Ayudantía 2

Carlos Lagos - carlos.lagosc@usm.cl

Créditos para Sebastián Torrealba

Errores de desbordamiento

Existen dos tipos de desbordamiento en numeros enteros, `overflow` y `underflow`.

- El `overflow` ocurre cuando el valor resultante de una operación aritmética es mayor que el valor máximo que puede almacenar un tipo de dato.
- El `underflow` ocurre cuando el valor resultante de una operación aritmética es menor que el valor mínimo que puede almacenar un tipo de dato.

Errores de desbordamiento

¿Cuál sería el resultado de la siguiente operación en un entero de 32 bits?

```
int a = INT_MAX; // 2147483647
int b = a + 1; // overflow
```

Fuera de rango

El término fuera de rango o out of bounds se refiere a una situación en la programación donde se intenta acceder a una posición de un array o estructura de datos que está fuera de los límites definidos por su tamaño.

Desbordamiento de pila (stack overflow)

El desbordamiento de pila ocurre cuando una función se llama a sí misma recursivamente sin una condición de término adecuada, agotando el espacio disponible en la pila.

Errores de precisión de punto flotante

Los errores de precisión de punto flotante son problemas que surgen debido a la forma en que los números en punto flotante se representan y manipulan en la memoria de una computadora. Los números en punto flotante se representan con un número finito de bits, lo que significa que no todos los números pueden ser representados con precisión.

Errores de precisión de punto flotante

¿Qué imprime?

```
cout << 1e300 + 1e300 << endl;
```

¿Qué pasará con lo siguiente?

```
cout << ceil((0.1 + 0.2) * 10.0) << endl;
// cout << setprecision(20) << fixed << (0.1 + 0.2) * 10.0 << endl;
```

Errores de precisión de punto flotante

¿Qué alternativa existe para calcular $\left\lceil \frac{a}{b} \right\rceil$, considerando que a y b son enteros, sin tener error de precisión?

```
int techo(int a, int b) {
    if (a % b == 0) return a / b;
    return a / b + 1;
}
```

Profiling

Profiling es el proceso de medir y analizar el rendimiento de un programa para identificar cuellos de botella y optimizar su eficiencia. Se enfoca en aspectos como el tiempo de ejecución, el uso de memoria y la frecuencia de llamadas a funciones, proporcionando información valiosa sobre cómo mejorar el código y reducir los tiempos de ejecución.

Profiling

gprof

`gprof` es una herramienta de profiling para programas en C, C++, y Fortran que permite analizar el rendimiento de aplicaciones. Utiliza un enfoque basado en muestreo para medir el tiempo de ejecución y la frecuencia de llamada de las funciones en un programa.

Para usar `gprof`, primero debes compilar tu programa con la opción `-pg` y luego ejecutar el programa. Después, se genera un archivo de perfil (`gmon.out`) que se puede analizar con `gprof` para obtener un informe detallado.

```
g++ -pg -o mi_programa mi_programa.cpp
./mi_programa
gprof mi_programa gmon.out > perfil.txt
```

GNU Debugger

Es una herramienta de depuración que permite a los programadores ejecutar, detener y analizar el comportamiento de programas en C, C++, y otros lenguajes. Facilita la identificación y corrección de errores al inspeccionar variables, rastrear la ejecución, y observar la memoria en tiempo real.

GNU Debugger

gdb

`gdb` es la implementación de GNU Debugger que proporciona un conjunto de comandos para el análisis detallado de programas.

Para usar `gdb` con un programa, debes compilar el código con la opción `-g`, que incluye información de depuración en el archivo ejecutable. Esto permite que `gdb` tenga acceso a la información necesaria para realizar un análisis detallado del programa.

Para usar `gdb`, compila el programa con la opción `-g`:

```
g++ -g -o mi_programa mi_programa.cpp
```

GNU Debugger

gdb

```
# Establece un punto de interrupción en la función o línea especificada.
break <función/linea>
# Inicia la ejecución del programa.
run
# Imprime el valor de una variable.
print <variable>
# Avanza al siguiente comando en la misma función.
next o n
# Entra en la función llamada en el punto actual.
step o s
# Reanuda la ejecución hasta el siguiente punto de interrupción.
continue o c
```

GNU Debugger

gdb

```
# Ejecuta hasta el final de la función actual y luego detiene la ejecución.
finish
# Muestra información sobre el estado actual del programa (e.g., info locals, info registers).
info <comando>
# Muestra el historial de llamadas de la pila.
backtrace o bt
 # Muestra el código fuente alrededor de la línea actual.
list o l
# Establece un punto de vigilancia que se activa cuando el valor de la variable cambia.
watch <variable>
# Modifica el valor de una variable en tiempo de ejecución.
set <variable>=<valor>
# Sale de gdb.
quit o q
```

Valgrind

Valgrind es una herramienta de análisis de programas que ayuda a detectar errores de memoria, fugas de memoria, y problemas de concurrencia en aplicaciones escritas en C, C++, y otros lenguajes. Es ampliamente utilizada para garantizar la correcta gestión de la memoria y mejorar la estabilidad del software.

Benchmarking

Benchmarking es el proceso de medir y comparar el rendimiento de diferentes algoritmos o implementaciones para determinar cuál es más eficiente en términos de tiempo de ejecución y/o uso de recursos.

Benchmarking

- **Tiempos de ejecución:** Medir el tiempo que tarda un algoritmo en completarse bajo diferentes condiciones.
- **Comparación de algoritmos:** Ejecutar múltiples algoritmos para resolver el mismo problema y comparar sus tiempos de ejecución.
- **Escalabilidad:** Evaluar cómo se comporta el rendimiento del algoritmo cuando se incrementa el tamaño de los datos de entrada.
- **Pruebas de estrés:** Ejecutar el algoritmo bajo condiciones extremas para identificar cuellos de botella y puntos de fallo.

`time` en Linux

El comando `time` en Linux se usa para medir el tiempo total de ejecución de un programa. Muestra los siguientes tiempos:

- **Tiempo real:** Tiempo total desde el inicio hasta el fin de la ejecución.
- **Tiempo de usuario:** Tiempo que el CPU pasa ejecutando el código del programa.
- **Tiempo de sistema:** Tiempo que el CPU pasa ejecutando operaciones del sistema en nombre del programa.

`chrono` en C++

La librería `chrono` en C++ permite realizar mediciones de tiempo con alta precisión.

- **`high_resolution_clock`:** Para mediciones precisas en nanosegundos.
- **`steady_clock`:** Un reloj que no se ajusta ni cambia, útil para mediciones consistentes.
- **`duration_cast`:** Para convertir las duraciones de tiempo en diferentes unidades (milisegundos, segundos, etc.).

Ejemplo:

```
auto start = chrono::high_resolution_clock::now();
// código a medir
auto end = chrono::high_resolution_clock::now();
```

FIN