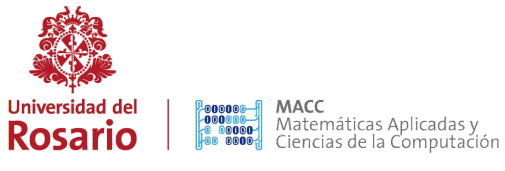
Pilas y Colas

Carlos E. Alvarez[1].

[1]Dep. de Matemáticas aplicadas y Ciencias de la Computación, Universidad del Rosario

2019-II



Pilas

Inicialización

```
#include <iostream>          // std::cout
#include <stack>             // std::stack
#include <vector>            // std::vector

int main(){
  std::vector<int> myvector(2, 200);

  std::stack<int> first; // empty stack
  //Initialize stack using vector
  std::stack<int,std::vector<int>> second(myvector);

  std::cout << "size of first: " << first.size() << '\n';
  std::cout << "size of second: " << second.size() << '\n';

  return 0;
}
```

Pilas

push / pop

```
int main(){
  std::stack<int> mystack;

  for (int i = 0; i < 5; ++i) mystack.push(i);

  std::cout << "Popping out elements...";
  while (!mystack.empty()) {
    std::cout << ' ' << mystack.top();
    mystack.pop();
  }

  std::cout << '\n';

  return 0;
}
```

Pilas

top

```
int main(){
  std::stack<int> mystack;

  mystack.push(10);
  mystack.push(20);

  mystack.top() -= 5;

  std::cout << "mystack.top() is now "
            << mystack.top() << '\n';

  return 0;
}
```

Pilas: Ejemplo(Revisar paréntesis)

```
int main(){
  string code = "if((a[3]=={2,4,1}) and has_element(b))";
  stack<char> p; //Pila vacía

  for(int i = 0; i < code.length(); i++){
    char c = code.at(i);
    if(is_parenthesis(c)){ //Si es un paréntesis
      if(!p.empty() && is_pair(p.top(),c)){
        p.pop();
      }else{
        if(c == '(' || c == '[' || c == '{'){
              p.push(c);
            }else{
              cout << "Closing unopened parenthesis\n";
              return -1;
            }
          }
      }
   }
```

Pilas: Ejemplo(Revisar paréntesis)

```
    if(!p.empty()){
        cout << "Parenthesis left unclosed\n";
        return -1;
    }

    cout << "Ok" << endl;

    return 0;
}
```

Pilas

Ejercicios:

1. Ponga comentarios a cada ciclo y condicional del programa explicando lo que hace
2. Complete e implemente el código que revisa paréntesis. La función `is_parenthesis` recibe un caracter y retorna true si es un paréntesis o false ecc. La función `is_pair` recibe dos caracteres c1 y c2. Si c1 es un paréntesis de apertura y c2 es el paréntesis correspondiente que lo cierra, retorna true. Retorna false ecc

Colas

Inicialización

```
#include <iostream>
#include <queue>

int main(){
  std::queue<int> first;  // empty queue
  std::queue<int> second(first);// another empty queue

  std::cout << "size of first: " << first.size() << '\n';
  std::cout << "size of second: " << second.size() << '\n';

  return 0;
}
```

Colas

push / pop

```
int main(){
  std::queue<int> myqueue;
  int myint;

  std::cout << "Enter some integers (0 to end):\n";
  do{
    std::cin >> myint;
    myqueue.push(myint);
  }while(myint);

  std::cout << "myqueue contains: ";
  while (!myqueue.empty()) {
    std::cout << ' ' << myqueue.front();
    myqueue.pop();
  }
  std::cout << '\n';
  return 0;
}
```

Colas

front

```
int main(){
  std::queue<int> myqueue;

  myqueue.push(77);
  myqueue.push(16);

  myqueue.front() -= myqueue.back(); // 77 - 16 = 61

  std::cout << "myqueue.front() is now "
            << myqueue.front() << '\n';

  return 0;
}
```

Colas

back

```
int main(){
  std::queue<int> myqueue;

  myqueue.push(12);
  myqueue.push(75);   // this is now the back

  myqueue.back() -= myqueue.front();

  std::cout << "myqueue.back() is now "
            << myqueue.back() << '\n';

  return 0;
}
```

Colas

Ejercicios:

- Escriba un programa que use una estructura de cola para simular una fila de atención al cliente
- El programa pide al usuario que ingrese uno a uno los nombres de los clientes y los añade a la cola, hasta que se ingrese `end`
- Luego, el programa despacha (elimina) uno a uno los clientes, iniciando por el que llegó primero e imprime

  ```
  Serving <client_name>
  ```

  por cada cliente en la cola hasta que termine