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 С ++"»
ХАІ.301. електроенергетика,електромеханіка і електротехніка
319а.11ЛР

Виконав студент гр. 319 а_____
(№ групи)

Єфімов Нікіта Максимович_____
(Підпис, дата) (П.І.Б.)

Перевірив к.т.н., доцент _____
(Науковий ступінь, вчене звання)
_____ О. В. Гавриленко
(Підпис, дата) (П.І.Б.)

,Використовуючи тип TTime (див. Param71), описати процедуру PrevSec (T) з
параметром типу TTime, яка змінює час на -1 секунду (якщо час T є неправильним, то воно не змінюється). Запис T є вхідним і вихідним параметром. Застосувати процедуру PrevSec до п'яти заданих моментів часу.

Ось як можна описати процедуру `PrevSec` у мові C++, використовуючи тип `TTime`:

cppCopy code

```
#include <iostream>

// Припустимо, що TTime вже визначено як тип часу у вашій програмі
```

```
// Перевизначення типу TTime для прикладу
typedef int TTime;

// Перевизначення типу TTime може бути яким завгодно
// Наприклад, ви можете використовувати структуру або
клас для представлення часу
// Наприклад:
// struct TTime {
// int hour;
// int minute;
// int second;
// };

// Функція, що зменшує час на одну секунду
void PrevSec(TTime& time) {
if (time > 0) { // перевіряємо, щоб час був додатнім
--time; // зменшуємо на одну секунду
}
}

int main() {
// П'ять заданих моментів часу
TTime time1 = 3600; // 1 година
TTime time2 = 7200; // 2 години
TTime time3 = 86399; // 23 години, 59 хвилин, 59 секунд
TTime time4 = 0; // Півночі
TTime time5 = -1; // Неправильний час

// Застосовуємо процедуру PrevSec до п'яти заданих
моментів часу
PrevSec(time1);
```

```
PrevSec(time2);
PrevSec(time3);
PrevSec(time4);
PrevSec(time5);

// Виводимо змінені значення часу
std::cout << "Time 1: " << time1 << std::endl; // 3599
(59 хвилин, 59 секунд)
std::cout << "Time 2: " << time2 << std::endl; // 7199
(1 година, 59 хвилин, 59 секунд)
std::cout << "Time 3: " << time3 << std::endl; // 86398
(23 години, 59 хвилин, 58 секунд)
std::cout << "Time 4: " << time4 << std::endl; // 0
(немає змін, бо 0 - 1 = -1 < 0)
std::cout << "Time 5: " << time5 << std::endl; // -1
(немає змін, бо неправильний час)

return 0;
}
```

У цьому прикладі `PrevSec` зменшує час на одну секунду, якщо час більше нуля. Функція `main` демонструє використання `PrevSec` на п'яти заданих моментах часу.

A. Описати структуру, яка містить всі вхідні і всі вихідні дані задачі.
B. Визначити функцію (*метод), що реалізує обробку структури відповідно до задачі.
C. Визначити функцію (*метод), що перевіряє на коректність і заповнює відповідні поля вхідних даних стуктури
D. Викликати функції (*методи) з пунктів C, B після оголошення змінної (об’єкту) структури.
E. Вивести значення полів вихідних даних.мовою C ++,Дано діаметр окружності d. Знайти її довжину L = π · d. Як значення π вважати

рівним 3.14.Дано числа A, B, C (число A не дорівнює 0). Розглянувши дискримінант D = B2
- 4 · A · C, перевірити істинність висловлювання: «Квадратне рівняння A · x 2 + B · x + C = 0 має дійсні коріння».

Ось реалізація цих вимог:

cppCopy code

```
#include <iostream>
#include <cmath>

struct CircleData {
double diameter; // вхідний параметр: діаметр окружності
double length; // вихідний параметр: довжина окружності
};

void processCircle(CircleData& data) {
data.length = M_PI * data.diameter; // обчислюємо довжину окружності
}

bool checkAndFill(double A, double B, double C, CircleData& data) {
if (A != 0) { // перевіряємо, що A не дорівнює 0
double discriminant = B * B - 4 * A * C; // обчислюємо дискримінант

if (discriminant >= 0) { // перевіряємо, чи існують дійсні корені
```

```
data.diameter = sqrt(discriminant); // користуємося дискримінантом для заповнення діаметра
return true; // повертаємо true, якщо у рівняння є дійсні корені
}
}
return false; // повертаємо false, якщо у рівняння немає дійсних коренів
}

int main() {
CircleData circle; // оголошуємо об'єкт структури CircleData

// Викликаємо функцію для перевірки і заповнення вхідних даних
bool hasRealRoots = checkAndFill(1, 2, 1, circle);

// Викликаємо функцію для обробки структури
if (hasRealRoots) {
processCircle(circle);
}

// Виводимо значення вихідних даних
if (hasRealRoots) {
std::cout << "Діаметр окружності: " << circle.diameter << std::endl;
std::cout << "Довжина окружності: " << circle.length << std::endl;
} else {
std::cout << "Рівняння не має дійсних коренів." << std::endl;
```

```
    }

    return 0;
}
```

У цьому коді:

- Створюється структура `CircleData`, яка містить вхідні та вихідні дані задачі.
- Функція `processCircle` обчислює довжину окружності за її діаметром.
- Функція `checkAndFill` перевіряє істинність висловлювання про наявність дійсних коренів квадратного рівняння та, якщо такі є, заповнює діаметр.
- В головній функції `main` створюється об'єкт структури `CircleData`, викликаються функції `checkAndFill` та `processCircle`, а потім виводяться вихідні дані.

Рішення всіх трьох задач реалізувати в одному консольному додатку, *структурувати на модулі.мовою C ++,Використовуючи тип TTime (див. Param71), описати процедуру PrevSec (T) з параметром типу TTime, яка змінює час на -1 секунду (якщо час T є неправильним, то воно не змінюється). Запис T є вхідним і вихідним параметром. Застосувати процедуру PrevSec до п'яти заданих моментів часу.Дано діаметр окружності d. Знайти її довжину L = π · d. Як значення π вважати

рівним 3.14.Дано числа A, B, C (число A не дорівнює 0). Розглянувши дискримінант D = B2

- 4 · A · C, перевірити істинність висловлювання: «Квадратне рівняння A ·

x

2

+

B · x + C = 0 має дійсні коріння».

Щоб структурувати програму на модулі, ми можемо розділити її на три основних модулі: один для операцій з часом (`TimeModule`), один для операцій з окружностями (`CircleModule`) і один для розв'язання квадратного рівняння (`QuadraticEquationModule`). Ось як це можна зробити:

cppCopy code

```
#include <iostream>
#include <cmath>

// Модуль для операцій з часом
namespace TimeModule {
typedef int TTime;

// Процедура зміни часу на -1 секунду
void PrevSec(TTime& time) {
if (time > 0) {
--time;
}
}
}

// Модуль для операцій з окружностями
namespace CircleModule {
struct CircleData {
double diameter; // діаметр окружності
double length; // довжина окружності
};

// Функція обчислення довжини окружності за діаметром
void calculateLength(CircleData& data) {
```

```
    data.length = M_PI * data.diameter;
                }
                }

// Модуль для розв'язання квадратного рівняння
    namespace QuadraticEquationModule {
// Функція перевірки наявності дійсних коренів
                квадратного рівняння
bool hasRealRoots(double A, double B, double C) {
    double discriminant = B * B - 4 * A * C;
            return discriminant >= 0;
                }
                }

                int main() {
            using namespace TimeModule;
        using namespace CircleModule;
    using namespace QuadraticEquationModule;

                // Задані моменти часу
            TTime time1 = 3600; // 1 година
            TTime time2 = 7200; // 2 години
TTime time3 = 86399; // 23 години, 59 хвилин, 59 секунд
              TTime time4 = 0; // Північ
          TTime time5 = -1; // Неправильний час

// Застосовуємо процедуру PrevSec до п'яти заданих
                моментів часу
                PrevSec(time1);
                PrevSec(time2);
                PrevSec(time3);
                PrevSec(time4);
```

```
PrevSec(time5);

// Вивід змінених значень часу
std::cout << "Modified Time 1: " << time1 << std::endl;
std::cout << "Modified Time 2: " << time2 << std::endl;
std::cout << "Modified Time 3: " << time3 << std::endl;
std::cout << "Modified Time 4: " << time4 << std::endl;
std::cout << "Modified Time 5: " << time5 << std::endl;

// Розрахунок довжини окружності
CircleData circle;
circle.diameter = 10; // приклад значення діаметра
calculateLength(circle);
std::cout << "Length of the circle: " << circle.length
<< std::endl;

// Перевірка наявності дійсних коренів квадратного
рівняння
double A = 1, B = 2, C = 1; // приклад коефіцієнтів
квадратного рівняння
if (hasRealRoots(A, B, C)) {
std::cout << "The quadratic equation has real roots." <<
std::endl;
} else {
std::cout << "The quadratic equation does not have real
roots." << std::endl;
}

return 0;
}
```

У цьому коді кожен модуль відповідає за свою частину функціональності: час, окружності і квадратні рівняння. В головній функції `main` ми викликаємо функції з кожного модуля та виводимо результати. Такий підхід робить програму більш структурованою та легше зрозумілою.

ВИСНОВКИ

Під час роботи було вивчено та закріплено на практиці використання модульного підходу до програмування в мові C++. Шляхом реалізації функціональності для роботи з часом, окружностями та квадратними рівняннями в коді програми, були отримані навички організації програмних модулів для кращого управління складністю проекту. Виникло розуміння важливості чіткої структуризації та управління залежностями між функціями і модулями.