OPERADORES DE MUTAÇÃO PARA A LINGUAGEM C

Os operadores de mutação para programas em linguagem C são transformações específicas aplicadas ao código-fonte para modificar certos elementos com o objetivo de introduzir falhas artificiais. Essas mutações são utilizadas principalmente em **testes de mutação**, uma técnica para avaliar a eficácia de testes de software. Abaixo estão alguns dos operadores de mutação comuns usados em programas C:

---

1. Operadores de Mutação Aritmética

Substituem operadores aritméticos por outros para simular erros de cálculo:

- + substituído por **-**, ***** ou /.
- **-** substituído por +, ***** ou /.
- ***** substituído por / ou **%**.
- / substituído por ***** ou **%**.

---

2. Operadores de Mutação Relacional

Substituem operadores relacionais por outros, simulando erros de comparação:

- < substituído por <=, >, >=, ==, **!=**.
- > substituído por >=, <, <=, ==, **!=**.
- <= substituído por <, >=, >, ==, **!=**.
- >= substituído por >, <=, <, ==, **!=**.
- == substituído por **!=**, <, >, <=, >=.
- **!=** substituído por ==, <, >, <=, >=.

---

3. Operadores de Mutação Lógica

Alteram operadores lógicos para simular erros de lógica:

- **&&** substituído por || ou removido.
- || substituído por **&&** ou removido.
- Negação lógica (**!**) adicionada ou removida.

---

4. Operadores de Mutação de Incremento/Decremento

Substituem ou modificam incrementos e decrementos:

- **i++** substituído por **i--**, **++i**, **--i**, ou removido.
- **i--** substituído por **i++**, **++i**, **--i**, ou removido.

---

5. Operadores de Mutação de Atribuição

Alteram operações de atribuição:

- = substituído por +=, -=, *=, /=, %=.
- += substituído por -=, *=, /=, %=, ou =.

---

6. Operadores de Mutação Condicional

Introduzem alterações em condições:

- Alteração de constantes em expressões condicionais.
- Substituição de expressões booleanas por valores constantes (**true** ou **false**).

---

7. Operadores de Mutação de Pontuação

Afetam manipulação de ponteiros:

- Alterações em operações de **&** (endereço) e ***** (desreferenciamento).
- Substituição de ponteiros por **NULL** ou outros valores.

---

8. Operadores de Mutação de Loop

Alteram estruturas de repetição:

- Substituição de um **for** por um **while** ou vice-versa.
- Remoção ou alteração da condição de parada.
- Incremento modificado dentro de laços.

---

9. Operadores de Mutação de Controle de Fluxo

Alteram o comportamento de controle:

- Substituição de **break** por **continue**.
- Remoção de instruções como **return**, **break**, ou **continue**.

Esses operadores são usados para criar **mutantes**, versões levemente alteradas do programa original. Os testes de mutação avaliam se os casos de teste são capazes de detectar as falhas introduzidas, ajudando a identificar fraquezas nos testes existentes.

Aqui está um exemplo de código em C original e versões mutadas usando alguns operadores de mutação mencionados. O código simula uma verificação simples de paridade e soma números positivos.

```c
#include <stdio.h>

int soma_pares(int arr[], int tamanho) {
    int soma = 0;
    for (int i = 0; i < tamanho; i++) {
        if (arr[i] % 2 == 0) {
            soma += arr[i];
        }
    }
    return soma;
}

int main() {
    int numeros[] = {1, 2, 3, 4, 5};
    int resultado = soma_pares(numeros, 5);
    printf("Soma dos números pares: %d\n", resultado);
    return 0;
}
```

Mutação 1: Substituição de Operador Relacional

Substituímos **% 2 == 0** por **% 2 != 0** para simular um erro na verificação de números pares.

```
c
Copiar código
if (arr[i] % 2 != 0) {
    soma += arr[i];
}
```

Resultado esperado: soma de números ímpares, em vez de pares.

Mutação 2: Alteração de Operador de Atribuição

Substituímos **soma += arr[i]** por **soma -= arr[i]**.

```
c
Copiar código
soma -= arr[i];
```

Resultado esperado: soma incorreta devido à subtração, não adição.

Mutação 3: Alteração de Incremento no Laço

Substituímos **i++** por **i--** para simular um erro no controle do loop.

```
c
Copiar código
for (int i = 0; i < tamanho; i--) {
```

Resultado esperado: loop infinito ou comportamento inesperado.

Mutação 4: Remoção de Controle de Fluxo

Removemos a verificação da condição no **if**, fazendo com que todos os números sejam somados.

```
c
Copiar código
soma += arr[i];
```

Resultado esperado: soma de todos os números, independente da paridade.

Como Avaliar os Testes

Os testes devem ser capazes de identificar todos os mutantes acima. Se algum mutante não for detectado (por exemplo, o teste ainda passa mesmo com o erro), isso indica que os casos de teste precisam ser melhorados.

Por exemplo:

- Um teste que verifica apenas números pares deve falhar para o **Mutante 1**.
- Um teste que verifica valores negativos deve falhar para o **Mutante 2**.
- Um teste que mede desempenho (como o tempo de execução) deve falhar para o **Mutante 3** (loop infinito).

Esse tipo de análise ajuda a aumentar a eficácia da suíte de testes.