<h1> Errores comunes c++ </h1>

Club de programación competitiva universitario Gallos

Cristian Israel Donato Flores

Errores de sintaxis

Revisa cuidadosamente el código en busca de errores tipográficos y verifica que todos los paréntesis, llaves y puntos y comas estén en su lugar.

```
int x = 5
cout << "Hello, world!: " << x << endl;
```

Índices fuera de rango

Asegúrate de que los índices utilizados estén dentro de los límites válidos. Utiliza bucles for con los límites correctos para evitar esto.

```
int arr[3] = {1, 2, 3};
int value = arr[5];  // Acceso a un índice fuera de rango
cout << value << endl;
```

Declaración de variables

Declara todas las variables antes de usarlas y en el ámbito adecuado. Utiliza nombres de variables descriptivos para evitar confusiones.

```
if (true) {
    int x = 10;
}
cout << x << endl;  // x no está en el ámbito correcto
return 0;
```

Desbordamiento o truncamiento

Utiliza tipos de datos apropiados para las operaciones. Si se esperan números grandes, considera el uso de tipos de datos de mayor capacidad.

```
short int num = 32767;
num = num + 1;  // Desbordamiento en short int
cout << num << endl;
```

Comparaciones incorrectas

Verifica que estás utilizando los operadores de comparación (==, !=, <, >, <=, >=) en lugar de operadores de asignación (=).

Olvidar incluir librerías

Asegúrate de incluir todas las librerías necesarias para las funciones y operaciones utilizadas en tu código.

Orden incorrecto de argumentos

Revisa las firmas de las funciones y los operadores que estás utilizando para asegurarte de que los argumentos estén en el orden correcto.

Errores de lógica

Comprende completamente el problema antes de empezar a codificar. Realiza pruebas en papel o en tu mente para verificar la lógica antes de implementarla.

```
int x = 10;
if (x > 5 && x < 8) {  // Condición lógica incorrecta
    cout << "x is between 5 and 8" << endl;
} else {
    cout << "x is not between 5 and 8" << endl;
}
```

Cálculos con punto flotante

Utiliza funciones y técnicas específicas para minimizar los errores de punto flotante, como redondeo y comparaciones aproximadas.

Cadena de formato incorrecta

Asegúrate de usar las cadenas de formato correctas para las funciones de entrada/salida, como cout y cin.

Uso incorrecto de estructuras de datos

Aprende cómo funcionan y cómo se utilizan las estructuras de datos antes de aplicarlas en tu código. Practica su uso en ejemplos simples.

```
int[] v = {1, 2, 3};
cout << v[0].size() << endl;  // Uso incorrecto de vector
```

Falta de optimización

Optimiza tu código mediante técnicas como la poda en algoritmos de búsqueda, uso eficiente de estructuras de datos y minimización de bucles anidados.

No leer las restricciones

Lee detenidamente las restricciones del problema y considera cómo afectarán tu enfoque y solución. Asegúrate de cumplir con los límites de tiempo y memoria.

Copiar y pegar errores

Si copias código de internet, asegúrate de entenderlo por completo y modifícalo según las necesidades de tu problema. Evita el copiar y pegar ciegamente.

Uso de variables no inicializadas

Siempre asegúrate de inicializar tus variables antes de utilizarlas para evitar comportamientos indefinidos.

```
int x;
cout << x << endl;  // Uso de variable no inicializada
```

Operaciones entre tipos incompatibles

Verifica que los tipos de datos involucrados en operaciones sean compatibles, por ejemplo, no realizar operaciones aritméticas con punteros.

```
int x = 5;
   char c = 'A';
   int result = x + c;  // Operación entre tipos incompatibles
   cout << result << endl;
```

Falta de manejo de errores

No verificar si las operaciones como la apertura de archivos o asignación de memoria han tenido éxito puede llevar a problemas difíciles de depurar.

Manipulación incorrecta de cadenas

Las cadenas en C++ son matrices de caracteres, por lo que asegúrate de trabajar dentro de los límites válidos y utilizar funciones de manejo de cadenas adecuadas.

Conversiones incorrectas de tipos de datos

Realizar conversiones entre tipos de datos de manera incorrecta puede llevar a pérdida de información o resultados incorrectos.

Falta de control de flujo

No usar declaraciones break o return correctamente en bucles o funciones puede causar un comportamiento inesperado.

Errores en expresiones booleanas

Verifica las condiciones en tus declaraciones if, while y for para asegurarte de que estén evaluando las condiciones de manera adecuada.

Uso incorrecto de funciones recursivas

Asegúrate de que las funciones recursivas tengan casos base y condiciones de término para evitar bucles infinitos.

```
int factorial(int n) {
    return n * factorial(n - 1);  // Falta caso base
}

int main() {
    int n = 5;
    int result = factorial(n);
    cout << "Factorial of " << n << " is " << result << endl;
    return 0;
}
```

Ignorar advertencias del compilador

Presta atención a las advertencias del compilador, ya que pueden señalar problemas potenciales en tu código.

Uso incorrecto de constantes y variables globales

Evita la sobreutilización de variables globales, ya que pueden dificultar el seguimiento y la depuración del código.

```
int globalVar = 10;  // Variable global

int main() {
    const int constVar = 5;  // Constante local
    globalVar = 20;  // Modificación de variable global
    constVar = 10;   // Error: No se puede modificar constante
    cout << globalVar << " " << constVar << endl;
    return 0;
}
```

No limpiar el estado del flujo de entrada/salida

Después de leer o escribir en flujos de entrada/salida, asegúrate de restablecer su estado para evitar problemas en operaciones posteriores.