

IE-0217 Estructuras abstractas de datos y algoritmos 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

<b>1. Enunciado</b>	<b>2</b>
<b>2. Consideraciones</b>	<b>3</b>
<b>3. Abordaje y conclusiones</b>	<b>3</b>
<b>4. Apéndice</b>	<b>3</b>
4.1. Código fuente . . . . .	3

---

# 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](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:

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

```
1 # ./a.out 8jsle(fasd)asd fds{aefts[fde]]>fd
2 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:

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

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

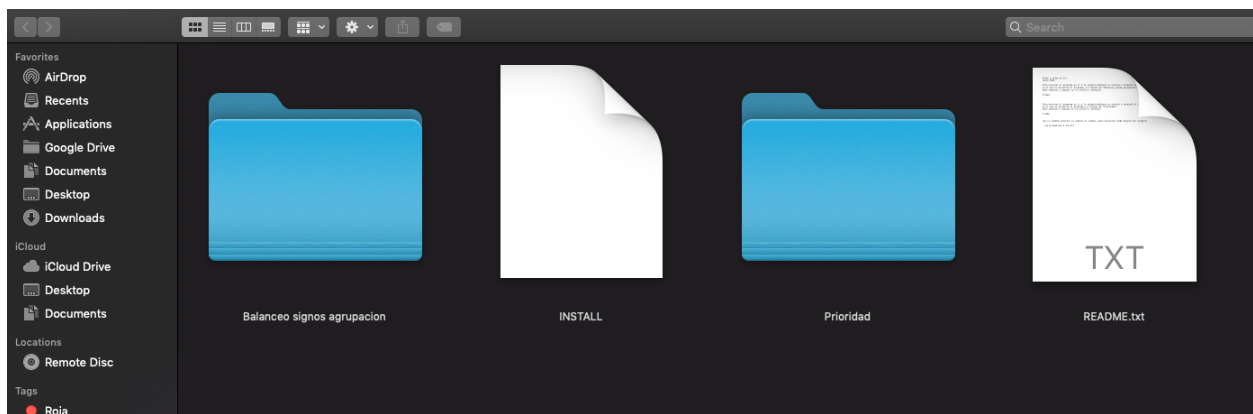


Figura 1: Dentro de Tarea3\_Estructuras

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
1
2  #include "header.h"
3
4  using namespace std;
5
6  bool Pares(char opening,char closing)
7  {
8      if(opening == '(' && closing == ')') return true;
9      else if(opening == '{' && closing == '}') return true;
10     else if(opening == '[' && closing == ']') return true;
11     else if(opening == '<' && closing == '>') return true;
12     return false;
13 }
14 bool Parentesis_balanceados(string exp)
15 {
16     Stack<char> S;
17     for(int i =0;i<exp.length();i++)
18     {
19         if(exp[i] == '(' || exp[i] == '{' || exp[i] == '[' || exp[i] == '<')
20             S.push(exp[i]);
21         else if(exp[i] == ')' || exp[i] == '}' || exp[i] == ']' || exp[i] == '>')
22         {
23             if(S.empty() || !Pares(S.pop(),exp[i]))
24                 return false;
25
26             else
27                 S.pop();
28         }
29     }
30     return S.empty() ? true:false;
31 }
~/Desktop/Tarea3_Estructuras/Balanceo signos agrupacion/balanceo_signos_agrupaci.cpp 22:4

```

Figura 7: Código de balance\_signos\_agrupaci.cpp

```

balanceo_signos_agrupaci.cpp
18 {
19 if(exp[i] == '(' || exp[i] == '{' || exp[i] == '[' || exp[i] == '<')
20     S.push(exp[i]);
21 else if(exp[i] == ')' || exp[i] == '}' || exp[i] == ']' || exp[i] == '>')
22     {
23 if(S.empty() || !Pares(S.pop(),exp[i]))
24     return false;
25
26     else
27         S.pop();
28     }
29 }
30 return S.empty() ? true:false;
31 }
32
33 int main()
34 {
35     string stack_caract;
36     cout<<" *****\n ";
37     cout<<"Digite una expresión con signos de agrupación para verificar\nque el uso de los mismo sea correcto:\n ";
38     cout<<" ***** \n ";
39     cin>>stack_caract;
40
41     if(Parentesis_balanceados(stack_caract))
42         cout<<"\nUso correcto, expresión balanceada\n";
43
44     else
45         cout<<"\nIncorrecto! Uso inadecuado, la expresión no está balanceada\n";
46
47 }
48
~Desktop/Tarea3_Estructuras/Balanceo signos agrupacion/balanceo_signos_agrupaci.cpp 22/4
LF UTF-8 C++ Not on branch GitHub Git (59778)

```

Figura 8: Código de balance\_signos\_agrupaci.cpp



```

    Digite una expresión con signos de agrupación para verificar
    que el uso de los mismo sea correcto:
    *****
    jdhbsdj(dsdhsjkd)shdksjd{dshjb232}

    Uso correcto, expresión balanceada
    rm *.exe
    rm -rf a.exe.dSYM
    Belindas-MacBook-Air:prueba2 belindabrown$ make
    g++ -g --std=c++11 -Wall balanceo_signos_agrupaci.cpp -o a.exe
    ./a.exe
    *****
    Digite una expresión con signos de agrupación para verificar
    que el uso de los mismo sea correcto:
    *****
    wkhksjd<sjkdsjkds}

    Incorrecto! Uso inadecuado, la expresión no está balanceada
    rm *.exe
    rm -rf a.exe.dSYM
    Belindas-MacBook-Air:prueba2 belindabrown$ make
    g++ -g --std=c++11 -Wall balanceo_signos_agrupaci.cpp -o a.exe
    ./a.exe
    *****
    Digite una expresión con signos de agrupación para verificar
    que el uso de los mismo sea correcto:
    *****
    kdkdsd(sd232

    Incorrecto! Uso inadecuado, la expresión no está balanceada
    rm *.exe
    rm -rf a.exe.dSYM
    Belindas-MacBook-Air:prueba2 belindabrown$ make
    g++ -g --std=c++11 -Wall balanceo_signos_agrupaci.cpp -o a.exe
    ./a.exe
    *****
    Digite una expresión con signos de agrupación para verificar
    que el uso de los mismo sea correcto:
    *****
    weuhewekw8323)

    Incorrecto! Uso inadecuado, la expresión no está balanceada

```

Figura 9: Resultados de Balance de signos de agrupación

```

1  #include "header.h"
2
3
4  using namespace std;
5
6  #define Data Element<int>
7
8  int main(int argc, char **argv){
9
10     srand(time(0));
11
12     int cant_tails = 0;
13     cant_tails = atoi(argv[1]);
14     int full_tails = cant_tails;
15     Queue<Data> tails [cant_tails];
16     int cap_tails[cant_tails];
17     int prioridad_tails[cant_tails];
18
19     for(int i = 0; i < (cant_tails*2 - 1); i += 2){
20         if(argv[2][i] != ':'){
21             prioridad_tails[i/2] = ((int)argv[2][i] - 13);
22         }
23     }
24
25     for(int i = 0; i < cant_tails; i++){
26         cap_tails[i] = (rand() % 19) + 4;
27     }
28
29     for (int j = 0; j < cant_tails; j++){
30         for (int i = 0; i < cap_tails[j]; i++){
31             tails[j].enqueue((rand() % 24) + 5);
32         }
33     }
34
35

```

~/Desktop/Tarea3\_Estructuras/Prioridad/prioridad\_tails.cpp 1:1

Figura 10: Código de prioridad\_tails.cpp

```

32     }
33 }
34
35 for (int i = 0; i < cant_tails; i++){
36     cout << "Tail # " << i+1 << ": ";
37     tails[i].imprimirCola();
38 }
39
40
41
42 cout <<"Output: ";
43 while(full_tails > 0){
44     full_tails = cant_tails;
45     for (int j = 0; j < cant_tails; j++){
46         if(!tails[j].empty()){
47             for (int i = 0; i < prioridad_tails[j]; i++){
48                 Data e = tails[j].dequeue();
49                 if (e.isValid())
50                 {
51                     cout << e.get() << " ";
52                 }
53             }
54         }
55         else{
56             full_tails --;
57         }
58     }
59 }
60
61 }
62 cout << endl;
63
64 }
65

```

~/Desktop/Tarea3\_Estructuras/Prioridad/prioridad\_tails.cpp 1:1

Figura 11: Código de prioridad\_tails.cpp

```

Belindas-MacBook-Air:Prioridad belindabrown$ ./prioridad.exe 4 1:1:2:1
Tail # 1: 25 27 23 7 27 11 15 11 22
Tail # 2: 5 14 28 26 13 27 19 7 11 22 21 24 13 19 22 14
Tail # 3: 20 7 9 24 13 17 5 5 7 13 6 6 17 15 16 5 9 7 23 14
Tail # 4: 14 5 25 20 20 11 12 23 14 27 28 24
Output: 25 27 23 7 27 11 15 11 22 5 14 28 26 13 27 19 7 11 22 21 24 13 19 22 14 20 7 9 24 13 17 5 5 7 13 6 6 17 15 16 5 9 7 23 14 14 5 25 20 20
11 12 23 14 27 28 24
Belindas-MacBook-Air:Prioridad belindabrown$ ./prioridad.exe 4 1:1:2:3
Tail # 1: 24 20 23 11
Tail # 2: 15 5 8 19 13 11 16 11 9 16 20 7 21 24 23 22
Tail # 3: 10 21 28 26 24 27 13 25 17 22 9 25 7 8 25 14 20 20 21
Tail # 4: 26 6 12 5
Output: 24 20 23 11 15 5 8 19 13 11 16 11 9 16 20 7 21 24 23 22 10 21 28 26 24 27 13 25 17 22 9 25 7 8 25 14 20 20 21 26 6 12 5
Belindas-MacBook-Air:Prioridad belindabrown$ ./prioridad.exe 7 1:1:2:3
Tail # 1: 22 21 12 18 25 16 19 28 17 16
Tail # 2: 20 22 16 19 7 21
Tail # 3: 14 22 19 16 5 25 17
Tail # 4: 13 12 18 26 20 6 14 15 9 12 17
Tail # 5: 24 28 5 5 8 16 18 6 12 17 20 10 22 18 12 7 16 8
Tail # 6: 22 28 15 5 10 9 27 18 7 12 18 25 21 8 21 24 8 7 12 8 23
Tail # 7: 18 18 20 22 16 9 12 19 25 11
Output: 22 21 12 18 25 16 19 28 17 16 20 22 16 19 7 21 14 22 19 16 5 25 17 13 12 18 26 20 6 14 15 9 12 17 24 28 5 5 8 16 18 6 12 17 20 10 22 18
12 7 16 8 22 28 15 5 10 9 27 18 7 12 18 25 21 8 21 24 8 7 12 8 23 18 18 20 22 16 9 12 19 25 11
Belindas-MacBook-Air:Prioridad belindabrown$ ./prioridad.exe 8 1:1:2:3
Tail # 1: 8 25 13 17 28 22
Tail # 2: 12 27 7 25 25 22 10 16 7 18 6
Tail # 3: 12 10 22 5 25
Tail # 4: 28 12 9 11 27 5 7 22 13 23 27 5 13 19 17 7 14 27 23 10 20 7
Tail # 5: 5 22 16 18 17 6 12 19 16 8 5 28 11 11 7 10 27 21 25 26 23
Tail # 6: 17 5 27 5 5 11
Tail # 7: 7 9 28 5 19 10 10 23 20 5 15 24 21 18 10 7 10 14 10
Tail # 8: 5 20 24 23 14 26 20 7 28 20 14 19 20 26 22 12 22 20
Output: 8 25 13 17 28 22 12 27 7 25 25 22 10 16 7 18 6 12 10 22 5 25 28 12 9 11 27 5 7 22 13 23 27 5 13 19 17 7 14 27 23 10 20 7 5 22 16 18 17
6 12 19 16 8 5 28 11 11 7 10 27 21 25 26 23 17 5 27 5 5 11 7 9 28 5 19 10 10 23 20 5 15 24 21 18 10 7 10 14 10 5 20 24 23 14 26 20 7 28 20 14 1
9 20 26 22 12 22 20
Belindas-MacBook-Air:Prioridad belindabrown$ █

```

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.