EIE
Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

`IE-0217` Estructuras abstractas de datos y algorítmos para ingeniería

Tarea3: Pilas y colas C++

Timna Belinda Brown Ramírez

B61254

`timna.brown@ucr.ac.cr`

`belindabrownr@gmail.com`

I-2019

Tabla de contenidos

---

1. Enunciado

Utilizando las implementación de pila y cola en `https://app.schoology.com/system/files/attachments/files/m/201906/course/1995876103/stack_queue_5d093ab61250e.zip`, escriba dos programas que:

1. dada una hilera de caracteres que contenga pares de paréntesis: (), [], , <>, y letras, sea capaz de decir si dicha hilera está bien escrita en el sentido de los pares de paréntesis; o sea, que siempre que se abra un paréntesis, se cierre antes de cerrar uno de otro tipo, por ejemplo:

```
# ./a.out 8jsle(fasd)asd fds{aefs[fde]}fd
hilera valida
```

```
# ./a.out 8jsle(fasd)asd fds{aefs[fde))>fd
hilera invalida
```

2. dada una cantidad de colas y una proporción de prioridad, construir un sistema de colas de prioridad, donde de manera aleatoria, se agreguen elementos a las colas y basada con la proporción de prioridades, se eliminen los elementos del sistema. Por ejemplo:

```
# ./a.out 3 3:2:1
cola1: 1, 2, 3, 15
cola2: 4, 5, 6, 13, 14
cola3: 7, 8, 9, 10, 11, 12

salidas: 1, 2, 3, 4, 5, 7, 15, 6, 13, 8, 14, 9, 10, 11, 12
```

```
# ./a.out 4 1:1:2:1
cola1: 1, 2, 3, 15
cola2: 4, 5, 6, 13
cola3: 7, 8, 9, 10, 11, 12
cola4: 16, 17, 18, 19

salidas: 1, 4, 7, 8, 16, 2, 5, 9, 10, 17, 3, 6, 11, 12, 18, 15, 13, 19
```

Realice un programa de pruebas para sus 2 programas.

2. Consideraciones

- Trabajo individual
- Genere un reporte en LaTeX que incluya al menos el enunciado, la solución propuesta.
- Cada estudiante debe subir el reporte a Schoology.
- Recuerde que por cada día tardío de entrega se le rebajaran puntos de acuerdo con la formula: 3^d, donde $d > 1$ es la cantidad de días naturales tardíos.

3. Abordaje y conclusiones

Para la resolución del laboratorio presentado, se realizaron una serie de clases y funciones que cumplen con los objetivos de la tarea 3.

Como conclusión se puso en práctica el uso del lenguaje C++, además, del uso de la lógica para cumplir el objetivo planteado.[2]

4. Apéndice

4.1. Código fuente

[1]

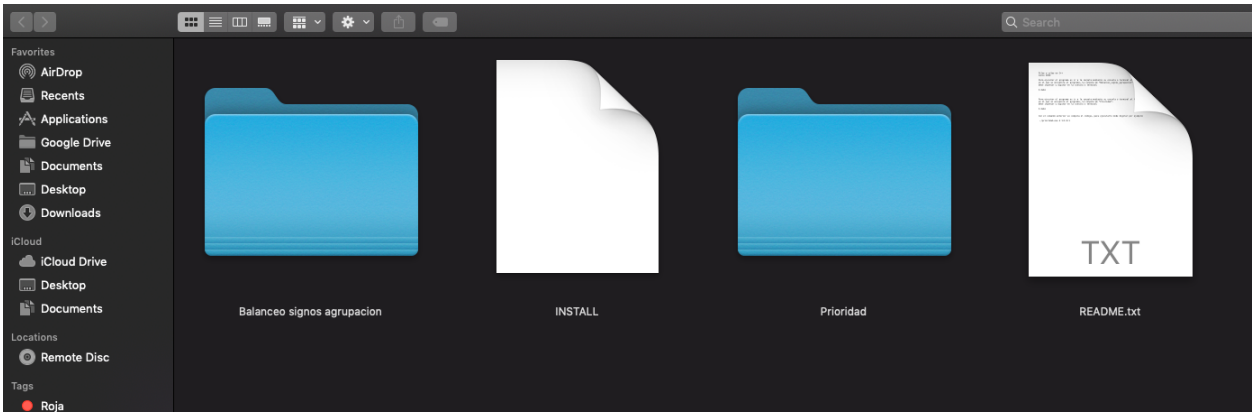

Figura 1: Dentro de Tarea3_Estructuras

Todos tienen en común lo siguiente:

```
Pilas y colas en C++
Junio 2019

Para ejecutar el programa es ir a la carpeta mediante su consola o terminal al folder
en el que se encuentra el programa, la carpeta de "Balanceo_signos_agrupacion".
Debe ingresar y digitar en la consola o terminal:

$ make



Para ejecutar el programa es ir a la carpeta mediante su consola o terminal al folder
en el que se encuentra el programa, la carpeta de "Prioridad".
Debe ingresar y digitar en la consola o terminal:

$ make


Con el comando anterior se compila el código, para ejecutarlo debe digitar por ejemplo:

 ./prioridad.exe 4 1:1:2:1
```

Figura 2: Readme

```
Pilas y colas usando Stack en C++

License Apache 2.0


Se distribuye un Makefile con 3 reglas:

    * build: compila los fuentes.
    * clean: borra los binarios.
```

Figura 3: Install

```
all: build run clean

build:
	g++ -g --std=c++11 -Wall balanceo_signos_agrupaci.cpp -o balance_signos.exe

run:
	./balance_signos.exe

clean:
	rm *.exe
	rm -rf *.exe.dSYM
```

Figura 4: Makefile de Balance

```
all: build

build:
	g++ -g --std=c++11 -Wall prioridad_tails.cpp -o prioridad.exe



clean:
	rm *.exe
	rm -rf *.exe.dSYM
```

Figura 5: Makefile de Prioridad

header.h

```
1   #ifndef HEADER_H
2   #define HEADER_H
3
4
5   #include <vector>
6   #include <iostream>
7   #include "Element.h"
8   #include "Stack.h"
9   #include "Queue.h"
10  #include <cstring>
11  #include<string>
12
13
14
15  #endif
16
```

Figura 6: Código de header.h

balanceo_signos_agrupaci.cpp

```cpp

#include "header.h"

using namespace std;

bool Pares(char opening,char closing)
{
if(opening == '(' && closing == ')') return true;
else if(opening == '{' && closing == '}') return true;
else if(opening == '[' && closing == ']') return true;
else if(opening == '<' && closing == '>') return true;
return false;
}
bool Parentesis_balanceados(string exp)
{
  Stack<char> S;
for(int i =0;i<exp.length();i++)
  {
if(exp[i] == '(' || exp[i] == '{' || exp[i] == '[' || exp[i] == '<')
      S.push(exp[i]);
else if(exp[i] == ')' || exp[i] == '}' || exp[i] == ']' || exp[i] == '>')
    {
if(S.empty() || !Pares(S.pop(),exp[i]))
return false;

else
        S.pop();
    }
  }
return S.empty() ? true:false;
}
```

~/Desktop/Tarea3_Estructuras/Balanceo signos agrupacion/balanceo_signos_agrupaci.cpp 22:4

Figura 7: Código de balance_signos_agrupaci.cpp

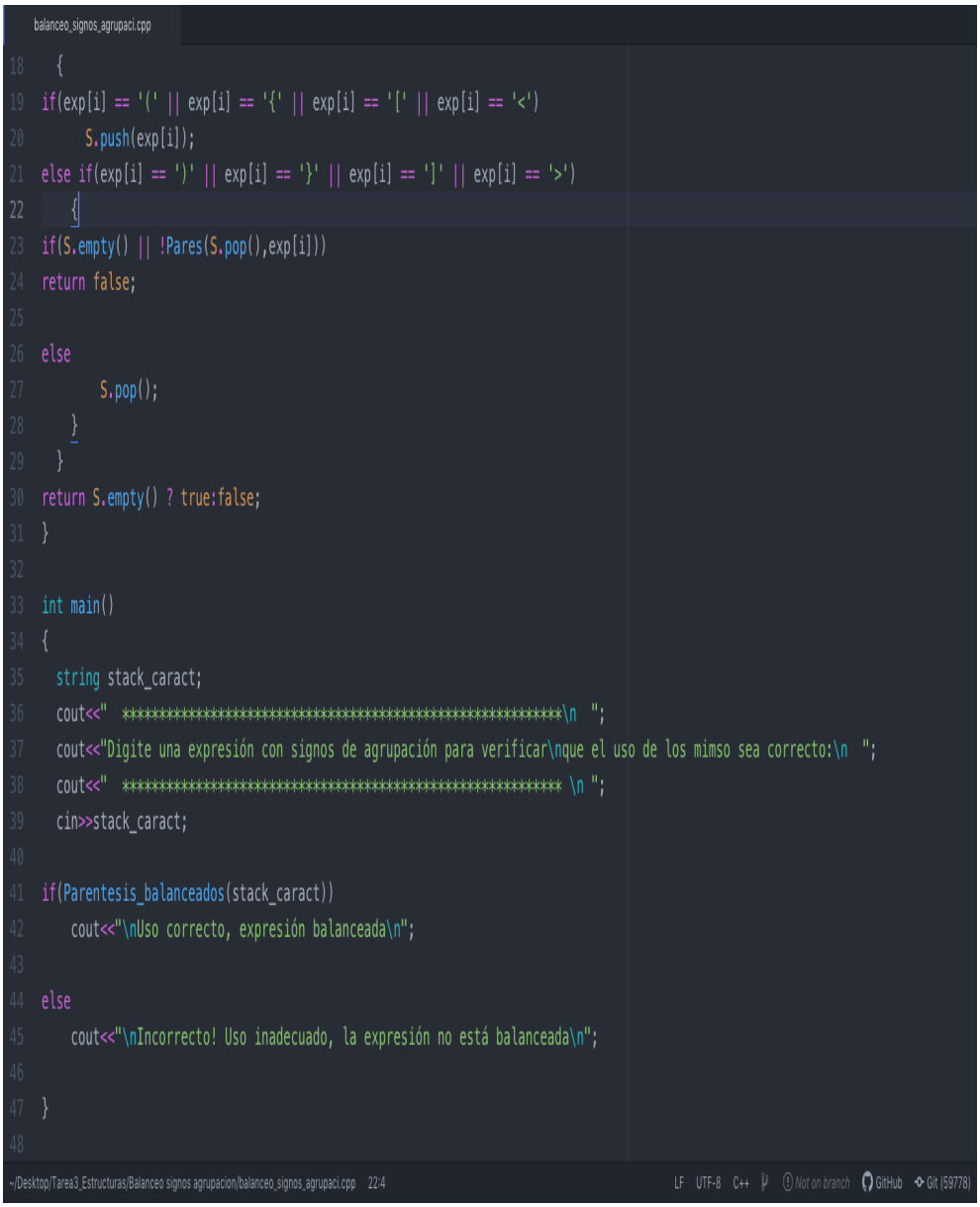

Figura 8: Código de balance_signos_agrupaci.cpp

Figura 9: Resultados de Balance de signos de agrupación

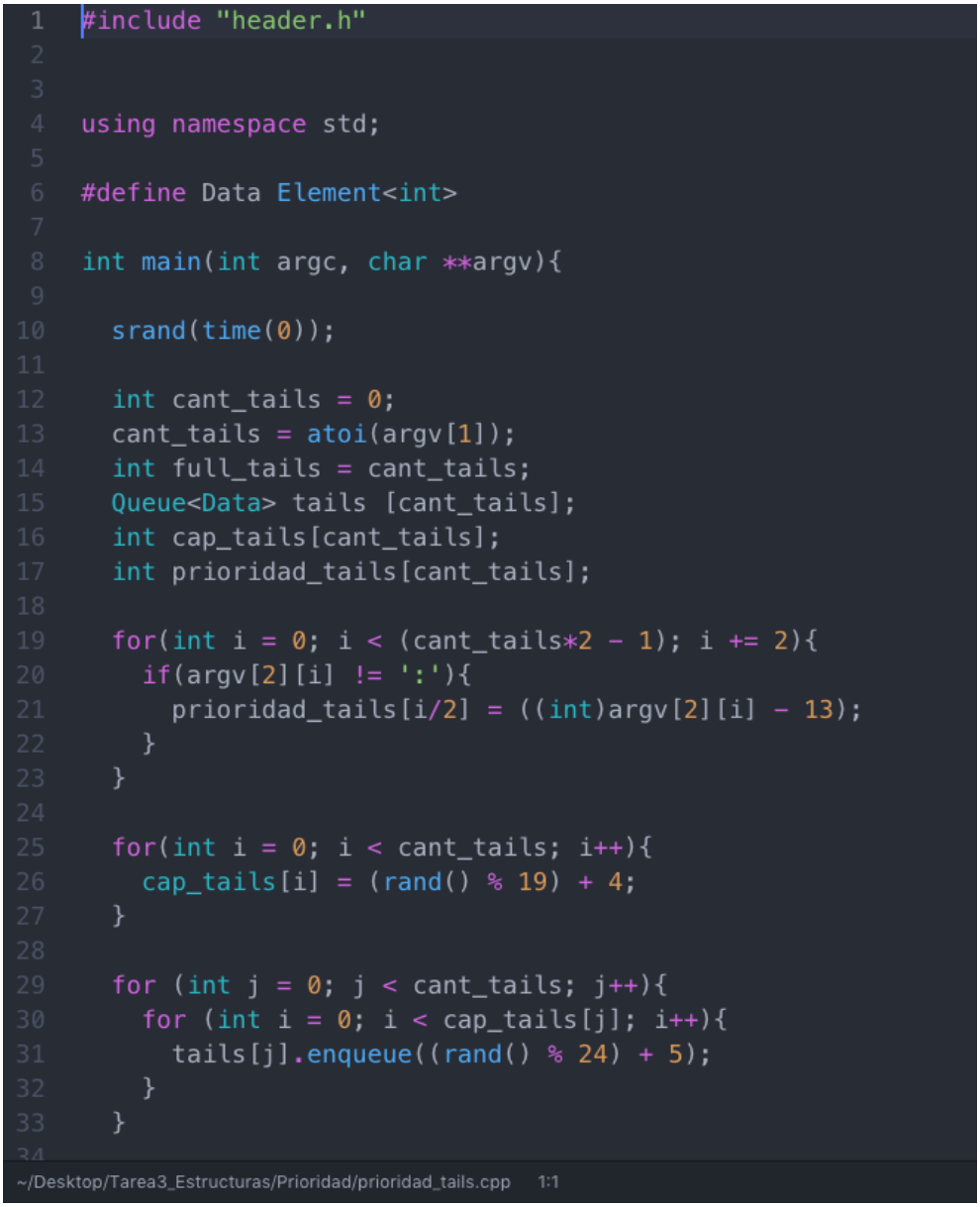

Figura 10: Código de prioridad_tails.cpp

Figura 11: Código de prioridad_tails.cpp

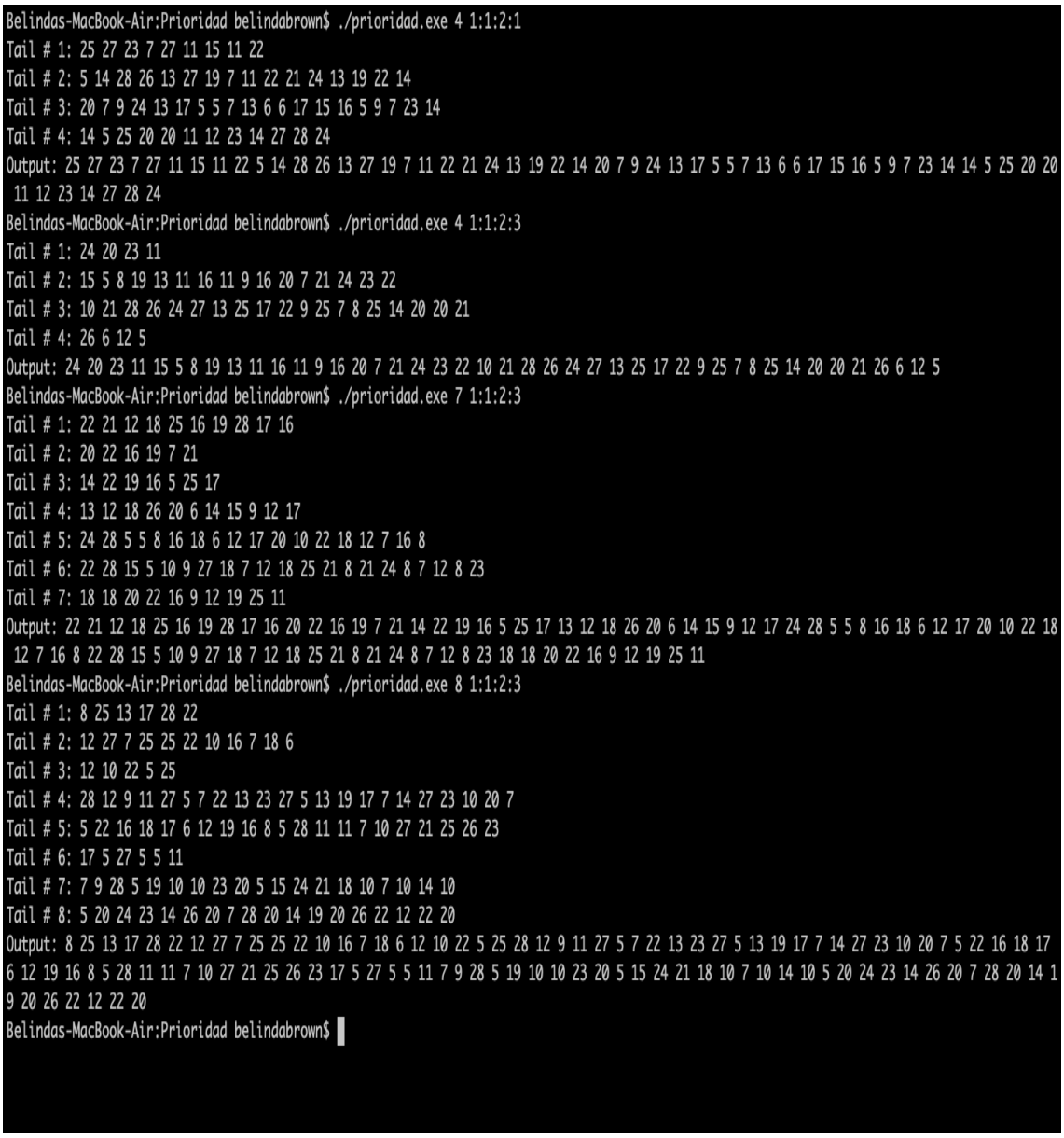

Figura 12: Resultados de Prioridad

Referencias

[1] Mark Summerfield. *Programming in Python 3: A Complete Introduction to the Python Language*. Anaya Multimedia, 2009.

[2] A. M. Turing. *On computable numbers with an application to the Entscheidungs problem*. Proceedings of the london mathematical society, 1997.