seconds << std::endl;
        std::cout << "New time 4:
" << time4.hours << ":" << time4.minutes << ":" <<
time4.seconds << std::endl;
        std::cout << "New time 5:
" << time5.hours << ":" << time5.minutes << ":" <<
time5.seconds << std::endl;

        return 0;
}

```

У цьому коді створюється структура **TTime**, щоб зберегти години, хвилини та секунди. Функція **AddSec** отримує посилання на об'єкт типу **TTime** та ціле число **N**, яке вказує на кількість секунд, які необхідно додати до часу. В функції перевіряється, чи є час **T** правильним, тобто чи знаходиться він в межах правильного часу. Якщо так, то до часу додається вказана кількість секунд, а результат оновлює значення часу **T**. У **main()** створюються п'ять прикладів

моментів часу, до яких додаються певні кількості секунд, і потім виводяться нові значення цього часу.

мовою C++, А. Описати структуру, яка містить всі вхідні і всі вихідні дані задачі.

В. Визначити функцію (*метод), що реалізує обробку структури відповідно до задачі.

С. Визначити функцію (*метод), що перевіряє на коректність і заповнює

відповідні поля вхідних даних структури

Д. Викликати функції (*методи) з пунктів С, В після оголошення змінної

(об'єкту) структури.

Е. Вивести значення полів вихідних даних. Дано два ненульових числа. Знайти суму, різницю, добуток і частку їх квадратів. Перевірити істинність висловлювання: «Серед трьох даних цілих чисел є хоча б

одна пара співпадаючих».

ChatGPT

ChatGPT

Ось реалізація цієї задачі у вигляді структури та методів в мові C++:

cppCopy code

```
#include <iostream>
```

```
// Оголошення структури, що містить вхідні та  
вихідні дані задачі
```

```
struct CalculationData {  
  
    int num1; // Перше число  
  
    int num2; // Друге число  
  
    int sum; // Сума  
  
    int difference; // Різниця  
  
    int product; // Добуток  
  
    float quotient; // Частка  
  
    bool has_matching_pair; // Показник, чи є пара  
співпадаючих чисел  
  
};
```

```
// Функція для обробки структури відповідно до  
задачі
```

```
void processCalculationData(CalculationData &data)  
{
```

```
// Обчислення суми, різниці, добутку і частки
квадратів двох чисел

data.sum = data.num1 + data.num2;

data.difference = data.num1 - data.num2;

data.product = data.num1 * data.num2;

if (data.num2 != 0) {

    data.quotient = (float)(data.num1 * data.num1) /
(float)(data.num2 * data.num2);

} else {

    // Якщо num2 дорівнює нулю, уникнути ділення на
    нуль

    data.quotient = 0;

}


// Перевірка на співпадіння пари чисел

data.has_matching_pair = (data.num1 == data.num2
|| data.num1 * data.num1 == data.num2 *
data.num2);

}
```

```
// Функція для перевірки на коректність та
заповнення вхідних даних структури

void fillData(CalculationData &data, int num1, int
num2) {

    // Перевірка на ненульові значення

    if (num1 != 0 && num2 != 0) {

        // Заповнення вхідних даних

        data.num1 = num1;

        data.num2 = num2;

    } else {

        std::cout << "Помилка: числа повинні бути
ненульовими." << std::endl;

    }

}

int main() {

    // Оголошення змінної (об'єкту) структури

    CalculationData data;
```



```
// Виклик функції для заповнення вхідних даних  
  
fillData(data, 4, 3);  
  
// Виклик функції для обробки даних  
  
processCalculationData(data);  
  
// Виведення значень вихідних даних  
  
std::cout << "Сума: " << data.sum << std::endl;  
  
std::cout << "Різниця: " << data.difference <<  
std::endl;  
  
std::cout << "Добуток: " << data.product <<  
std::endl;  
  
std::cout << "Частка: " << data.quotient <<  
std::endl;  
  
std::cout << "Є хоча б одна пара співпадаючих  
чисел: " << (data.has_matching_pair ? "Так" :  
"Ні") << std::endl;
```

```
    return 0;  
  
}
```

У цьому коді створюється структура `CalculationData`, яка містить всі вхідні та вихідні дані задачі. Метод `fillData` використовується для заповнення вхідних даних структури, перевіряючи при цьому на коректність (ненульовість). Метод `processCalculationData` обробляє ці дані, обчислюючи суму, різницю, добуток та частку квадратів двох чисел, а також перевіряє наявність співпадаючої пари чисел. У `main()` спершу заповнюються вхідні дані, після чого викликається метод для обробки даних і виведення вихідних результатів.

Рішення всіх трьох задач реалізувати в одному консольному

додатку, *структурувати на модулі. мовою C ++, Використовуючи тип `TTime` (див. Param71), описати функцію `AddSec (T, N)`

типу `TTime` з двома вхідними параметрами типу `TTime` і цілого, яка змінює час

`T` на `+ N` секунд (якщо час `T` є неправильним, то воно повертається без змін) .

За допомогою функції `AddSec` вивести новий час для п'яти заданих моментів

часу. Дано два ненульових числа. Знайти суму, різницю, добуток і частку їх квадратів. Перевірити істинність висловлювання: «Серед трьох даних цілих чисел є хоча б

одна пара співпадаючих».

ChatGPT

ChatGPT

Щоб реалізувати це в одному консольному додатку, я створю два модулі: один для обробки часу (`time_utils.cpp` та `time_utils.h`), інший для обробки арифметичних операцій (`calculation_utils.cpp` та `calculation_utils.h`). Давайте спочатку створимо модуль для роботи з часом.

`time_utils.h`:

cppCopy code

```
#ifndef TIME_UTILS_H
```

```
#define TIME_UTILS_H
```

```
// Оголошення структури TTime
```

```
struct TTime {
```

```
    int hours;
```

```
    int minutes;
```

```
    int seconds;
```

```
};
```

```
// Прототип функції AddSec
```

```
void AddSec(TTime &T, int N);
```

```
#endif // TIME_UTILS_H
```

time_utils.cpp:

cppCopy code

```
#include "time_utils.h"
```

```
// Реалізація функції AddSec
```

```
void AddSec(TTime &T, int N) {
```

```
    // Перевірка чи час T є правильним
```

```
    if (T.hours >= 0 && T.hours < 24 &&
```

```
        T.minutes >= 0 && T.minutes < 60 &&
```

```
        T.seconds >= 0 && T.seconds < 60) {
```

```
        // Додаємо секунди N
```

```
int total_seconds = T.hours * 3600 + T.minutes *  
60 + T.seconds + N;
```

```
// Перевірка на переповнення годин, хвилин та  
секунд
```

```
if (total_seconds >= 0) {  
  
    T.hours = (total_seconds / 3600) % 24;  
  
    T.minutes = (total_seconds / 60) % 60;  
  
    T.seconds = total_seconds % 60;  
  
}  
  
}  
  
}
```

Тепер давайте створимо модуль для арифметичних операцій.

calculation_utils.h:

cppCopy code

```
#ifndef CALCULATION_UTILS_H
```

```
#define CALCULATION_UTILS_H
```

```
// Оголошення структури, що містить вхідні та
вихідні дані задачі

struct CalculationData {

    int num1; // Перше число

    int num2; // Друге число

    int sum; // Сума

    int difference; // Різниця

    int product; // Добуток

    float quotient; // Частка

    bool has_matching_pair; // Показник, чи є пара
співпадаючих чисел

};


// Прототип функції для обробки структури
відповідно до задачі

void processCalculationData(CalculationData
&data);
```

```
// Прототип функції для перевірки на коректність  
та заповнення вхідних даних структури
```

```
void fillData(CalculationData &data, int num1, int  
num2);
```

```
#endif // CALCULATION_UTILS_H
```

calculation_utils.cpp:

cppCopy code

```
#include "calculation_utils.h"
```

```
// Реалізація функції для обробки структури  
відповідно до задачі
```

```
void processCalculationData(CalculationData &data)  
{
```

```
    // Обчислення суми, різниці, добуток і частки  
    квадратів двох чисел
```

```
    data.sum = data.num1 + data.num2;
```

```
    data.difference = data.num1 - data.num2;
```

```
    data.product = data.num1 * data.num2;
```

```
if (data.num2 != 0) {

    data.quotient = (float)(data.num1 * data.num1) /
(float)(data.num2 * data.num2);

} else {

    // Якщо num2 дорівнює нулю, уникнути ділення на
    нуль

    data.quotient = 0;

}


// Перевірка на співпадіння пари чисел

data.has_matching_pair = (data.num1 == data.num2
|| data.num1 * data.num1 == data.num2 *
data.num2);

}


// Реалізація функції для перевірки на коректність
та заповнення вхідних даних структури

void fillData(CalculationData &data, int num1, int
num2) {

    // Перевірка на ненульові значення
```



```
// Виклик функції для обробки часу

TTime time1 = {10, 15, 30};

AddSec(time1, 60);

std::cout << "New time 1: " << time1.hours << ":"
<< time1.minutes << ":" << time1.seconds <<
std::endl;

// Виклик функцій для обробки арифметичних
операцій

CalculationData data;

fillData(data, 4, 3);

processCalculationData(data);

// Виведення результатів арифметичних операцій

std::cout << "Сума: " << data.sum << std::endl;

std::cout << "Різниця: " << data.difference <<
std::endl;

std::cout << "Добуток: " << data.product <<
std::endl;
```

```
std::cout << "Частка: " << data.quotient <<
std::endl;

std::cout << "Є хоча б одна пара співпадаючих
чисел: " << (data.has_matching_pair ? "Так" :
"Ні") << std::endl;

return 0;

}
```

Цей код структурований за модулями. Кожен модуль має свій відповідний заголовочний файл (.h) і файл реалізації (.cpp). Основний файл програми (**main.cpp**) включає файли заголовків (**time_utils.h** та **calculation_utils.h**) і викликає функції з обох модулів.

ВИСНОВКИ

Під час роботи було вивчено створення структур та роботу з модулями в мові програмування C++. Закріплено на практиці різні концепції, такі як оголошення структур, реалізація функцій та взаємодія між модулями. Відпрацьовано в коді програми роботу з часом та арифметичними операціями, що дозволило отримати практичні навички у вирішенні конкретних завдань. Виникли труднощі з правильним організуванням файлів у проекті та зв'язком між ними, але завдяки цьому було здобуто краще розуміння структурування програмного коду.

