Pontificia Universidad Javeriana

Taller 01

Juan Guillermo Pabon Vargas

Estructuras de datos

John Corredor Franco

09 Marzo, 2025

Indice

1. Resumen

El punto de este taller era realizar un acercamiento inicial al proceso de compilación y depuración de programas en línea de comandos. Además de esto el taller sirvió como una oportunidad para aprender cómo definir un plan de pruebas. Todo esto se hizo probando diferentes comandos que fueron mencionados en el enunciado y documentando los resultados después de ejecutar los diferentes comandos en el terminal de Linux.

2. Introduccion

En este taller se realizará un acercamiento inicial al proceso de compilación y depuración de programas en línea de comandos. También se va definir un plan de pruebas simple para las operaciones a realizar.

3. Desarrollo

3.1 Actividad 1:

En la actividad 1, se compilo y ejecuto directamente el programa “Excercise1.cpp”. Se usó los comandos “g++ -std=c++11 -o exercise1 excercise1.cpp” Para compilar el programa y “./exercise1” para ejecutar el programa.

3.2 Actividad 2:

Se compiló “excercise2.cxx”, y despues se ejecutó el programa por medio del depurador. Se uso los comandos “g++ -std=c++11 -g -o excercise2 excercise2.cxx” para compilar el programa, y “gdb exercise2” para ejecutarlo por medio del depurador. Se Introdujeron unos datos de ejemplo para documentar los resultados

3.3 Actividad 3:

Se ejecutó el programa “excercise2.cxx” varias veces. Cada vez se usaron los diferentes datos de entrada que ofreció el enunciado para ver, documentar y comparar los resultados con lo que debería ser.

4. Resultados

4.1 Actividad 1:

```
~$ g++ -std=c++11 -o exercise1 exercise1.cpp
~$ ./exercise1
Creating Node, 1 are in existence right now
Creating Node, 2 are in existence right now
Creating Node, 3 are in existence right now
Creating Node, 4 are in existence right now
The fully created list is:
4
3
2
1

Now removing elements:
Creating Node, 5 are in existence right now
Destroying Node, 4 are in existence right now
4
3
2
1

Segmentation fault (core dumped)
~$
```

Imagen 1, Resultado de la compilación y ejecución del programa “excercise1.cpp”

Analisis:

En la actividad 1, se compilo y ejecuto directamente el programa “excercise1.cpp”. Después de ejecutarlo, se puede ver que se crearon 4 nodos y se agregaron a la lista enlazada. El resultado muestra un Segmentation fault porque hay un error en la función para remover nodos donde crea un nuevo nodo envés de apuntar al siguiente nodo.

4.2 Actividad 2:

```
~$ g++ -std=c++11 -g -o exercise2 exercise2.cxx
~$ gdb exercise2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from exercise2...
(gdb) run
Starting program: /home/user/exercise2
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Ingrese coordenada X de la posicion del rectangulo: 0
Ingrese coordenada Y de la posicion del rectangulo: 0

Perimetro del rectangulo: 8
Area del rectangulo: 6
Distancia del rectangulo al origen de coordenadas: 0
[Inferior 1 (process 354) exited normally]
(gdb) backtrace
No stack.
(gdb) exit
```

Imagen 2, compilacion y ejecucion por medio del depurador el programa “excercise2.cxx”

Analisis:

Después de compilar “excercise2.cxx”, se ejecutó el programa por medio del Depurador. Ya después de ingresar la coordenada X y la coordenada Y del rectángulo, y las dimensiones del rectángulo, se imprimió el perímetro, área y la distancia del rectángulo al origen. Como se puede ver los resultados son incorrectos porque las ecuaciones usadas en los métodos no tienen paréntesis para separar las operaciones y también no están hechas correctamente. Sin esos paréntesis, se hacen las operaciones de una forma diferente a lo que queremos. Esto solo aplica para las funciones de calcular el área y el perimetro del rectangulo

4.3 Actividad 3:

Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
1: Alto como el doble de Ancho	Ancho = 2 Alto = 4	Perimetro: 12	Perimetro: 8
2: Alto igual a Ancho	Ancho = 3 Ancho = 3	Perimetro: 12	Perimetro: 9
3: un número en cero	Ancho = 5 Alto = 0	Perimetro: error	Perimetro: 10

Tabla 1, Plan de pruebas: función perímetro del rectángulo

Analisis:

Como no hay paréntesis alrededor de la suma entre el ancho y la altura del rectángulo, se multiplica el ancho del rectángulo primero por 2, y después se le agrega la altura.

Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
1: Alto como el doble de Ancho	Ancho = 2 Alto = 4	Area: 8	Area: 6
2: Alto igual a Ancho	Ancho = 3 Ancho = 3	Area: 9	Area: 6
3: un número en cero	Ancho = 5 Alto = 0	Area: 0	Area: 5

Tabla 2, Plan de pruebas: función Área del rectángulo

Analisis:

El área de un rectángulo se encuentra cuando se multiplica el ancho con la Altura. En la función, envés de hacer esto se suman los valores. Por esa razón, no se obtienen los resultados que deberían ser.

Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
1: Números positivos	X = 15 Y = 32	35.34	35.3412
2: Alto igual a Ancho	X = 0 Y = 3	32	32
3: un número en cero	X =15 Y = 15	21.21	21.2132

Tabla 3, Plan de pruebas: función Distancia del rectángulo

Analisis:

Los resultados obtenidos son similares a lo que esperábamos. Esta es la única función que si funciona como debería.

¿Cuáles funciones presentan errores en sus resultados?

Las funciones del perímetro y del área del rectángulo están presentando errores en sus resultados. Esto es porque no usan paréntesis para separar la ecuación y también porque la ecuación fue escrita incorrectamente.

5. Directorio

Para confirmar la existencia de todos los archivos, se muestra una imagen del directorio con los archivos adentro.

```
~$ cd Taller001
~/Taller001$ ls
exercise1      exercise1.o  exercise2.cxx  rectangle.h
exercise1.cpp  exercise2    rectangle.cxx
~/Taller001$
```

Imagen 3, Directorio del taller

6. Conclusion

Este taller sirvió como una forma de ver cómo compilar y depurar programas por línea de comandos. También se aprendió cómo se realizan los plan de pruebas para probar el funcionamiento del programa y comparar los resultados con los resultados esperados. Gracias a estos planes de pruebas, se pudo identificar que las funciones para calcular el perímetro y el área del rectángulo están mal hechas.

Estas dos funciones tenían errores en sus ecuaciones para calcular los valores. En el del perímetro, no se incluyeron paréntesis alrededor de la suma del ancho y de la altura para multiplicar ese resultado por 2 para obtener el resultado esperado. En la función de calcular el área, se sumaron el alto y el ancho a través de multiplicarlos para obtener los valores esperados.

Es gracias al plan de pruebas que se puede identificar los errores en los programas que están relacionados a la lógica implementada y no de un error que saque después de compilar.