



UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

**FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA PROGRAMA DE
INGENIERO EN SOFTWARE Y TECNOLOGÍAS EMERGENTES**

Patrones de Software (13134)

Identificación y manejo de material de laboratorio

29 de Junio 2023

Docente:

Manuel Castañón Puga

Participante(es):

Luis Eduardo Galindo Amaya (1274895)

Juan Fransisco Perez Valdez (324342)

Índice

1. Hola	2
1.1. Ejemplos de código desde archivos externos	2
1.2. Nam vestibulum accumsan nisl	7
2. asdad	7
3. Hola como estan kaskas	8
3.1. Test	8
4. Test Table	9
5. Referencias	9

Universidad Autónoma de Baja California

Facultad de ciencias químicas e ingeniería

Ingeniero en software y tecnologías emergentes

Información de la materia

Nombre de la materia y clave: Patrones de Software (13134)
Grupo y periodo: 13134 (2022-1)
Profesor: Manuel Castañón Puga.

Información de la actividad

Nombre de la actividad: Identificación y manejo de material de laboratorio
Lugar y fecha: 29 de Junio 2023
Carácter de la actividad: Individual.

Reporte de actividades

1. Hola

Nam euismod tellus id erat. Pellentesque dapibus suscipit ligula. Donec posuere augue in quam. Etiam vel tortor sodales tellus ultricies commodo. Suspendisse potenti. Aenean in sem ac leo mollis blandit. Donec neque quam, dignissim in, mollis nec, sagittis eu, wisi.

1.1. Ejemplos de código desde archivos externos

Phasellus lacus. Etiam laoreet quam sed arcu. Phasellus at dui in ligula mollis ultricies. Integer placerat tristique nisl. Praesent augue. Fusce commodo. Vestibulum convallis, lorem a tempus semper,

```
1 (setq org-latex-caption-above nil)
2
3 (use-package modus-themes
4   :config
5   ;; (load-theme 'modus-operandi t)
6   (load-theme 'modus-vivendi t)
7   )
8
9 (defun reverse-region (beg end)
10  "Reverse characters between BEG and END."
11  (interactive "r")
12  (let ((region (buffer-substring beg end)))
13    (delete-region beg end)
14    (insert (nreverse region))))
```

```

1  #include <assert.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define ELEMENTS_TYPE int
6
7  /**
8   * Estructura que representa la doble cola
9   * @param size número de elementos en la cola
10  * @param maxSize tamaño máximo de la cola
11  * @param lastSize número de elementos en la cola desde el final
12  * @param firstSize número de elementos en la cola desde el principio
13  */
14  typedef struct DoubleQueue {
15      ELEMENTS_TYPE *array;
16      int maxSize;
17      int size;
18      int lastSize;
19      int firstSize;
20  } DoubleQueue;
21
22  /**
23   * Inicializa la doble cola con el tamaño dado
24   */
25  void DoubleQueueInit(DoubleQueue *queue, int size) {
26      queue->array = calloc(sizeof(ELEMENTS_TYPE), size);
27      queue->maxSize = size;
28      queue->size = 0;
29      queue->firstSize = 0;
30      queue->lastSize = 0;
31  }
32
33  /**
34   * Función para imprimir información de depuración de la doble cola
35   */
36  void printDQDebug(DoubleQueue *queue) {
37
38      printf("\nQueue Head(%d): ", queue->firstSize);
39      for (int i = 0; i < queue->firstSize; i++) {
40          printf("%d ", queue->array[i]);
41      }
42
43      printf("\nQueue rear(%d): ", queue->lastSize);
44      for (int i = 0; i < queue->lastSize; i++) {
45          printf("%d ", queue->array[queue->maxSize - i]);
46      }
47  }

```

```

48
49 /**
50  * Agrega un elemento a la doble cola en el índice dado (0 para agregar
51  * al principio, 1 para agregar al final)
52  */
53 void DoubleQueueEnqueue(DoubleQueue *queue,
54                          int queueIndex,
55                          ELEMENTS_TYPE element) {
56     /* Solo permitir el 1 o el 0 como índice */
57     assert(queueIndex == 1 || queueIndex == 0);
58
59     /* Verificar que haya espacio disponible en la cola */
60     assert(queue->size < queue->maxSize);
61
62     if (queueIndex) {
63         (queue->array)[queue->maxSize - queue->lastSize] = element;
64         (queue->lastSize)++;
65     } else {
66         (queue->array)[queue->firstSize] = element;
67         (queue->firstSize)++;
68     }
69
70     (queue->size)++;
71 }
72
73 /**
74  * Libera la memoria del arreglo
75  */
76 void DoubleQueueFree(DoubleQueue *queue) {
77     free(queue->array);
78     queue->array = NULL;
79     queue->firstSize = 0;
80     queue->lastSize = 0;
81     queue->size = 0;
82     queue->maxSize = 0;
83 }
84
85 /**
86  * Extrae el primer elemento de la doble cola en el índice dado (0 para
87  * extraer del principio, 1 para extraer del final)
88  */
89 ELEMENTS_TYPE DoubleQueueDequeue(DoubleQueue *queue, int queueIndex) {
90     /* Solo permitir el 1 o el 0 como índice */
91     assert(queueIndex == 1 || queueIndex == 0);
92
93     ELEMENTS_TYPE element;
94     ELEMENTS_TYPE *queueArray = queue->array;
95

```

```

96     if (queueIndex) {
97         assert(queue->lastSize > 0);
98         element = queueArray[queue->maxSize];
99
100        /* Reordenar la cola después de extraer el elemento */
101
102        for (int i = 0; i < queue->lastSize; i++) {
103            queueArray[queue->maxSize - i] = queueArray[queue->maxSize - i - 1];
104        }
105
106        (queue->lastSize)--;
107
108    } else {
109        assert(queue->firstSize > 0);
110        element = queueArray[0];
111
112        /* Reordenar la cola después de extraer el elemento */
113        for (int i = 1; i < queue->firstSize; i++) {
114            queueArray[i - 1] = queueArray[i];
115        }
116
117        (queue->firstSize)--;
118    }
119
120    (queue->size)--;
121    return element;
122 }
123
124 ELEMENTS_TYPE DoubleQueueFront(DoubleQueue *queue, int queueIndex) {
125     /* Solo permitir el 1 o el 0 como índice */
126     assert(queueIndex == 1 || queueIndex == 0);
127
128     ELEMENTS_TYPE element;
129     ELEMENTS_TYPE *queueArray = queue->array;
130
131     if (queueIndex) { /* last */
132         assert(queue->lastSize > 0);
133         element = queueArray[queue->maxSize];
134
135     } else { /* first */
136         assert(queue->firstSize > 0);
137         element = queueArray[0];
138     }
139
140     return element;
141 }
142
143

```

```

144 int main() {
145     DoubleQueue dq;
146     int queue;
147     int array_size;
148     int op;
149
150     /* array setup ----- */
151 array_setup:
152     printf("\n Setup \n"
153           "Tamaño del Array: ");
154     scanf("%d", &array_size);
155     puts(" ");
156
157     if(array_size <= 0) {
158         goto exit;
159     }
160
161     DoubleQueueInit(&dq, array_size);
162     goto queue_select;
163
164
165     /* seleccionar cola ----- */
166 queue_select:
167
168     printf("\n\n Select queue \n");
169     printDQDebug(&dq);
170
171     printf("\n"
172           "1) first      \n"
173           "2) last        \n"
174           "0) Exit        \n");
175     printf("> ");
176     scanf("%d", &queue);
177
178     if(queue) {
179         queue--;
180         goto operation_select;
181     }
182
183     goto exit;
184
185
186     /* Menu ----- */
187 operation_select:
188     printf("\n Operation \n"
189           "1) Enqueue  \n"
190           "2) Dequeue   \n"
191           "0) Return    \n");

```

```

192
193     printf("> ");
194     scanf("%d", &op);
195
196     switch (op) {
197     case 1:
198         printf("Enqueue value (int): ");
199         int a;
200         scanf("%d", &a);
201         DoubleQueueEnqueue(&dq, queue, a);
202         break;
203
204     case 2:
205         printf("Dequeue value: %d\n", DoubleQueueDequeue(&dq, queue));
206         break;
207     }
208
209     goto queue_select;
210
211
212
213     exit:
214     return 0;
215 }

```

```

1 compile:
2     gcc -W -Wall -pedantic -std=c99 main.c -o duda.out
3     ./duda.out

```

Using biblatex you can display a bibliography divided into sections, depending on citation type. Let's cite! Einstein's journal paper

1.2. Nam vestibulum accumsan nisl

dui dui euismod elit, vitae placerat urna tortor vitae lacus. Nullam libero mauris, consequat quis, varius et, dictum id, arcu. Mauris

2. asdad

mollis tincidunt felis. Aliquam feugiat tellus ut neque. Nulla facilisis, risus a rhoncus fermentum, tellus tellus lacinia purus, et dictum nunc justo sit amet elit Einstein, 1905.

3. Hola como estan kaskas

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla posuere. Donec vitae dolor. Nullam tristique diam non turpis. Cras placerat accumsan nulla. Nullam rutrum. Nam vestibulum accumsan nisl.



Figura 1: Marienallee in Dahlem, Euskirchen district

3.1. Test

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec hendrerit tempor tellus. Donec pretium posuere tellus. Proin quam nisl, tincidunt et, mattis eget, convallis nec, purus.

```
1 import numpy as np
2
3 def incmatrix(genl1, genl2):
4     m = len(genl1)
5     n = len(genl2)
6     M = None #to become the incidence matrix
7     VT = np.zeros((n*m,1), int) #dummy variable
8
9     #compute the bitwise xor matrix
10    M1 = bitxormatrix(genl1)
11    M2 = np.triu(bitxormatrix(genl2),1)
12
13    for i in range(m-1):
14        for j in range(i+1, m):
15            [r,c] = np.where(M2 == M1[i,j])
16            for k in range(len(r)):
17                VT[(i)*n + r[k]] = 1;
18                VT[(i)*n + c[k]] = 1;
19                VT[(j)*n + r[k]] = 1;
20                VT[(j)*n + c[k]] = 1;
21
```

```

22         if M is None:
23             M = np.copy(VT)
24         else:
25             M = np.concatenate((M, VT), 1)
26
27         VT = np.zeros((n*m,1), int)
28
29     return M

```

Figura 2: Hola como esatn

4. Test Table

Hola	Como	Estan en este dia
Aliquam posuere. j wqeqwe	jk dsadsa k eweqw	jk eweweq jkjkj

Tabla 1: Hola

5. Referencias

Einstein, A. (1905). Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]. *Annalen der Physik*, 322(10), 891-921. <https://doi.org/http://dx.doi.org/10.1002/andp.19053221004>