Teoria (while)

Para que el programa realice una acción mientras existe alguna condición, se utiliza un bucle `while`.

```cpp
while (hay en la heladera torta) {
    ir a la heladera;
}
```

Digamos que el usuario ingresa el número `n`. Entonces, con un bucle `while`, puede calcular la suma de números de `1` a `n`.

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

int main() {
    int n;
    cin >> n;
    int sum = 0;
    int i = 1;
    while (i ⩽ n) {
        sum += i;
        ++i;
    }
    cout << sum << endl;
}
```

El siguiente programa usa un ciclo `while` para encontrar el `mcd`, el máximo común divisor de dos enteros positivos `a` y `b`. El cuerpo del ciclo se repite hasta que una de las variables `a` o `b` se vuelve cero. La segunda variable contendrá el valor mcd:

```cpp
#include <iostream>

using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    // Los numeros a y b deben ser naturales
    while (b ≠ 0) {
        int c = b;
        b = a % b;
        a = c;
```

```
    }
    cout << a << endl;
}
```

Teoria (do-while)

El ciclo `while` tiene su analogo `do-while`:

```language-cpp
do {
    ir a la heladera;
} while (hay en la heladera torta);
```

En un bucle `do-while`, ejecuta el cuerpo del bucle al menos una vez, ya que al final se cumple la condición de continuar el bucle. En un bucle `while`, la condición de continuación del bucle se comprueba desde el principio, por lo que es posible que el cuerpo del bucle no se ejecute ni una sola vez si la condición es inicialmente `falsa`.

Mire el fragmento de código de un juego interactivo en el que el usuario adivina un número:

```language-cpp
int secret = 5; // el numero secreto
int answer;     // respuesta del usuario
do {
    cout << "Guess the number: "s << endl;
    cin >> answer;
} while (answer ≠ secret);
cout << "You are right!"s << endl;
```

El usuario ingresa un número y el código repite la operación hasta que se adivina el número. Si el usuario adivina el número, el programa le dirá: `“You are right!”`.
Una vez que comprenda el ciclo `while`, puede pasar al ciclo `for`.

Teoria

A veces hay situaciones en las que al programador le gustaría detener la iteración actual del bucle y pasar al siguiente paso. Para decirle al programa que `"vaya al siguiente paso en el ciclo"`, use la instrucción `continue`. En el siguiente ejemplo, el ciclo debe ejecutarse `num_iters` veces. Al comienzo de cada iteración, el usuario ingresa los índices de caracteres de la cadena `str` desde el teclado. El programa debe comparar los caracteres en los índices especificados y mostrar el resultado en la pantalla. Pero si la entrada no es válida, es decir, los índices ingresados son negativos o mayores que la longitud de `str`, entonces no se puede comparar nada y solo necesita pasar a la siguiente iteración del ciclo:

```language-cpp
string str = "Drawing indices for fun and profit"s;
int num_iters = 0;
```

```
// leemos cuantas veces quisieramos repetir el ciclo
cin >> num_iters;

for (int i = 0; i < num_iters; ++i) {
    int index1, index2;
    // pedimos los indices al usuario
    cin >> index1 >> index2;

    // si index1 negativo o mayor que la longitud de nuestra string
    // entonces continuar con este paso del ciclo no es posible
    if (index1 < 0 || index1 ⩾ str.size()) {
        // al ordenar continue, el programador pide pasar a la
        // siguiente iteración (paso) sin terminar la actual
        continue;
    }

    // la misma logica para el index2
    if (index2 < 0 || index2 ⩾ str.size()) {
        continue;
    }
    // muestra el resultado de comparar los caracteres de la cadena
    // en los índices especificados
    cout << (str[index1] == str[index2]) << endl;
}
```

Este código sin `continue` seria muy difícil de manejar.

Pero a veces se desea no solo pasar a la siguiente iteración, sino salir completamente del ciclo. Para finalizar un ciclo antes de tiempo, use la instrucción `break`. Así es como se verá un programa que imprime el índice de la primera letra `a` en la palabra ingresada. En este caso, la línea `Yes!` se imprimirá en cualquier caso, si se encuentra la letra `a` y si no.

language-cpp
```
#include <iostream>
#include <string>

using namespace std;

int main() {
    string animal;

    // leemos el nombre del animal
    cin >> animal;

    for (int i = 0; i < animal.size(); ++i) {
        // si la letra actual de la string es a,
```

```
        if (animal[i] == 'a') {
            // entonces mostramos el indice i en la pantalla
            // y se termina el ciclo
            cout << i << endl;
            break;
        }
    }

    cout << "Yes!"s << endl;
}
```