Universidad Nacional de Ingeniería
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación
Introducción a la Programación Universidad

Cuarta Práctica Calificada

CC102-AB **Ciclo: 2017-1**

Sugerencia: Para leer una cadena de caracteres **cc** del teclado puede usar gets(cc); y la saldrá un **warning**; déjelo pasar por el exámen; pero si tiene tiempo de ser perfecto, puede usar leerString(**cc**, **n**):

Sintaxis	**Ejemplo**
Int leerString(char **cc**[ ], int **n**); **Entradas**: **cc**: un arreglo de caracteres n = longitud del arreglo **Proceso:** Lee el buffer hasta encontrar <enter> ó se completen n-1 caracteres, agrega '\0' al final de la cadena y desecha la cola del buffer. **Salida**: número da caracteres leídos	char arr[6]; leerString(arr, 6); // 6 es el tamaño del arreglo Si tipea: **a b**<enter> arr ← "**a b\0**" y limpia el buffer. Si tipea: **ab cde**<enter> arr contiene "**ab cd\0**" y limpia el buffer.. Si tipea: **ab cdefg**<enter> arr contiene "**ab cd\0**" y limpia el buffer.

```
int leerString(char cc[ ], int n){
   int c, m=0;
   while((c=getchar())!=10) if(m<n-1)cc[m++]= c;
   cc[m] = '\0';
   return m;
}
```

1. [5 ptos.] Un estudio de colas de pasajeros del Metropolitano requiere programar lo siguiente: En un intervalo de tiempo, a una estación, llegan grupos de n1, n2, … pasajeros, que se leen desde el teclado, terminar ingresando 0 pasajeros; n1, n2, … se guardan en memoria dinámica apuntada por *personas la cual crece 1 unidad por cada grupo que llega. Lea los datos, y los va grabando en *personas, reporte los datos, ordene ascendentemente y vuelva a reportarlos.
   Sugerencia: Puede utilizar la siguiente main() o algo similar:

```
void leer    (int **vPersonas, int *n);
void ordenar (int *vPersonas,  int  n);
void reportar(int *vPersonas,  int  n);
void main(void){
    int *vPersonas = NULL, n = 0;        // n = tamaño de la memoria dinámica
    leer    (&vPersonas, &n);
    reportar( vPersonas,  n);
    ordenar ( vPersonas,  n);
    printf ("Datos ordenados\n");
    reportar( vPersonas,  n);
    free(vPersonas);
}
```

SOLUCION:

```
// AB1.c
#include<stdio.h>
#include <stdlib.h>
void leer    (int **vPersonas, int *n);
void ordenar (int *vPersonas,  int  n);
void reportar(int *vPersonas,  int  n);
void main(void){
        int *vPersonas = NULL, n = 0;
        leer    (&vPersonas, &n);
```

```
	reportar( vPersonas,  n);
	ordenar ( vPersonas,  n);
	printf ("Datos ordenados\n");
	reportar( vPersonas,  n);
	free(vPersonas);
}
void leer(int **vPersonas, int *n){
	int nPer;
	do {
		printf("Ingrese numero de personas que llegan; 0 para terminar: ");
		scanf("%d", &nPer);
		if(nPer) {
			(*n)++;
			*vPersonas = realloc(*vPersonas,*n*sizeof(int));
			if(*vPersonas==NULL) {
				printf("No hay suficiente memoria");
				exit(EXIT_FAILURE);
			}
			*(*vPersonas+*n-1) = nPer;
		}
	} while(nPer);
}
void ordenar(int *vPersonas, int n){
	int *imin, min, *p = vPersonas, *pmax = p + n, *q;
	while(p < pmax-1){
		min  = *p;
		imin = p;
		for(q=p+1; q < pmax; q++)
			if(min > *q){
				min  = *q;
				imin =  q;
			}
		if(imin > p){
			*imin = *p;
			*p = min;
		}
		p++;
	}
}
void reportar(int *vPersonas, int n){
	int *p = vPersonas, *pmax = p + n;
	while (p<pmax) printf("%d\n", *p++);
}
```

2. [5 ptos.] Continuando con el estudio del Metropolitano (problema 1, si desea copie el programa 1.c y lo continua). Aloje 10 datos ordenados: 2, 2, 4, 4, 5, 5, 5, 6, 6, 7 en memoria dinámica apuntada por *personas. Se requiere la siguiente estadística de *personas:
   - Número de grupos que llegaron = 10
   - La moda (número que más se repite) y la cantidad de veces que se repite = 5, 3
   - El promedio de personas en todo el experimento = 4.6

   Sugerencia: Puede utilizar la siguiente main() o algo similar:

```
	void inicio  (int **vPersonas, int n);
	void reportar(int  *vPersonas, int n);
```

```
void main(void){
    int *vPersonas = NULL, n = 10;
    inicio  (&vPersonas, n);
    reportar( vPersonas, n);
    free(vPersonas);
}
```

SOLUCION:

```
// AB2.c
#include<stdio.h>
#include <stdlib.h>
void inicio  (int **vPersonas, int n);
void reportar(int  *vPersonas, int n);
void main(void){
	int *vPersonas = NULL, n = 10;
	inicio  (&vPersonas, n);
	reportar( vPersonas, n);
	free(vPersonas);
}
void inicio(int **vPersonas, int n){
	*vPersonas = malloc(n*sizeof(int));
	if(*vPersonas==NULL) {
		printf("No hay suficiente memoria");
		exit(EXIT_FAILURE);
	}
	int *p = *vPersonas, *pmax = p + n;
	*p++ = 2;
	*p++ = 2;
	*p++ = 4;
	*p++ = 4;
	*p++ = 5;
	*p++ = 5;
	*p++ = 5;
	*p++ = 6;
	*p++ = 6;
	*p++ = 7;
}
void reportar(int *vPersonas, int n){
	int *p = vPersonas, *pmax = p + n;
	int moda = *p, nmoda = 1, modaTotal = moda, nmodaTotal = 1, promedio = 0, mediana;
	printf("\nEstadistica\n");
	printf("Número de grupos: %d\n", n);
// moda
	p++;
	while(p<pmax){
		if(*(p-1) == *p) nmoda++;
		else {
			if(nmodaTotal < nmoda){
				nmodaTotal = nmoda;
				 modaTotal =  moda;
			}
			 moda = *p;
			nmoda = 1;
		}
		p++;
```

```
        }
        if(nmodaTotal < nmoda) {
                nmodaTotal = nmoda;
                 modaTotal =  moda;
        }
        printf("Moda          : %d, se repite: %d\n", modaTotal, nmodaTotal);
// promedio
        p = vPersonas;
        while(p<pmax){
                promedio += *p;
                p++;
        }
        printf("promedio      : %.2f\n", promedio/(float)n);
}
```

3. [5 ptos.] El dueño del gimnasio “LYON GYM” desea automatizar la información de los deportistas que asisten a su gimnasio para realizar ejercicios. Los datos con que cuenta de cada deportista son: nombre, edad, peso (kg), altura. Escriba un programa que registre un nuevo cliente del gimnasio e imprima sus datos

SOLUCION:

```
// AB3.c
#include<stdio.h>
#include<stdio.h>
typedef struct Deportista{
        char nombre[30];
        int edad;
        float peso, altura;
} depor;

int leerString(char cc[ ], int n){
   int c, m=0;
   while((c=getchar())!=10) if(m<n-1)cc[m++]= c;
   cc[m] = '\0';
   return m;
}

void main(void){
        depor d1;
        printf("Nombre del deportista: ");
        leerString(d1.nombre, 30);
        printf("Edad: ");
        scanf("%d",&d1.edad);
        printf("Peso(Kg): ");
        scanf("%f",&d1.peso);
        printf("Altura: " );
        scanf("%f",&d1.altura);
        printf("Datos del Deportista\n");
        printf("Nombre del deportista: %s\n",d1.nombre);
        printf("Edad: %d\n",d1.edad);
        printf("Peso: %f\n",d1.peso);
        printf("Altura: %f\n",d1.altura);
}
```

4. [5 ptos.] Dada una estructura llamada complejo{ int real, int imaginario}. Realice un programa que

sume dos números complejos, ingresados desde el teclado.

SOLUCION:

```
// AB4.c
#include <stdio.h>
struct complejo {
        int real;
        int imaginario;
};
void main(void) {
        struct complejo a,b,c;
        printf("Ingrese el primer numero complejo a = n1 + in2\n");
        printf("n1: ");
        scanf("%d",&a.real);
        printf("\nn2: ");
        scanf("%d",&a.imaginario);
        printf("\nIngrese el segundo numero complejo a = n3 + in4\n");
        printf("n3: ");
        scanf("%d",&b.real);
        printf("\nn4: ");
        scanf("%d",&b.imaginario);
        c.real=a.real + b.real;
        c.imaginario=a.imaginario + b.imaginario;
        printf("La suma de numeros complejos = %d + %di \n",c.real,c.imaginario);
}
```

Cuarta Práctica Calificada

CC102-CD **Ciclo: 2017-1**

Sugerencia: Para leer una cadena de caracteres **cc** del teclado puede usar gets(cc); y la saldrá un **warning**; déjelo pasar por el exámen; pero si tiene tiempo de ser perfecto, puede usar leerString(**cc**, **n**):

Sintaxis	**Ejemplo**
Int leerString(char **cc**[ ], int **n**); **Entradas**: **cc**: un arreglo de caracteres n = longitud del arreglo **Proceso:** Lee el buffer hasta encontrar <enter> ó se completen n-1 caracteres, agrega '\0' al final de la cadena y desecha la cola del buffer. **Salida**: número da caracteres leídos	char arr[6]; leerString(arr, 6); // 6 es el tamaño del arreglo Si tipea: **a b**<enter> arr ← "**a b\0"** y limpia el buffer. Si tipea: **ab cde**<enter> arr contiene "**ab cd\0"** y limpia el buffer.. Si tipea: **ab cdefg**<enter> arr contiene "**ab cd\0"** y limpia el buffer.

```
int leerString(char cc[ ], int n){
   int c, m=0;
   while((c=getchar())!=10) if(m<n-1)cc[m++]= c;
   cc[m] = '\0';
   return m;
}
```

1. [5 ptos.] Escriba un programa que lea un número entero **m** en base 10, para convertirlo a base binaria, guardando cada cifra binaria en un vector dinámico llamado *vBinario. El algoritmo es el siguiente:
   a. Ingresar el número entero **m** >= 0.
   b. A medida que se hace las divisiones sucesivas entre 2, se aumenta en 1 unidad el tamaño de la memoria dinámica apuntada por *vBinario y se guarda el residuo al final de *vBinario. Se recalcula m dividiéndolo entre 2.
   c. El proceso termina cuando **m** = 0.

d. Se imprimen todos los valores de *vBinario de atrás para adelante y se verá el número binario.

SOLUCION:

```
// CD1.c
#include<stdio.h>
#include<stdlib.h>
void main(void){
	int *vBinario = NULL, *p, nBin = 0;
	unsigned int m;
	printf("Ingrese un entero >= 0: ");
	scanf("%u", &m);
	do{
		vBinario = (int *) realloc (vBinario, ++nBin *sizeof(int));
		if(vBinario==NULL) {
			printf("No hay suficiente memoria");
			exit(EXIT_FAILURE);
		}
		*(vBinario + nBin -1 )= m%2;
		m /= 2;
	} while(m>0);
	for(p = vBinario + nBin -1; p >= vBinario; p--) printf("%d", *p);
	printf("\n");
	free(vBinario);
}
```

2. [5 ptos.] Escriba un programa que lea dos 2 enteros positivos **m** y **n**, defina un área dinámica de tamaño m*n para alojar datos de una matriz **mm[m][n]** con valores **mm[i][j] = i+j**, finalmente imprima la matriz; ejemplo si m = 3, **n = 2**, **mm** será:

```
0  1
1  2
2  3
```

SOLUCION:

```
// CD2.c
#include<stdio.h>
#include<stdlib.h>
void main(void){
	int m, n, *mm, i, j, ij;
	printf("Ingrese dos entero >= 0: ");
	scanf("%d %d", &m, &n);
	mm = (int *) malloc (m * n *sizeof(int));
	if(mm==NULL) {
		printf("No hay suficiente memoria");
		exit(EXIT_FAILURE);
	}
	for(i=0; i<m; i++){
		for(j=0; j<n; j++){
			ij = i*n + j;
			*(mm + ij) = i + j;
			printf("%d\t", *(mm + ij));
		}
		printf("\n");
	}
	free(mm);
}
```

3. [5 ptos.] Escriba un programa que muestre iterativamente el siguiente menú:

¿Desea ingresar una población?
1.- Sí.
2.- No.
Seleccione una de las opciones:

Si se selecciona la opción 1, se pide ingresar el nombre de un país y su población (en unidades de millones). Luego, se muestran todos los datos ingresados hasta ese momento. Finalmente, si se selecciona la opción 2, el programa debe terminar.

Sugerencia: Puede utilizar la siguiente main() o algo similar:

```
struct pais {
    char nombre[25];
    int poblacion;
};

void ingresar_pais(struct pais **lista, int *N);
void mostrar_pais (struct pais  *lista, int  N);
int main() {
    struct pais *lista = NULL;
    int opc, N = 0;
    do {
        printf("\n ¿Desea ingresar la población de un pais?\n 1.- Sí.\n 2.- No.\n Seleccione una de las opciones: ");
        scanf("%d", &opc);
        switch(opc)    {
            case(1): ingresar_pais(&lista, &N);                    break;
            case(2): printf ("\n Chau.\n\n");       free (lista);    break;
            default: printf ("\n Ingreso incorrecto.\n\n");
        }
    } while(opc != 2);
}
```

SOLUCION:

```
// CD3.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct pais {
        char nombre[25];
        int poblacion;
};

int leerString(char cc[ ], int n);
void ingresar_pais(struct pais **lista, int *N);
void mostrar_pais (struct pais  *lista, int  N);
int main() {
        struct pais *lista = NULL;
        int opc, N = 0;
        do {
                printf("\n ¿Desea ingresar la población de un pais?\n 1.- Sí.\n 2.- No.\n Seleccione una de las opciones: ");
                scanf("%d", &opc);
                switch(opc)    {
                        case(1): ingresar_pais(&lista, &N);                                     break;
                        case(2): printf ("\n Chau.\n\n");       free (lista);    break;
```

```
                        default: printf ("\n Ingreso incorrecto.\n\n");
                }
        } while(opc != 2);
}
void ingresar_pais (struct pais **lista, int *N){
        (*N)++;
        *lista = (struct pais *)realloc (*lista, *N*sizeof(struct pais) );
        if (*lista == NULL) {
                printf ("\n No hay más espacio en la memoria.\n\n");
                exit(EXIT_FAILURE);
        }
        struct pais *p = *lista + *N -1;
        printf ("\nIngrese el nombre del pais: ");
        getchar();
        leerString(p->nombre, 25);
        printf ("Ingrese la poblacion de dicho pais: ");
        scanf ("%d", &p->poblacion);
        mostrar_pais (*lista,  *N);
}

void mostrar_pais(struct pais *lista, int N) {
        struct pais *p = lista, *pmax = p+N;
        printf ("\nPoblaciones:\n");
        while(p < pmax){
                printf ("%s\t\t %d\n", p->nombre, p->poblacion);
                p++;
        }
}

int leerString(char cc[ ], int n){
   int c, m=0;
   while((c=getchar())!=10) if(m<n-1)cc[m++]= c;
   cc[m] = '\0';
   return m;
}
```

4. [5 ptos.]E scriba un programa que lea, desde el teclado, los datos de una variable *struct* **Estudiante** {nombre, codigo, nota[5]} calcule e imprima el promedio de las 5 notas de un estudiante.

SOLUCION:

```
// CD4.c
#include<stdio.h>
#include<string.h>
struct Estudiante{
        char nombre[30];
        char num_carnet[11];
        int nota[5];;
};
int leerString(char cc[ ], int n){
   int c, m=0;
   while((c=getchar())!=10) if(m<n-1)cc[m++]= c;
   cc[m] = '\0';
   return m;
}
void main(void){
```

```
	struct Estudiante est;
	int n, sum=0;
	float prom;
	printf("Nombre del estudiante: ");
	gets(est.nombre);
	printf("Numero de carnet: ");
	leerString(est.num_carnet, 11);
	printf("Las cinco calificaciones son:\n");
	for(n=0;n<5;n++){
		printf("Nota[%d]: ",n+1);
		scanf("%d",&est.nota[n]);
		sum+=est.nota[n];
	}
	prom=sum/5;
	printf("\nDATOS DEL ESTUDIANTE");
	printf("Nombre: %s\n",est.nombre);
	printf("Numero de carnet:%s\n",est.num_carnet);
	printf("Promedio:%f\n",prom);
}
```