Tema2programacion.pdf

Julieta_G_V

Programación I

1º Grado en Ciberseguridad e Inteligencia Artificial

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Tema 2 programación

Introducción

Tipos de datos :

- int (%d) : números enteros
- double (%lg) : números decimales
- char (%c) : caracteres, es decir, letras y símbolos
- bool (%d, solo en printf) : valores true y false

Bibliotecas :

```
#include <stdio.h>
#include <stdbool.h>
```

Constantes y variables :

La información que conocemos a priori es aquella que incluimos en las constantes antes del int main() y que no cambia. Por el contrario, las variables almacenan información cuyo valor cambia durante el programa

```
int const centimos_en_1_euro=100; //constante antes del int m
int num; //variable dentro del int main()
```

Operadores :

- / : devuelve el cociente de la división
- % : devuelve el resto de la división
- && : conjunción (y)
- || : disyunción (o)

Asignación y expresiones aritméticas :

```
++x/x++ >> x=x+1
--x/x-- >> x=x-1
```

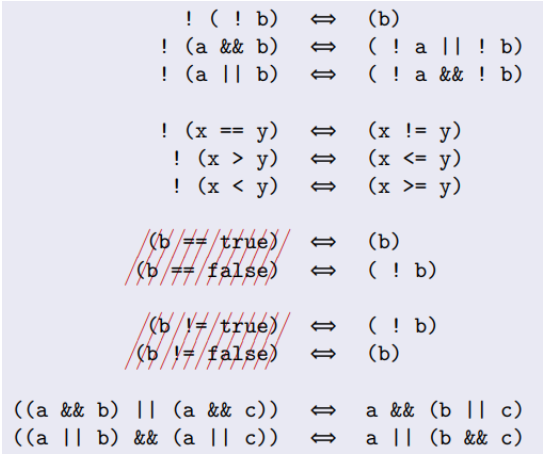
```
x+=(expr) >> x=x+(expr)
x-=(expr) >> x=x-(expr)
x*=(expr) >> x=x*(expr)
x/=(expr) >> x=x/(expr)
x%=(expr) >> x=x%(expr)
```

Casting (conversión del tipo de dato) :

```
//convertir la variable numero de int a double
int numero2 = (double)numero;

//convertir letras a numeros(ascii)
char letras_en_ascii = (int)letras
```

Expresiones lógicas. Equivalencias :



Sentencias de selección

- **if :**

```
if(expresion){
        instrucciones;
}
```

Si la condición se cumple, se ejecutan las instrucciones. Si no, no se ejecuta nada.

¿DÍA DE CLASES infinitas?

WRIGLEY'S Orbit

masca y fluye

Programación I

Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

Si ponemos varios if uno detrás de otro, pueden cumplirse varios de ellos a la vez.

- **if-else :**

```
if(expresion){
        instrucciones;
} else {
                instrucciones2;
                }
```

Si la condición se cumple, se ejecutan las instrucciones en el if. Si no, se ejecutan las instrucciones en el else.

- **if-else-if :**

```
if(expresion){
        instrucciones;
} else if (expresion2){
                    instrucciones2;
                }else if (expresion3){
                                instrucciones3;
                            } ........
```

Se van evaluando las expresiones en el orden indicado hasta que una de ellas se cumpla y entonces se ejecutan las instrucciones indicadas. Entonces, el resto de expresiones no se evalúan.

Esta sentencia es exclusiva, es decir, no pueden cumplirse dos condiciones a la vez.

- **switch :**

```
switch(expresion_a_sustituir){
case cte1:
        instrucciones1;
        break;
case cte2;
        instrucciones2;
        break;
...
```

```
default://si no se cumple ninguno de los casos especificados,
        instrucciones_otros_casos;
        break;
}
```

No se pueden declarar variables dentro del switch.

Operador condicional ternario (?:)

```
int mayor=(x>y)?x:y; //ejemplo
```

El operador condicional ternario (cond ? op1 : op2) evalúa la condición. Si es verdadera, el resultado de la expresión es el valor de op1. En otro caso, el resultado es el valor de op2.

Sentencias de iteración. Bucles

Si el número de iteraciones a realizar por el bucle es predeterminado, se utiliza el bucle for. Si el número de iteraciones es no predeterminado, se utilizan bucles while y do-while.

La condición de control del bucle debe tener un valor de inicialización.

- **Bucle “for” :**

```
for(inicialización; condición; incremento){
        instrucciones;
}

//ejemplo o caso común:

for(i=0; i<=num; i++){
        instrucciones;
}
```

Si queremos que en cada iteración se nos muestre la actualización, el printf debe ir dentro del bucle. Si sólo queremos que se nos muestre el resultado final, el printf debe ir fuera del bucle.

No se debe modificar la variable de control en el cuerpo del bucle.

- **Bucle “while” :**

```
while(condición){
        instrucciones;
}

//ejemplo o caso común:

while(num>0){
        instrucciones;
}
```

En este bucle, mientras se cumpla la condición establecida, el bucle seguirá ejecutándose. Cuando esta condición deje de cumplirse, el bucle se finaliza.

- **Bucle “do-while” :**

```
do{
        instrucciones;
} while(condición);

//ejemplo o caso común:

{ int i=1;
    do{
            instrucciones;
            i++;
}while(i<=10);
}
```

En este bucle, primero se ejecutan las acciones introducidas y después se comprueba si se cumple la condición. Hace la misma función que el bucle while.

Si hay que asignarle un valor inicial a la variable del bucle, éste se asigna antes del do.

Salida formateada de datos

- Se puede especificar la anchura de campo:

```
printf(" %05d\n", numero) //muestra 00numero (anchura de campo
```

Errores frecuentes

- Denominador=0
- Uso de = en lugar de ==
- =+ y =-no existe >> += y -=
- 0≤x≤9 no existe >> 0≤x && x≤9