Laboratorio de Paradigmas de Programación

Práctica 1-3

Nombre: Daniel Cruz Enriquez
Matricula: 2001314h

1. A continuación se muestra la sintaxis de la sentencia `goto` en lenguaje C, realice algún código para familiarizarse con la forma de operación.

```
/* algún código */
/* ... */
etiqueta1:
/* ... */
goto etiqueta1;
```

2. Responda a las siguientes preguntas:

   2.1. ¿Es necesario que la etiqueta a la que se salta esté antes o después de la llamada a `goto`?

   La etiqueta puede estar antes o despues de la llamada

   2.2. ¿Es posible saltar a etiquetas fuera de la función en que se llama a `goto`?

   Si es posible

   2.3. ¿Qué sucede cuando se indica una etiqueta que no ha sido declarada?

   El goto no puede ir hacia esa etiqueta ya que se hace referencia pero no esta definida

3. Implemente el equivalente de las siguientes construcciones de C utilizando únicamente sentencias `if` y `goto`, donde cada `if` debe ser seguido inmediatamente de un `goto` y no se puede utilizar `else`. Es permitido utilizar `goto` sin necesidad de un `if` previo. Realice un programa para cada uno e incluya abundantes comentarios sobre la forma en que funciona:

   3.1. `while`

ProgramasParadigmas > C whileYgoto.c > ...

```c
#include <stdio.h>

int main(){
    int i=0;
    int n=0;
// implementacion normal de la funcion while

    while (i<5)
    {
        printf("que onda\n");

        i++;
    }
    printf("FIN while\n");

    // variante de while usando goto
    inicio:
    printf("hola\n");
    n++;

    if(n<5)
        goto inicio;


    printf("FIN goto\n");

    return 0;

}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
que onda
que onda
que onda
que onda
FIN while
hola
hola
hola
hola
hola
FIN goto
linuxroom@fedora:~/Documentos/ProgParadigmas$
```

3.2. for

ProgramasParadigmas > C forYgoto.c > ...

```c
#include <stdio.h>

int main(){
    int i;


    //implementacion normal de for:

    for (i = 0; i < 5; i++)
    {
        int multi= i*2;
        printf("multiplicación: %d\n",multi);
    }

    printf("fin FOR\n");
// implementacion de un for con goto
    int b=0;

    inicio:
    int mul = b*2;
    printf("multi %d\n",mul);
    b++;

    if (b<5)
        goto inicio;

     printf("fin GOTO\n");

    return 0;

}
```

```
multiplicación: 0
multiplicación: 2
multiplicación: 4
multiplicación: 6
multiplicación: 8
fin FOR
multi 0
multi 2
multi 4
multi 6
multi 8
fin GOTO
linuxroom@fedora:~/Documentos/ProgParadigmas$
```

3.3. do while

ProgramasParadigmas > C dowhileYgoto.c > main()

```c
#include <stdio.h>

int main(){
    int i=0;

    //implementacion normal de do while
    do
    {
        printf("WA SAAA\n");
        i++;
    } while (i<5);

    printf("fin do while\n");
    //implementacion de do while usando goto
    int a=0;
    inicio:
    printf("WA SAAA\n");
    a++;

    if (a < 5)
        goto inicio;  // vuelve al inicio


    printf("fin GOTO\n");

    return 0;
}
```

```
WA SAAA
WA SAAA
WA SAAA
WA SAAA
WA SAAA
fin do while
WA SAAA
WA SAAA
WA SAAA
WA SAAA
WA SAAA
fin GOTO
linuxroom@fedora:~/Documentos/ProgParadigmas$
```

3.4. switch

ProgramasParadigmas > C switchYgoto.c > main()

```c
#include <stdio.h>

int main(){
    int opcion = 2;

    switch (opcion) {
        case 1:
            printf("Opción 1 seleccionada\n");
            break;
        case 2:
            printf("Opción 2 seleccionada\n");
            break;
        case 3:
            printf("Opción 3 seleccionada\n");
            break;
        default:
            printf("Opción no válida\n");
            break;
    }
    printf("fin SWITCH\n");

    //implementacion de switch con goto

    if (opcion == 1) goto caso1;
    if (opcion == 2) goto caso2;
    if (opcion == 3) goto caso3;
    goto casoDefault;   // si no fue ninguna, va a default

caso1:
    printf("Opción 1 seleccionada\n");
    goto finSwitch;

caso2:
    printf("Opción 2 seleccionada\n");
    goto finSwitch;

caso3:
    printf("Opción 3 seleccionada\n");
    goto finSwitch;

casoDefault:
    printf("Opción no válida\n");
    goto finSwitch;

finSwitch:
    printf("Fin del switch GOTO\n");
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Opción 2 seleccionada
fin SWITCH
Opción 2 seleccionada
Fin del switch GOTO
linuxroom@fedora:~/Documentos/ProgParadigmas$
```

4. Responda a las siguientes preguntas:

   4.1. ¿Es posible implementar el equivalente a las funciones por medio de llamadas a `if` y `goto`? Argumente la razón de su respuesta y explique como funcionaría dicho equivalente en caso que juzgue sea posible.

   Es posible porque las llamadas a función se pueden simular usando los saltos de goto hacia un bloque de código,ejecutarlo y luego regresar a donde fue llamado.

ProgramasParadigmas > C funcionesYgoto.c > main()

```c
#include <stdio.h>

void saludo(){
    printf("hola k ase\n");
}

int main()
{
    saludo();
    printf("FIN\n");


    //implementacion de funcion usando goto

    goto llamar_saludo;   // simulación de la "llamada a función"

retorno1:                 // punto de regreso
    printf("Fin del programa\n");
    return 0;  // "return" hacia el punto de regreso


llamar_saludo:            // "cuerpo de la función"
    printf("Hola desde función con goto\n");
    goto retorno1;

}
```

```
hola k ase
FIN
Hola desde función con goto
Fin de GOTO
linuxroom@fedora:~/Documentos/ProgParadigmas$
```

4.2. ¿Qué ventajas tiene `goto`?

Permite ir y ejecutar partes de un programa sin ninguna condición directamente, esto puede ser útil cuando se quiere ejecutar una parte concreta de un código pero no se tiene ninguna otra forma de regresar a ella.

4.3. ¿Qué desventajas tiene `goto`?

Si se usa demasiado goto el programa se haría difícil de seguir y de entender por su capacidad de ir y ejecutar partes de código sin ninguna condición como **for** o **while**, lo que dificulta que otro programador o nosotros mismos ya pasado un tiempo entendamos que esta pasando en el código.