

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №1  
з дисципліни «Якість програмного забезпечення та тестування»**

**на тему:**

**«Тестування методами білої скриньки.»**

Виконав: студент гр. ПЗ2011

Кулик С.В.

Прийняла:

Куроп’ятник О.С.

Дніпро, 2024

**Тема:** Тестування методами білої скриньки.

**Мета:** Отримати практичні навички тестування методами білої скриньки.

**Завдання:**Підготувати дві чи більше функцій для тестування. Функції повинні містити не менше шести керуючих конструкцій (розгалужень і циклів). Зазначені структури мають бути вкладеними. Виконати тестування функцій методами білої скриньки.

**Специфікації:**

ФункціяcalculateSalary обчислює загальну заробітну плату протягом року. Якщо працівник відпрацював 10 і менше 20 днів, обчислюється зарплата за формулою zp=zpd\*day+300

Функція countWorkDays підраховує кількість робочих днів протягом місяця. Перебирає кожен день місяця і перевіряє, чи присутній на роботі ( користувач вводить: 1 – якщо був на роботі, 0 – якщо не був ).

**Текст програми:**

#include <iostream>

#include <locale>

using namespace std;

int calculateSalary(int daysWorked[], int dailyRate) {

int totalSalary = 0; //1

for (int i = 0; i < 12; i++) { //2

if (daysWorked[i] >= 10 && daysWorked[i] <= 20) { //3

totalSalary += (dailyRate \* daysWorked[i]) + 300; //4

}

else {

totalSalary += dailyRate \* daysWorked[i]; //5

}

}

return totalSalary; //6

}

int countWorkDays(int daysWorked[]) {

int workDays = 0; //1

for (int i = 0; i < 30; i++) { //2

if (daysWorked[i] == 1 || daysWorked[i]> 1) { //3

workDays++;//4

}

}

return workDays; //5

}

int main() {

setlocale(LC\_CTYPE, "ukr");

int daysWorked[12];

int dailyRate;

cout << "Введiть кiлькiсть вiдпрацьованих днiв за кожен мiсяць:\n";

for (int i = 0; i < 12; i++) {

cout << "Мiсяць " << i + 1 << ": ";

cin >> daysWorked[i];

}

cout << "Введiть заробiтну плату за один день: ";

cin >> dailyRate;

int salary = calculateSalary(daysWorked, dailyRate);

int visitWork[30];

cout << "Введiть 1 - якщо був на роботi, 0 - якщо не був: \n";

for (int i = 0; i < 30; i++) {

cout << "Чи був присутнiй в день: " << i + 1 << ": ";

cin >> visitWork[i];

}

int workDays = countWorkDays(visitWork);

cout << "Загальна заробiтна плата за рiк: " << salary << " грн" << endl;

cout << "Кiлькiсть робочих днiв: " << workDays << endl;

return 0;

}

}

**Тестування**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікувані результати |
| 1 | Визначення зарплати за рік | daysWorked : 15,18,16,18,17,19,14,16,18,18,15 ,19 dailyRate: 450 | calculateSalary: 94 950 |
| 2 | Визначення зарплати за рікІ | daysWorked : 19,20,21,2,25,18,15,16,17,24,21,23 dailyRate: 425 | calculateSalary: 95 725 |
| 3 | Визначення зарплати за рік | daysWorked : 22,23,15,21,7,17,14,19,13,9,10,19 dailyRate: 500 | calculateSalary: 96 600 |

**Тестування**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікувані результати |
| 1 | Визначення робочих днів за місяць | visitWork : 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 | 15 |
| 2 | Визначення робочих днів за місяць | visitWork : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 |
| 3 | Визначення робочих днів за місяць | visitWork : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 | 30 |

**Таблиця покриття операторів**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Оператор  Тест | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | \* | \* | \* | \* |  | \* |
| 2 | \* | \* | \* | \* | \* | \* |
| 3 | \* | \* | \* | \* | \* | \* |

**Таблиця покриття операторів**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Оператор  Тест | 1 | 2 | 3 | 4 | 5 |
| 1 | \* | \* | \* | \* | \* |
| 2 | \* | \* | \* |  | \* |
| 3 | \* | \* | \* | \* | \* |

**Таблиця покриття умов**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Умова  Тест | i < 12 | | daysWorked[i] >= 10 | | daysWorked[i] <= 20 | |
| + | - | + | - | + | - |
| 1 |  | \* | \* |  | \* |  |
| 2 |  | \* | \* |  | \* | \* |
| 3 |  | \* | \* | \* | \* | \* |

**Таблиця покриття умов**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Умова  Тест | i < 30 | | daysWorked[i] == 1 | | daysWorked[i]> 1 | |
| + | - | + | - | + | - |
| 1 | \* | \* | \* | \* |  | \* |
| 2 | \* | \* |  | \* |  | \* |
| 3 | \* | \* | \* |  | \* |  |

**Таблиця покриття рішень**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Умова  Тест | i < 12 | | daysWorked[i] >= 10 && daysWorked[i] <= 20 | |
| + | - | + | - |
| 1 | \* | \* | \* |  |
| 2 | \* | \* | \* | \* |
| 3 | \* | \* | \* | \* |

**Таблиця покриття рішень**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Умова  Тест | i < 30 | | daysWorked[i] == 1 || daysWorked[i]> 1 | |
| + | - | + | - |
| 1 | \* | \* | \* |  |
| 2 | \* | \* |  | \* |
| 3 | \* | \* | \* |  |

**Таблиця комбінаторного покриття умов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Умова  Тест | daysWorked[i] >= 10 && daysWorked[i] <= 20 | | | |
| + + | +- | -+ | -- |
| 1 | \* |  |  |  |
| 2 | \* | \* |  |  |
| 3 | \* | \* | \* |  |

**Таблиця комбінаторного покриття умов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Умова  Тест | daysWorked[i] == 1 || daysWorked[i]> 1 | | | |
| + + | +- | -+ | -- |
| 1 |  | \* |  | \* |
| 2 |  |  |  | \* |
| 3 |  | \* | \* |  |

**Висновок:**

Тестування методами білої скриньки є необхідною складовою процесу розробки програмного забезпечення, оскільки воно дозволяє аналізувати внутрішню структуру програми, включаючи модулі, функції та гілки коду. Основна мета цього типу тестування - переконатися, що програма відповідає вимогам і працює стабільно.

Тестування білої скриньки проводиться паралельно з іншими методами для повноцінної оцінки програмного продукту. Під час цього процесу тестувальники мають можливість оглядати внутрішню структуру програми, що дозволяє їм створювати тестові випадки, базуючись на коді.

В результаті тестування методами білої скриньки можуть бути виявлені різні дефекти, такі як логічні помилки та проблеми з безпекою. Це сприяє підвищенню якості і надійності програмного продукту, що є ключовим завданням у процесі розробки програмного забезпечення.