



Taller 01 Plan de pruebas

Juan Sebastián Rodríguez Pabon

David Rodríguez Jurado

Julián Pérez Gomes

Daniel Galviz

Dirigido a: John Corredor

28 de agosto del 2024

Bogotá D.C

Introducción

La compilación en g++ es esencial en la programación debido a que permite observar mediante consola todas las operaciones dentro del programa, permite documentar de mejor manera el programa en C++ debido a que está diseñado especialmente para ejecutar sus respectivos programas. En la elaboración del taller se realizará una detallada documentación acerca de la compilación en g++ de un código enfocado en Nodos, cada uno cumpliendo una función esencial en el programa.

En la compilación se compilarán archivos con extensiones *.c, *.cpp, *.cxx debido al formato de estos mismos, es decir que se puede realizar su compilación en g++ mediante una serie de comandos que se presentarán a continuación.

Ejecución

Se realizará la ejecución por partes, a medida de la implementación de comandos, finalmente se mostrará el resultado final.

En la ejecución del programa, se usa el comando `ls` este comando permite visualizar archivos y directorios en el directorio base actual.

En la imagen se encuentra “ArchivoTaller01.zip” debido a que en el directorio base actual es una máquina virtual la cual aún no se han creado más archivos y “ArchivoTaller01.zip” es el único archivo creado hasta el momento. Esta función permite visualizar los archivos guardados en nuestro programa.

```
estudiante@ing-gen261:~/Downloads$ ls
ArchivoTaller01.zip  exercise1.cpp  Taller01
```

Una vez se haya guardado el archivo .cpp se usa el comando `g++` seguido del nombre del programa anteriormente guardado. La función `g++` se implementa con su objetivo de compilar el archivo guardado, compilar se refiere a ejecutar y verificar su correcto funcionamiento, en caso de contener errores se deberán de corregir y repetir el proceso.

Otra función `./a.out` también se implementa para compilar el archivo guardado, si el archivo no contiene errores se podrá ejecutar correctamente.

En la siguiente imagen se evidencia la compilación mediante el comando `g++` explicado anteriormente. En la compilación se evidencia que en su ejecución se crearon listas enlazadas, añadiendo y creando Nodes uno por uno: Node 1. Node 2, Node 3 y Node 4. Después de la aquella creación se eliminan Nodes, eliminando el Node 4, este se elimina con el propósito de seguir con el propósito de una pila, ya que en una pila el ultimo nodo añadido es el primero en ser eliminado

```
estudiante@ing-gen261:~/Downloads$ g++ exercise1.c
estudiante@ing-gen261:~/Downloads$ ./a.out
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)
estudiante@ing-gen261:~/Downloads$
```

Se evidencia que el archivo se ejecutó correctamente y hace referencia a 4 nodos los cuales se conforma por una lista enlazada genérica, con la capacidad de insertar y eliminar nodos.

Al juntar los comandos mencionados anteriormente, el programa tendrá la siguiente estructura:

```
estudiante@ing-gen261:~/Downloads$ ls
ArchivoTaller01.zip exercise1.cpp Taller01
estudiante@ing-gen261:~/Downloads$ g++ exercise1.cpp
estudiante@ing-gen261:~/Downloads$ ./a.out
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)
estudiante@ing-gen261:~/Downloads$
```

A continuación, se realizará la ejecución del ejercicio 2 ejecutando el archivo llamado rectangle.h. Para su correcta compilación se realizará con exactitud el proceso mencionado anteriormente.

En la ejecución del programa se usa el comando **ls** este comando nos permite visualizar archivos y directorios en el directorio base actual.

En la imagen se encuentra el archivo “ArchivoTaller01.zip” en la carpeta llamada “exercise2.cxx” y el programa “rectangle.h” el cual es programa para ejecutar.

```
estudiante@ing-gen261:~/Downloads$ ls
ArchivoTaller01.zip  exercise2.cxx  rectangle.h
exercise1.cpp       rectangle.cxx  Taller01
```

En la siguiente imagen se ejecuta el archivo “exercise2.cxx” por medio del comando **g++** seguido del nombre del programa anteriormente guardado. La función **g++** se implementa con su objetivo de compilar el archivo guardado, compilar se refiere a ejecutar y verificar su correcto funcionamiento, en caso de contener errores se deberán de corregir y repetir el proceso.

El comando **-g** se utiliza para incluir información de depuración en el archivo ejecutable

Generado, depurar hace referencia a diagnosticar fu funcionalidad general y corregir errores.

Se usa el comando **gdb** y **a.out** para depurar y después ejecutar el programa.

Al ejecutar el programa se muestran propiedades del software y sus respectivas configuraciones internas implementadas para poder ejecutar el código y después lo importante es que se puede evidenciar propiedades de un rectángulo, es decir las entradas del código; coordenadas (X,Y)

Ancho y altura. Gracias a estos datos se pueden realizar las operaciones y funciones como lo son:

Perímetro del rectángulo, Área del rectángulo, distancia del rectángulo al origen de coordenadas. El

programa no detecta errores y con todos sus comandos mencionados anteriormente quedaría así:

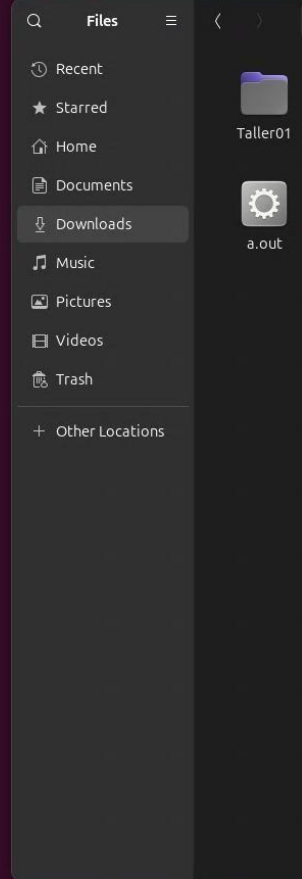
Estructura de datos

```
estudiante@ing-gen261:~/Download$ ls
ArchivoTaller01.zip  exercise2.cxx  rectangle.h
exercise1.cpp        rectangle.cxx  Taller01
estudiante@ing-gen261:~/Download$ g++ exercise2.cxx
estudiante@ing-gen261:~/Download$ g++ -g exercise2.cxx
estudiante@ing-gen261:~/Download$ gdb a.out
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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 a.out...
(gdb) run
Starting program: /home/estudiante/Downloads/a.out

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[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 posición del rectángulo: 8
Ingrese coordenada Y de la posición del rectángulo: 5
Ingrese ancho del rectángulo: 9
Ingrese alto del rectángulo: 10

Perímetro del rectángulo: 28
Área del rectángulo: 19
Distancia del rectángulo al origen de coordenadas: 9.43398
[Inferior 1 (process 64481) exited normally]
(gdb) quit
estudiante@ing-gen261:~/Download$
```



Plan de pruebas

De acuerdo con el programa se buscar realizar una prueba para cada función, la cual se busca un resultado esperado. Se realizarán las pruebas para: Perímetro del rectángulo, área del rectángulo y distancia del rectángulo al origen.

Plan de pruebas: función Perímetro del rectángulo			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
1: Alto como el doble de Ancho	Ancho = 2, Alto = 4	12	12
2: Alto igual a Ancho	Ancho = 3, Alto = 3	12	12
3: un numero en cero	Ancho = 5, Alto = 0	10	10

Plan de pruebas: función Área del rectángulo			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
1: Alto como el doble de Ancho	Ancho = 2, Alto = 4	8	8
2: Alto igual a Ancho	Ancho = 3, Alto = 3	9	9
3: un numero en cero	Ancho = 5, Alto = 0	0	0

Plan de pruebas: función Distancia del rectángulo al origen			
Descripción de caso	Valores de entrada	Resultado esperado	Resultado obtenido
1: números positivos	$x = 15, y = 32$	35.34	35.34
2: un número 0	$x = 0, y = 32$	32	32
3: números iguales	$x = 15, x = 15$	21.21	21.21

Realizando las operaciones, se comprueba que los resultados esperados son los resultados obtenidos, sin embargo, hay discrepancia porque se debería de cambiar la inicialización de variable `int` debido a que este solo recibe valores enteros, no recibe valores flotantes, si se realizara este cambio, los resultados no generarían problema.

Conclusión

La correcta ejecución del programa utilizando el compilador g++, se pudo comprobar la funcionalidad de un sistema compuesto por dos archivos de código, los cuales implementan diversas funciones orientadas a estructuras lineales. Estas funciones demostraron un manejo eficiente de operaciones, como el cálculo del perímetro, el área y la distancia de un rectángulo al origen, utilizando las estructuras lineales definidas en el código. Este proceso validó el correcto funcionamiento de las funciones implementadas en un entorno C++.