Apuntes OIFem II Nivel 1

Condicionales y bucles

Condicionales

Existen diferentes tipos de condicionales en C++: "if", "else", "else if" y "switch".

if

La instrucción "if" ejecuta una instrucción solo cuando una condición es verdadera. Por ejemplo, se podría entender como: "si $2 + 2 = 4$, es verdadero".

Código-if

```cpp
#include <iostream>

using namespace std;

int main() {
    int x, y;
    cin >> x >> y;
    if (x > y) { //comprobamos la condición que x es mayor que y
        cout << x << " es mayor que " << y << '\n';
    }
}
```

else

La instrucción "else" se ejecuta solo cuando una condición es falsa.

Código-else

```cpp
#include <iostream>

using namespace std;

int main() {
    int x, y;
    cin >> x >> y;
    if (x > y) { //comprobamos la condición que x es mayor que y
        cout << x << " es mayor que " << y << '\n';
```

```
    }
    else { //la condición es falsa, por lo tanto x NO es mayor que y
        cout << x << " es menor o igual que " << y << '\n';
    }
}
```

else if

La instrucción "else if" especifica una nueva condición solamente si la primera es falsa. Se escribe después de un "if" pero nunca de un "else".

Nota: Si en un condicional escribimos la instrucción "else", entonces sabemos seguro que el programa habrá entrado en al menos uno de los bloques, mientras que si lo finalizamos con un "else if", podría pasar que nos hubiéramos dejado un caso y el programa no haría nada para eso.

Código-else if

```
#include <iostream>

using namespace std;

int main() {
    int x, y;
    cin >> x >> y;
    if (x > y) { //comprobamos la condición1: si x es mayor que y
        cout << x << " es mayor que " << y << '\n';
    }
    else if (x == y) { //la condición1 es falsa, por lo tanto x NO es mayor que y.
    ↪  Entonces comprobamos la condición2: si x es igual a y
        cout << x << " es igual a " << y << '\n';
    }
    else { //tanto la condición1 como la condición2 son falsas, por lo tanto, x NO
    ↪  es mayor que y, y además, x NO es igual a y
        cout << x << " es menor que " << y << '\n';
    }
}
```

switch

La instrucción "switch" se usa para ejecutar uno de diversos casos posibles. Se inicia con una expresión, que se compara después con los valores de los posibles casos. Si son iguales, se ejecuta el código de ese bloque. Al final de cada caso escribimos la instrucción "break" para que no se ejecuten los siguientes casos. Puede haber un caso "default", que se ejecuta cuando ningún otro caso es seleccionado.

Código-switch

```
#include <iostream>

using namespace std;

int main() {
```

```
    int day;
    cin >> day;
    switch (day) {
        default:
            cout << "Dias laborables";
            break;
        case 6:
            cout << "Sabado";
            break;
        case 7:
            cout << "Domingo";
            break;
    }
}
```

Bucles

Los bucles ejecutan el código mientras que una condición específica se cumpla.

while

El bucle "while" ejecuta el código mientras que la condición especificada sea verdadera.

Nota: Un error común es olvidar de aumentar el valor de la variable que itera en el bucle, por lo que el bucle nunca acabaría.

Código-while

```
#include <iostream>

using namespace std;

int main() {
    int i = 0;
    while (i<5) { //entra en el bucle cuando i=0, y entonces se repite con i=1,
    ↪  i=2, i=3 y i=4
        cout << i << '\n';
        ++i;
    }
}
```

do-while

La instrucción "do/while" es una variante del "while". Se diferencian porque el "do/while" siempre entrará al menos una vez dentro del bucle y después se comprueba la condición, mientras que "while" solo se ejecuta cuando la condición especificada sea verdadera, por lo que no entrará en el bucle si esta no se cumple.

Código-do/while

```cpp
#include <iostream>

using namespace std;

int main() {
    int i = 6;
    do {
        cout << i << '\n';
        ++i;
    } while (i<5); //el bucle se ejecutará una sola vez ya que la condición se comprueba al final
}
```

for

El bucle "for" se usa cuando sabemos cuántas veces se debe recorrer el bloque de código dentro del bucle. Se declaran tres instrucciones junto con el "for". La instrucción 1 se ejecuta solo una vez, al inicio del bucle. La instrucción 2 define la condición para ejecutar el bloque de código. La instrucción 3 se ejecuta cada vez que se haya ejecutado el bloque de código.

Código-for

```cpp
#include <iostream>

using namespace std;

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cout << i << '\n'; //se imprimen los números de 0 a n-1
    }
}
```

Nota: Podemos cambiar la instrucción 3 del bucle "for" para que se incremente con cualquier valor. Es decir, en vez de escribir $++i$ podemos escribir $i = i + 3$ y entonces la variable i aumentará de 3 en 3.

Ejemplo práctico: NOCHEVIEJA

Ramón se pasa el día de Nochevieja contando los minutos que faltan para que den las uvas. ¿Puedes ayudarle?

ENTRADA: La entrada consta de una única línea que contiene un número n, $0 \leq n \leq 5$, que indica cuántos mensajes hay que emitir.

SALIDA: Para cada caso de prueba se mostrará una línea con el número de minutos que faltan para medianoche.

Link al problema: Nochevieja

Solución

Calculamos cuántos minutos quedan para las 00 computando los minutos que tiene un día y restando los que ya han transcurrido.

```cpp
#include <iostream>
using namespace std;

int main() {
        int hora, min;
        char c;
        while (cin >> hora >> c >> min) {
                if (hora == 0 and min == 0) {
                        break;
                } else {
                        cout << (24*60)-(hora*60+min) << '\n';
                }
        }
}
```

Bucles anidados

Un bucle anidado es un bucle que se encuentra dentro del bloque de código que se ejecuta. Puede existir cualquier nivel de anidamiento.

Nota: Dentro de los bucles anidados, no podemos utilizar la misma variable en la declaración del iterador.

Código-for

```cpp
#include <iostream>

using namespace std;

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j <= i; ++j) {
            cout << j << ' '; //en cada nueva línea, imprimimos los números de 0 a
            ↪ i
        }
        cout << '\n';
    }
}
```

Ejemplo práctico: Chess 1

Considere un tablero de ajedrez con r filas y c columnas, donde cada casilla contiene entre 0 y 9 monedas. Escribe un programa tal que, dado un tablero de ajedrez, calcule el número total de monedas que tiene.

ENTRADA: La entrada comienza con el número de filas r y el número de columnas c. Se siguen r líneas, cada una con c caracteres entre 0 y 9.

SALIDA: Imprime el número total de monedas en el tablero.

Link al problema: Chess 1

Solución

Leemos los números del tablero por filas, y separamos los números utilizando la función "módulo" y la división.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int r, c;
    int total = 0;
    cin >> r >> c;
    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < c; ++j) {
            char k;
            cin >> k;
            total += k - '0';
        }
    }
    cout << total << '\n';
}
```

Ahora veremos variaciones de este problema: contaremos las monedas de solo algunas casillas del tablero de ajedrez (las casillas blancas, la diagonal).

Ejemplo práctico: Chess 2

Considere un tablero de ajedrez con r filas y c columnas, donde cada casilla contiene entre 0 y 9 monedas. Escribe un programa tal que, dado un tablero de ajedrez, calcule el número total de monedas que tiene en las **casillas blancas**.

Considere que la casilla de arriba a la derecha es siempre blanca.

ENTRADA: La entrada comienza con el número de filas r y el número de columnas c. Se siguen r líneas, cada una con c caracteres entre 0 y 9.

SALIDA: Imprime el número total de monedas en las casillas blancas del tablero.

Link al problema: Chess 2

Solución

Usamos el mismo razonamiento que en el problema anterior (Chess 1) para leer las monedas de cada casilla. Ahora, solo debemos contar las monedas en las casillas blancas. Observamos que la paridad de la suma de los iteradores es 0 cuando señalan una casilla balnca y 1 cuando es negra. Por lo tanto, añadimos la condición de que módulo 2 deben sumar 0.

```cpp
#include <iostream>
using namespace std;
```

```cpp

int main()
{
    // Contar el número de monedas en las casillas blancas del tablero.
    // B N B N ...
    // N B N B ...
    int r, c;
    int total = 0;
    cin >> r >> c;
    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < c; ++j) {
            char k;
            cin >> k;
            if ((i - j) % 2 == 0)
                total += k - '0';
        }
    }
    cout << total << endl;
}
```

Ejemplo práctico: Temperatures

Escriba un programa que lea un número entero que represente una temperatura dada en grados Celsius y que indique si hace calor, si hace frío o se está bien. Suponga que hace calor si la temperatura es superior a 30 grados, que hace frío si la temperatura es inferior a 10 grados y que, en caso contrario, se está bien. Además, avise si con la temperatura indicada el agua hierve, o si el agua se congela. Suponga que el agua hierve a 100 grados o más y que el agua se congela a 0 grados o menos.

ENTRADA: La entrada consta de un número entero.

SALIDA: Imprime una línea que indique si hace calor, si hace frío o si está bien. Además, imprima otra línea si el agua hierve o si el agua se congela.

Link al problema: Temperatures

Solución

Hacemos un programa con `if`. Si la temperatura es mayor que $30°C$, entonces escribimos *it's hot*, y además tenemos que comprobar si la temperatura es mayor o igual a $100°C$, y entonces escribimos *water would boil*.

Utilizamos un `else if` y por lo tanto se asegura que la temperatura es menor o igual a $30°C$. Ahora debemos comprobar si es menor que $10°C$ y escrbir *it's cold*, y también mirar si la temperatura es menor o igual que $0°C$ y escribir *water would freeze*.

Ahora, como utlizamos la instrucción `else`, tenemos que la temperatura es menor o igual a $30°C$ (porque no ha entrado en el primer `if`) y que la temperatura es mayor o igual a $10°C$ (porque no ha entrado en el segudo `else if`). Así pues, este es el último caso, y escribimos *it's ok*.

```cpp
#include <iostream>
using namespace std;

int main()
```

```
{
    int x;
    cin >> x;
    if (x > 30) {
        cout << "it's hot" << endl;
        if (x >= 100)
            cout << "water would boil" << endl;
    } else if (x < 10) {
        cout << "it's cold" << endl;
        if (x <= 0)
            cout << "water would freeze" << endl;
    } else
        cout << "it's ok" << endl;
}
```

Ejemplo práctico: First cubes

Escriba un programa que lea un número n e imprima 0^3, 1^3, ..., $(n-1)^3$, n^3.

ENTRADA: La entrada consta de un número natural n.

SALIDA: Imprima una línea con 0^3, 1^3, ..., $(n-1)^3$, n^3. Separe los números con comas.

Link al problema: First cubes

Solución

Nos dan el número de cubos que debemos escribir, por lo tanto hacemos un `for` y escribimos el cubo del iterador. Debemos vigilar cuando escribir una coma o no: en el último caso no tiene que ir coma.

```
#include <iomanip>
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    for (int i = 0; i <= n; ++i) {
        cout << i * i * i;
        if (i != n)
            cout << ',';
    }
    cout << endl;
}
```