12 de septiembre de 2023

Computación

Física

Facultad de Ciencias, U.N.A.M.

Grupo: 8415. Salón: Aula 2 de Computación, Física.
Prof.: Valente Vázquez
Ayte.: Natalia Baez

Examen Parcial 2

Instrucciones: Lee detenidamente las preguntas. Este examen se debe entregar hoy mismo antes de las 23:59 horas. Deberás responder este examen en computadora, el formato de entrega es PDF, no olvides escribir tu número. En algunas preguntas se necesitará operar según el número de cuenta, según el siguiente ejemplo: No. Cuenta: 123456789

Dato A: 1+2+3+4+5+6+7+8+9 = **45**
Dato B: (1+2+3+4+5+6+7+8+9)/4 = **12** (Redondeo hacía arriba si no es cerrada la división)

Rábago García Carlos Alexis 321099398

Dato A = 3+2+1+0+9+9+3+9+8 = 44 = 0010 1100
Dato B = A/4 = 11 = 0000 1011

1. (1 ptos) Contesta solamente tres de las siguientes preguntas:
   a. ¿Cuál es la diferencia entre un lenguaje interpretado y uno compilado?
   b. Explica la diferencia entre el símbolo “==” y “=”
      “=” se utiliza para asignarle algún valor a una variable, mientras que “==” compara el contenido de una variable con respecto a lo que se encuentre del otro lado del operador
   c. Menciona el tamaño de memoria de las siguientes variables:
      i. `char` 8 bits (1B)
      ii. `int` 32 bits (4B)
      iii. `float` 32 bits (4B)
   d. Explica con detalle las diferencias entre unsigned char y signed char
      Un tipo de dato sin signo, como unsigned char, no utiliza el bit extra reservado para determinar el signo, por lo que su máximo valor llega a ser el doble del valor absoluto del máximo de un signed char, i.e. que en vez de abarcar un rango de valores de [-128, 127], abarca el rango [0, 255].

2. (1 ptos) Respecto a los ciclos, contesta lo siguiente:
   a. Explica la diferencia entre while y do while
      `while` y `do-while` son ambos loops casi idénticos en su funcionamiento, con la diferencia de que mientras `while` sigue una estructura de evaluar la condición, ejecutar el código, y posteriormente reevaluar la condición para decidir si sigue iterando, `do-while` primero ejecuta el código al menos una vez antes de evaluar la condición para decidir si sigue iterando. Como consecuencia de esto, si la

condición es falsa desde el inicio, `while` nunca ejecutará el código mientras que `do-while` lo ejecutará al menos una vez.

b. Define que hacen las instrucciones *`break`* y *`continue`*, escribe un ejemplo distinto al que vimos en clase.

`break` y `continue` son palabras reservadas utilizadas en el contexto de loops para indicar un cambio en el control de flujo previamente determinado. En el caso de `break`, el programa al encontrarse con esta instrucción se saldrá inmediatamente del loop en el que break fue llamado y continuará con el código posterior al loop. En cambio, con `continue` el programa se saltará todo el código debajo de donde fue llamada la instrucción pero continuará con la siguiente iteración del loop si así la condición lo permite. Ejemplo:

```c
#include <stdio.h>

void main() {
   int input;
   printf("Input a number: ");
   scanf("%d", &input);
   for (int i = 0; i < 10; i++)
   {
       printf("Loop %d\n", i+1);
       if (input % 3 == 0)
       {
           printf("Number is divisible by 3\n");
           continue;
       }
       printf("Mid loop output\n");
       if (input % 2 == 0)
       {
           printf("Number is even\n");
           break;
       }
   }
}
```

Aquí si `input` es divisible entre 3, imprimirá “Loop 1”, seguido de “Number is divisible by 3”, se saltará el printf de “Mid loop output” y el if, y continuará iterando de la misma manera hasta el Loop 10. Si `input` es par, imprimirá primero el “Loop 1” seguido de el “Mid loop output”, y después “Number is even” pero ahí se saldrá completamente del `for` y no continuará iterando. Finalmente si `input` no es ni divisible entre 3 ni par, entonces únicamente iterará “Loop i” y “Mid loop output” hasta el Loop 10.

3. (2 pto) Resuelve las siguientes operaciones según tu número de cuenta:

Asumiendo que A y B son datos del tipo unsigned char (8 bits):

a) A&B

A	=	0010 1100	= 44
B	=	0000 1011	= 11
AND	=	0000 1000	= 8

b) A|B

A	=	0010 1100	= 44
B	=	0000 1011	= 11
OR	=	0010 1111	= 47

c) ¬A

A	=	0010 1100	= 44
NOT	=	1101 0011	= 211

d) ¬B

B	=	0000 1011	= 11
NOT	=	1111 0100	= 244

e) A^B

A	=	0010 1100	= 44
B	=	0000 1011	= 11
XOR	=	0010 0111	= 39

4. (3 ptos) Realiza los siguientes códigos:

a. Un programa en C que realice la suma de los primeros 100 números naturales

```c
#include <stdio.h>
void main() {
   int ans=0;
   for (int i = 1; i < 101; i++) {
       ans = ans + i;
   }
   printf("%d\n", ans);
}
```

b. Un programa en C que dibuje un tablero de ajedrez escribiendo solo una vez las impresiones de blanco, negro y salto de línea

```c
#include <stdio.h>
void main() {
   for (int i = 0; i < 8; i++)
   {
       for (int j = 0; j < 8; j++)
       {
           if ((i+j) % 2 == 0)
           {
               printf("\u2591");
           }
           else{
               printf("\u2589");
           }
       }
       printf("\n");
   }
}
```

c. Un programa en C que realice el promedio de 100 datos generados al azar entre 0 y 20, utilizando la menor cantidad de memoria posible, por ejemplo, sin utilizar arreglos ni declarar 100 variables.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main() {
   srand(time(NULL));
   short int i, a;
   for (i = 0; i < 100; i++)
   {
       a = a + (rand() % 20);
   }
   printf("%d\n", a/i);
}
```

5. (1.5 ptos) Contesta las siguientes preguntas:

a. ¿Qué es un puntero? Da un ejemplo

Un puntero es una variable que guarda como valor a la dirección de la memoria (en hexadecimal) donde se encuentra guardado el valor de otra variable. Ejemplo:

```c
#include <stdio.h>


void main(){
   int a = 32;
   int* a_ptr = &a;
   printf("a: %d\n", a);
   printf("a_ptr: %p\n", a_ptr);
}
```

En este ejemplo la variable `int a` guarda el valor entero de `32`, mientras que `int* a_ptr` contiene un valor de la forma `0x7fffffffdb3c` donde dicho valor es la dirección en la memoria donde se encuentra guardado el `32` de `int a`. Así, el output queda:

```
a: 32
a_ptr: 0x7fffffffdb3c
```

b. Completa las siguientes líneas de códigos y escribe el resultado según las siguientes variables:

```c
   int var1 = 3;
   int var2 = 2;
   float var3 = 22.45;
   char var4 = 'Z';
   int var5 = 45;
   int* var6 = &var5;

   printf("La suma de %d y %d es: %d\n", var1, var2, var1+var2);
   printf("La temperatura actual es: %.2f °C\n", var3);
   printf("La opción seleccionada es la opción: %c\n", var4);
   printf("La dirección donde se almacena el valor %d, es: %p\n", var5,
   var6);
```

Output:

```
La suma de 3 y 2 es: 5
La temperatura actual es: 22.45 °C
La opción seleccionada es la opción: Z
La dirección donde se almacena el valor 45, es: 0x7fffffffdb40
```

c. ¿Qué es un cast en C? Da un ejemplo

Un cast, o type-casting es la acción de cambiar el tipo de dato de un valor a otro tipo de dato distinto. Por ejemplo:

```c
#include <stdio.h>

void main(){
   int x_int = 62;
   float x_flt = (float) x_int;
   printf("x_int: %d\n", x_int);
   printf("x_flt: %f\n", x_flt);
}
```

Aquí `x_int` es el número entero 62, pero mediante un cast a float podemos asignarle a `x_flt` el valor de `x_int` correspondiente a un float, siendo este 62.000000. Así, el output termina siendo:

```
x_int: 62
x_flt: 62.000000
```

d. Explica la diferencia entre una variable global y una local, da un ejemplo con código.

Una variable global es aquélla declarada fuera de `main` y de todas las otras funciones del programa, y que puede ser utilizada desde donde sea, ya sea `main` o cualquier otra función. Una variable local es aquélla que fue declarada dentro de una función, ya sea `main` o cualquier otra. Éstas solo pueden ser utilizadas dentro de su propia función a menos que sean heredadas de una función a otra mediante los argumentos de la que recibe. Ejemplo:

```c
#include <stdio.h>

int global_a=1, global_b=2;

void func(int x, int y) {
   int local_c=3, local_d=4;
   printf("global_a = %d, global_b = %d, local_c = %d, local_d = %d, local_e = %d, local_f = %d\n", global_a, global_b, local_c, local_d, x, y);
}

int main() {
   int local_e=5, local_f=6;
   func(local_e,local_f);
   return 0;
}
```

Aquí `global_a` y `global_b` son variables globales disponibles en `func()` y `main()`, y utilizadas dentro de `func()`. Por otro lado `local_c` y `local_d` son variables locales declaradas y utilizadas únicamente dentro de `func()`, ya que son imposibles de utilizar dentro de `main()`. Finalmente `local_e` y `local_f` son variables locales declaradas en `main()`, pero pueden ser utilizadas en `func()` ya que son heredadas como argumentos de dicha función. Así, `func()` tiene acceso a todas las variables y puede imprimirlas sin mayor problema.

6. (1.5 ptos) Revisa los siguientes códigos y menciona cuál es el o los errores dentro de estos:

a.

```
#include <stdio.h>
void main() {
      unsigned int numeroA = 10;
      unsigned int numeroB = 20;

      printf("La resta es: %f.\n", numeroA - numeroB);
      return 0;
}
```

El error radica en que el printf llama a `%f` esperando un float o double, mientras que (`numeroA-numeroB`) es un unsigned int, es arreglado cambiando la línea 5 a: `printf("La resta es: %d.\n", numeroA - numeroB);` Además la línea 6 `return 0;` es innecesaria, ya que nuestra función `main` fue declarada como `void`. El código correcto queda como:

```
#include <stdio.h>
void main() {
   unsigned int numeroA = 10;
   unsigned int numeroB = 20;
   printf("La resta es: %d.\n", numeroA - numeroB);
}
```

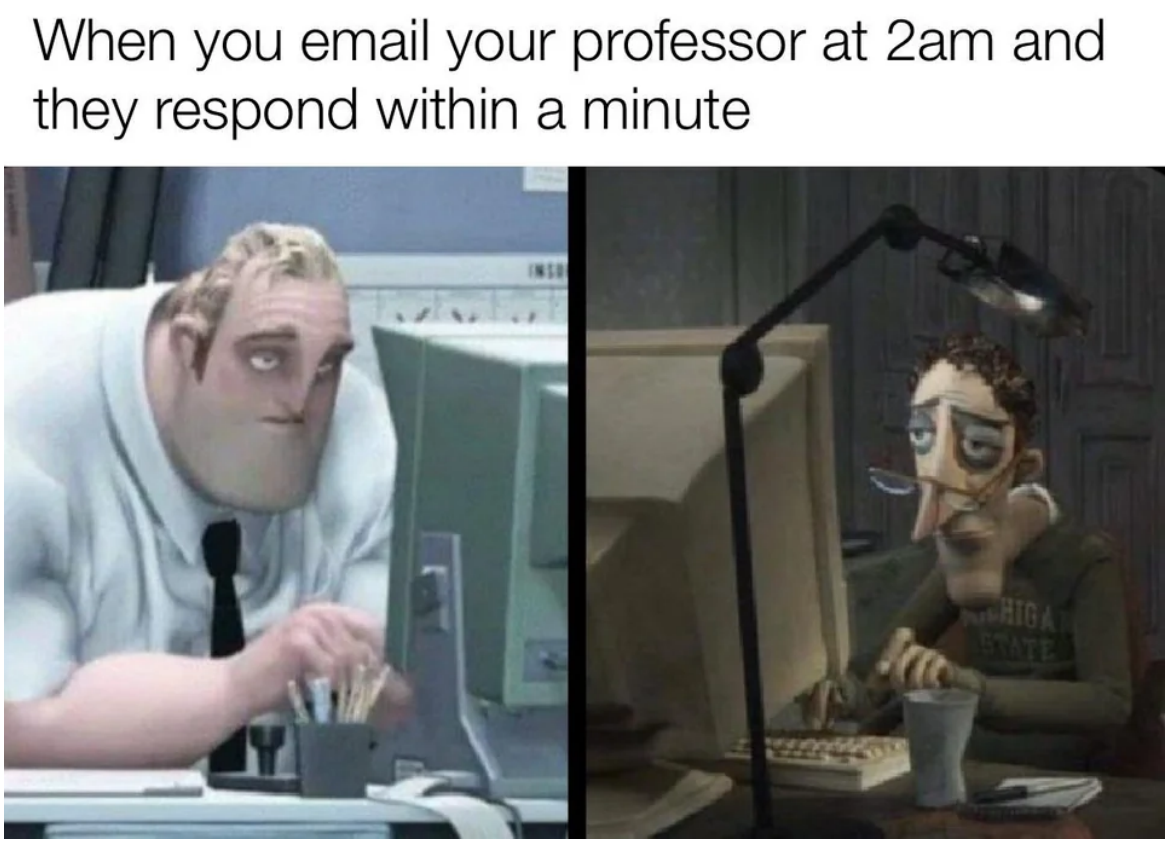
b.

```
#include <stdio.h>
int main() {
      int numero = 10;
      if (numero % 2 = 0) {
            printf("El número es par.\n");
      }
      else {
            printf("El número es impar.\n");
      }
      return 0;
}
```

El error aquí es en la línea 4, donde se usa el operador de asignación = en lugar del operador de comparación ==. El código correcto sería:

```c
#include <stdio.h>
int main() {
   int numero = 10;
   if (numero % 2 == 0) {
       printf("El número es par.\n");
   }
   else {
       printf("El número es impar.\n");
   }
   return 0;
}
```

7. (Punto extra) Inserta tu mejor meme, si nos hace reír tienes un punto extra.



8. (2 Puntos extra) Las tres primeras personas que envíen al grupo de Whatsapp el link de youtube con alguna de sus canciones favoritas, tienen dos puntos extras en este examen.

¡ÉXITO!