UNIVERSIDAD CHAMPAGNAT

Programación II

Práctico – Modularidad

1. Explique con sus palabras el concepto de “dividir para reinar”.

   Este paradigma, ¨ divide y vencerás ¨, separa un problema en subproblemas que se parecen al problema original, de manera recursiva resuelve los subproblemas y, por último, combina las soluciones de los subproblemas para resolver el problema original.

   Esto lo aplico normalmente para no confundirme con pasos y poder entender bien la consigna.

2. ¿Qué forma pueden tomar los diferentes módulos de un programa (según su función) ?

   -En C + +, se llaman funciones.
   -En JavaScript se denominan Script.
   -En Java, se llaman Module.
   -En Python, Module.
   -En Nodejs, Patrón De Sub Pila.
   -En PHP, Módulo, entre otros.

3. Que partes bien diferenciadas tiene el módulo de un programa ?

   -La definición de un módulo es donde se dice qué es lo que él mismo va a hacer, es decir, cuáles serán sus efectos al ser invocado, es decir cuando se ejecute. Una invocación o llamada ocurre cuando se usa el módulo, por ej, cuando hacemos la llamada de una función de un subproceso en el main principal.

4. ¿Cómo se llaman los módulos en lenguaje C ?

   -Procesos, funciones, subprocesos.

5. Mencione al menos cinco funciones predefinidas del lenguaje C.

   -strlen, strcpy, scanf, printf, srand.

6. En qué librería se encuentra cada una de las funciones del punto anterior ?

   -strlen y strcpy, se encuentran en la librería string.h
   -scanf y printf, se encuentran en la librería stdio.h y stdlib.h

-srand, se encuentra en la librería time.h

7. ¿Qué sucede cada vez que se invoca una función ?

   -Cuando invocamos una función, lo que hace el programa es identificar y leer todas las líneas de código dentro de la misma, y procede a ejecutarla, luego vuelve y espera nuevamente a ser ejecutado, una función también puede devolver un valor como salida.

8. Explique con ejemplos:

   a. ¿Qué son los parámetros formales ?

      -Los parámetros formales son variables locales a una función, ya que son creados al entrar en la función (se crean en la pila) y destruidos cuando ésta termina. (Es decir, aquellas variables que son definidas fuera del main, son como “copias”.)

   b. ¿Cuáles son los parámetros reales ?

      -Son la expresiones que se utilizan en la llamada de la función, sus valores se copian en los **parámetros** formales (.Los parámetros reales son aquellas variables que son definidas dentro del main,)

9. Explique. ¿Qué es un prototipo en C ?, ¿para qué sirve ?

   -Un prototipo es una declaración de una función. Consiste en una presentación de la función, exactamente con la misma estructura que la definición, pero sin cuerpo y terminada con un «;».

10. Explique “pasaje de parámetros por valor”, y “pasaje de parámetros por referencia”.

    -Pasaje por valor: en C la invocación de funciones implica que el paso de parámetros es, con carácter general y predefinido, por valor. De acuerdo con lo que indicamos cuando hablamos de pseudocódigo, esto impide que la variable transferida sea manipulada dentro de una función, ya que la función únicamente recibe un valor (no manipulable) copia del contenido de la variable.

    -Pasaje por referencia: al pasar como parámetros punteros en lugar de valores, se pueden modificar los valores señalados por los punteros (los parámetros reales), por lo tanto estos pueden ser modificados.

11. Teniendo en cuenta el punto anterior, de un ejemplo en lenguaje C de cada uno de los tipos de pasajes.

-Pasaje por valor:

```c
#include <iostream>
#include <stdio.h>

int pasajevalor (int nro);

int main (){
    int n1;
    printf ("Ingrese un numero. \n");
    scanf ("%d", &n1);
    printf ("El numero ingresado y multiplicado por 10 es: %d", n1 = pasajevalor(n1));
}

int pasajevalor (int nro){
    nro * 10;
    return nro;
}
```

-Pasaje por referencia:

```c
#include <iostream>
#include <stdio.h>

int pasajevalor (int *nro);

int main (){
    int n1;
    printf ("Ingrese un numero. \n");
    scanf ("%d", &n1);
    printf ("El numero ingresado y multiplicado por 10 es: %d", pasajevalor(&n1));
}

int pasajevalor (int *nro){
    *nro * 10;
    return *nro;
}
```

12. ¿Cuál es el ámbito de un parámetro formal definido en una función ?

    -El ámbito de un parámetro formal va a encontrarse solamente dentro de la función que estemos utilizando

13. ¿Cuál es el ámbito de una variable declarada fuera de una función ?

    -El ámbito de una variable declarada fuera de una función va a encontrarse en toda la ejecución global del programa, por lo tanto esta puede utilizarse dentro de todo el programa en cualquier lugar que se requiera.

14. ¿Qué produce el siguiente programa ?

    - El programa no compila, ni ejecuta, debido a que la variable b, no está declarada en el ámbito correspondiente de la función main.

```c
#include <stdio.h>

void main(){
    int a=0;
    if(a==0) {

        int b=1;
        printf("%d" , a);

    }

    printf("%d", b);

}
```

15. ¿Para que se utiliza *goto* en C ?

    -Es un comando utilizado en C, y sirve para saltar en forma incondicional a cualquier otra parte de la misma función.

16. ¿Qué produce el siguiente programa ?

    -El programa sirve para ingresar valores numéricos y que estos sean multiplicados según los valores ingresados y la cantidad de valores que se ingresan según la cantidad de números en la constante declarada como max Input. Además este programa nos da un promedio según los números ingresados, dividido a la cantidad de números que se solicitan en la constante.

    En este programa a su vez encontramos la sentencia goto JUMP, la cual le indica al programa que debe saltar a la función jump, y ejecutar las líneas de código dentro de ella.

```c
#include <stdio.h>

int main() {

  const int maxInput = 100;
  int i;
  double number, average, sum = 0.0;

  for (i = 1; i <= maxInput; ++i) {
    printf("%d. Enter a number: ", i);
    scanf("%lf", &number);

    if (number < 0.0) {
      goto jump;
    }
    sum += number;
  }

jump:
```

```
    average = sum / (i - 1);
    printf("Sum = %.2f\n", sum);
    printf("Average = %.2f", average);

    return 0;
}
```

17. Explique detalladamente qué es el tiempo de vida de una variable.

    -El tiempo durante el que una variable conserva su valor. El valor de una variable puede cambiar durante su duración, pero conservará algún valor. Cuando una variable pierde ámbito, ya no tiene un valor.