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 Castañon Puga

Participante(es):
Luis Eduardo Galindo Amaya (1274895)
Juan Fransisco Perez Valdez (324342)

Índice

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 Castañon 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 codigo 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,

```
(setq org-latex-caption-above nil)

(use-package modus-themes
  :config
  ;; (load-theme 'modus-operandi t)
  (load-theme 'modus-vivendi t)
  )

(defun reverse-region (beg end)
  "Reverse characters between BEG and END."
  (interactive "r")
  (let ((region (buffer-substring beg end)))
    (delete-region beg end)
    (insert (nreverse region))))
```

```c
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>

#define ELEMENTS_TYPE int

/**
 * Estructura que representa la doble cola
 * @param size número de elementos en la cola
 * @param maxSize tamaño máximo de la cola
 * @param lastSize número de elementos en la cola desde el final
 * @param firstSize número de elementos en la cola desde el principio
 */
typedef struct DoubleQueue {
  ELEMENTS_TYPE *array;
  int maxSize;
  int size;
  int lastSize;
  int firstSize;
} DoubleQueue;

/**
 * Inicializa la doble cola con el tamaño dado
 */
void DoubleQueueInit(DoubleQueue *queue, int size) {
  queue->array = calloc(sizeof(ELEMENTS_TYPE), size);
  queue->maxSize = size;
  queue->size = 0;
  queue->firstSize = 0;
  queue->lastSize = 0;
}

/**
 * Función para imprimir información de depuración de la doble cola
 */
void printDQDebug(DoubleQueue *queue) {

  printf("\nQueue Head(%d): ", queue->firstSize);
  for (int i = 0; i < queue->firstSize; i++) {
    printf("%d ", queue->array[i]);
  }

  printf("\nQueue rear(%d): ", queue->lastSize);
  for (int i = 0; i < queue->lastSize; i++) {
    printf("%d ", queue->array[queue->maxSize - i]);
  }
}
```

```
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
```

```
  if (queueIndex) {
    assert(queue->lastSize > 0);
    element = queueArray[queue->maxSize];

    /* Reordenar la cola después de extraer el elemento */

    for (int i = 0; i < queue->lastSize; i++) {
      queueArray[queue->maxSize - i] = queueArray[queue->maxSize - i - 1];
    }

    (queue->lastSize)--;

  } else {
    assert(queue->firstSize > 0);
    element = queueArray[0];

    /* Reordenar la cola después de extraer el elemento */
    for (int i = 1; i < queue->firstSize; i++) {
      queueArray[i - 1] = queueArray[i];
    }

    (queue->firstSize)--;
  }

  (queue->size)--;
  return element;
}

ELEMENTS_TYPE DoubleQueueFront(DoubleQueue *queue, int queueIndex) {
  /* Solo permitir el 1 o el 0 como índice */
  assert(queueIndex == 1 || queueIndex == 0);

  ELEMENTS_TYPE element;
  ELEMENTS_TYPE *queueArray = queue->array;

  if (queueIndex) { /* last */
    assert(queue->lastSize > 0);
    element = queueArray[queue->maxSize];

  } else { /* first */
    assert(queue->firstSize > 0);
    element = queueArray[0];
  }

  return element;
}
```

```
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, consectetuer adipiscing elit. Donec hendrerit tempor tellus. Donec pretium posuere tellus. Proin quam nisl, tincidunt et, mattis eget, convallis nec, purus.

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

```

```
                    if M is None:
                        M = np.copy(VT)
                    else:
                        M = np.concatenate((M, VT), 1)

                    VT = np.zeros((n*m,1), int)

    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