МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Основи програмної інженерії»**

**на тему: *«Налагодження програм*»**

Виконав: студент гр. ПЗ2011

Кулик Сергій Вадимович

Прийняла: доц. Нежуміра О.І.

Дніпро, 2020

Тема. Налагодження програм.

Мета. Ознайомитися з методами та інструментами налагодження програм та отримати практичні навички їх використання.

**Завдання**

Для поданих функцій розробіть тести та програму-драйвер. Виконайте тестування функцій методом еквівалентного розбиття. Задокументуйте результати тестування.

Знайдіть та виправте помилки. Для кожної помилки визначте та опишіть (вкажіть):

– ситуацію, в якій проявляється. Вкажіть номер тесту, під час якого було виявлено помилку, отримані вихідні дані або опис ситуації (помилка етапу компіляції, зациклення тощо);

– тип: синтаксична, лексична, логічна;

– місце виникнення помилки: рядок програмного коду з оператором/операцією або групою операторів, лексемою (іншим компонентом), що призводить до помилки. Вкажіть засіб, використаний для локалізації: точки зупину, покрокове виконання, друк для налагодження

– можливі дії для усунення (припущення/пропозиції щодо усунення помилки); – виконані дії для усунення.

Відредагуйте код для ліквідації помилки. Редагування не має призводити до появи нових помилок.

Наведіть відредагований код програма з позначеними змінами.

Функції визначаються індивідуально викладачем, який проводить лабораторне заняття.

Варіант 5

|  |  |  |
| --- | --- | --- |
| 5. | /// <summary>  /// В рядку підрахувати кількість символів,  /// які не є символами латинського алфавіту  /// </summary>  /// <param name="str">рядок для перевірки</param>  /// <returns>кількість символів не латинського алфавіту</returns>  int notLatin(const char str)  {  counter = 0;  for (auto index = 0; index <= strlen(str); ++index) {  if (str[index] < "A" &&  (str[index] > "Z" || str[index] < "a") &&  str[index] > "z")  ++counter;  ++index;  }  return index  } | /// <summary>  /// Поміняти місцями елементи в парах,  /// якщо перше число в парі більше другого.  /// Пари: 1 і 2 елемент, 3 і 4 елемент, і так далі.  /// </summary>  /// <param name="\_array">масив цілих чисел</param>  /// <param name="\_size">розмір масиву</param>  int swapInPairs(int\* \_array, int \_size)  {  int index = 0;  while (index < \_size) {  if (\_array[index - 1] > \_array[index])  int tmp = \_array[index];  \_array[index - 1] = \_array[index];  \_array[index - 1] = tmp  }  }  } |

**1. Класи еквівалентності та тестові набори**

Функція 1.

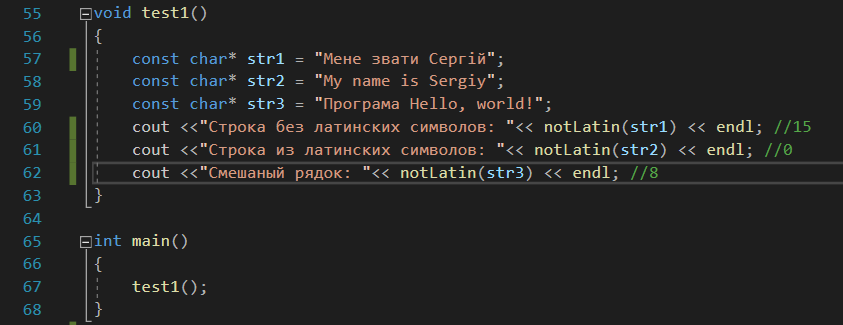
Правильні класи еквівалентності:  
1) Всі символи рядка не є літерами латинського алфавіту.  
2) Деякі символи рядка не є літерами латинського алфавіту.

Неправильні класи еквівалентності:  
1) Всі символи рядка євляються літерами латинського алфавіту.

# Тестові набори даних для функції 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва тесту | Умовне позначення | Вхідні дані | Очікуванні результати |
| 1 | Рядок без латинських символів | str | “Мене звати Сергій” | 15 |
| 2 | Рядок із латинських символів | str | “My name is Sergiy” | 0 |
| 3 | Мішаний рядок | str | “Програма Hello, world!” | 8 |

Програма-драйвер



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № тесту | № рядка з помилкою | Тип помилки | Опис помилки | Засіб локалізації | Можливі дії для усунення | Виконані дії для усунення |
| Помилка компіляції | 10 | Синтаксична | Пропущений символ вказівника в параметрах функції | Компілятор підкреслив червоною лінією | Написати символ вказівника в параметрах | Написав символ вказівника в параметрах |
| Помилка компіляції | 12 | Синтаксична | Не написаний тип даних змінної | Компілятор підкреслив червоною лінією | Написати тип данних int | Написатив тип даних int |
| Помилка компіляції | 14, 15, 16 | Синтаксична | Одиничні символи в подвійних лапках | Компілятор підкреслив червоною лінією | Змінити подвійні лапки на одиничні | Змінив подвійні лапки на одиничні |
| Помилка компіляції | 20 | Логічна | Функція повертає змінну index, а не counter | Компілятор підкреслив червоною лінією | Замінити index на counter | Замінив index на counter |
| Помилка компіляції | 20 | Синтаксична | Пропущена крапка з комою | Компілятор підкреслив червоною лінією | Написати крапку з комою | Написав крапку з комою |
| 1, 2, 3 | 13 | Логічна | В умовах роботи циклу index <= strlen(str) | Покрокове виконання | Прибрати символ ‘=’ | Прибрав символ ‘=’ |
| 1, 2, 3 | 14 | Логічна | Не прописані умови для не ураховування знаків пунктуації в рядку | Побачив, коли вивчав функцію та специфікації | Прописати необхідні умови за допомогою операторів if else | Прописав необхідні умови за допомогою операторів if else |
| 1, 2, 3 | 24, 25, 26 | Логічна | Не правильні умови конструкції if | Покрокове виконання | Замінити оператори логічного ‘і’ на логічне ‘або’ та навпаки | Замінив оператори логічного ‘і’ на логічне ‘або’ та навпаки та прибрав непотрібні лапки |
| 1, 2, 3 | 28 | Логічна | Не потрібний інкремент index | Покрокове виконання | Прибрати інкремент index | Прибрав інкремент index |

Код налагодженої функції

/// <summary>

/// В рядку підрахувати кількість символів,

/// які не є символами латинського алфавіту

/// </summary>

/// <param name="str">рядок для перевірки</param>

/// <returns>кількість символів не латинського алфавіту</returns>

int notLatin(const char\* str)

{

int counter = 0;

for (auto index = 0; index < strlen(str); ++index) {

if (str[index] == ' ' ||

str[index] == '!' ||

str[index] == ',' ||

str[index] == '.' ||

str[index] == '"' ||

str[index] == ':')

{

}

else

{

if (str[index] < 'A' ||

str[index] > 'Z' && str[index] < 'a' ||

str[index] > 'z')

++counter;

//++index;

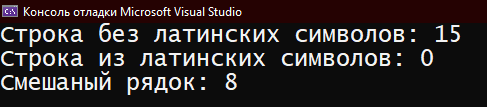
}

}

return counter;

}

Результати тестувань



**2. Класи еквівалентності та тестові набори**

Функція 2.

Правильні класи еквівалентності:

1) Всі перші числа в парах більші за другі.

2) Деякі числа в парах більші за другі

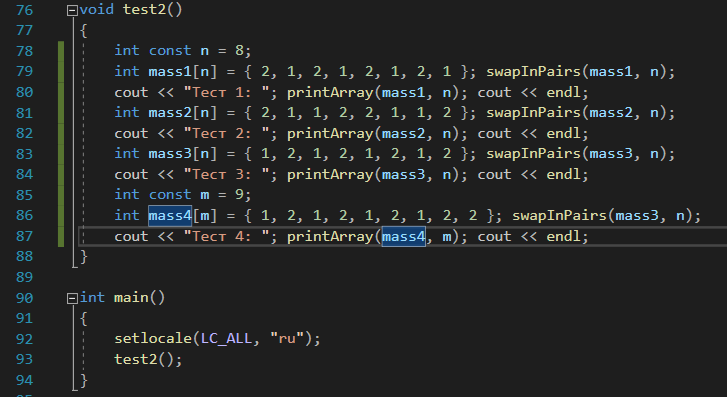
Неправильні класи еквівалентності:

1) Всі перші числа в парах менші за другі.

# Тестові набори даних для функції 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва тесту | Умовне позначення | Вхідні дані | Очікуванні результати |
| 1 | Всі перші числа в парах більші за другі. | \_array | [2, 1, 2, 1, 2, 1, 2, 1] | [1, 2, 1, 2, 1, 2, 1, 2] |
| 2 | Деякі числа в парах більші за другі | \_array | [2, 1, 1, 2, 2, 1, 1, 2] | [1, 2, 1, 2, 1, 2, 1, 2] |
| 3 | Всі перші числа в парах менші за другі. | \_array | [1, 2, 1, 2, 1, 2, 1, 2] | [1, 2, 1, 2, 1, 2, 1, 2] |
| 4 | Не парна кількість елементів | \_array | [2, 1, 2, 1, 2, 1, 2, 1] | [1, 2, 1, 2, 1, 2, 1, 2 , 2] |

Програма-драйвер



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № тесту | № рядка з помилкою | Тип помилки | Опис помилки | Засіб локалізації | Можливі дії для усунення | Виконані дії для усунення |
| Помилка компіляції | 48 | Синтаксична | Пропущена крапка з комою вкінці рядка | Помилка компіляції | Дописати крапку з комою | Дописав крапку з комою |
| Помилка компіляції | 45 | Синтаксична | Пропущена блочна дужка після умовного оператору | Помилка компіляції | Дописати блочну дужку | Дописав блочну дужку |
| Помилка компіляції | 41 | Логічна | Ця функціє не обов'язково має повертати значення | Помилка компіляції | Змінити тип повертаємого значення на void.  Прописати return в кінці | Змінив тип повертаємого значення на void |
| 1, 2, 3, 4 | 50 | Логічна | Не виконується умова виходу із циклу | Покрокове виконання | В кінці циклу прописати  подвійний інкремент index | В кінці циклу прописав інкремент index |
| 1, 2, 3,4 | 45 | Логічна | Не правильна умова  оператору if | Покрокове виконання | Виправити умову | Виправив умову |
| 1, 2, 3,4 | 46, 47,48 | Логічна | Не правильна функція заміни місцями | Покрокове виконання | Виправити функцію | Прописав правильні індекси в функції swap |
| 4 | - | Логічна | При непарній кількості елементів відбувається вихід за межі масиву | Помилка виконання | Прописати перед циклом перевірку на парність | Прописав перед циклом перевірку на парність |

Код налагодженої функції

/// <summary>

/// Поміняти місцями елементи в парах,

/// якщо перше число в парі більше другого.

/// Пари: 1 і 2 елемент, 3 і 4 елемент, і так далі.

/// </summary>

/// <param name="\_array">масив цілих чисел</param>

/// <param name="\_size">розмір масиву</param>

void swapInPairs(int\* \_array, int \_size)

{

int index = 0;

if (\_size % 2 != 0)

{

\_size--;

}

while (index < \_size) {

if (\_array[index] > \_array[index + 1]) {

int tmp = \_array[index];

\_array[index] = \_array[index + 1];

\_array[index + 1] = tmp;

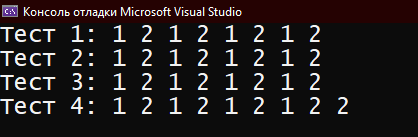
}

index += 2;

}

}

Результати тестувань



**3. Аналіз результатів**

В результаті відлагодження, були виправлені дві функції. Кожна з них успішно пройшла тестування.

**4. Висновки**

Під час виконання лабороторної роботи відлагоджував дві функції. Для кожної з них були розроблені правильні та не правильні класи еквівалентності та тестові набори. Під час відлагодження користувався точками зупину, покроковим виконанням та спостеріганням за контрольними значеннями змінних. Для мене цей метод є найбільш ефективним.