

# Algoritmos y Estructura de Datos I

## Taller de Debugging

Departamento de Computación, FCEyN, Universidad de Buenos Aires.

16 de Abril de 2018

# Motivación... ¿Cuál es el error?

## Ejercicio

Dado un string *s*, devolver el *i*ésimo substring que se encuentre entre dos delimitadores.

```
indice_substring("esto#es#un#ejemplo", '#', 2)
```

devuelve el string "un".

```
indice_substring("#esto#es#un#ejemplo", '#', 2)
```

devuelve el string "es".

```
indice_substring("#esto#es#un#ejemplo", '#', 0)
```

devuelve el string vacío.

```
indice_substring("#esto##es#un#ejemplo", '#', 2)
```

devuelve el string vacío.

## Motivación... ¿Cuál es el error?

```
string indice_substring(string input_string, char delim, int ind)
/* Inicialización de variables */
while(i < s.size() && !encontre) {
    if(s[i] != delim){
        j = i;
        while(j<s.size() && s[j] != delim){
            res.push_back(s[j]);
            j++;
        }
    }
    if (ind==actual){
        encontre = true;
    } else {
        res.clear();
        i = j + 1;
        actual++;
    }
}
if (encontre || ind == actual)
    return res;
else
    return "Indice fuera de rango";
}
```

# Debugging

## ¿Qué es un bug?

Error de software que produce un resultado o un comportamiento indeseado.

## ¿Cómo encontramos un bug?

Tests! (más en próximas clases).

## ¿Qué es debuggear (en español, depurar)?

Realizar un seguimiento línea por línea de nuestro programa usando un debugger.

## ¿Y qué es un debugger?

Es un programa que toma como entrada otro programa (un binario) y nos permite controlar el flujo de ejecución. En CLion (y en todas las IDEs) es una herramienta que ya viene incorporada.

# ¿Hace falta?

## Alternativa 1: Imprimir por pantalla los resultados parciales.

A favor:

- ▶ Nos da una idea del recorrido del programa.

En contra:

- ▶ Proceso que consume mucho tiempo: agregar el código necesario, recompilar todo, correr el programa y analizar la salida. Todo eso cada vez que encontramos un bug.
- ▶ Ensuciamos el código.
- ▶ Al final hay que borrar todo lo que agregamos.

## Alternativa 2: Tratar de encontrar el error mirando el código.

A favor:

- ▶ Más rápido.

En contra:

- ▶ Para algoritmos no triviales, es difícil darse cuenta.

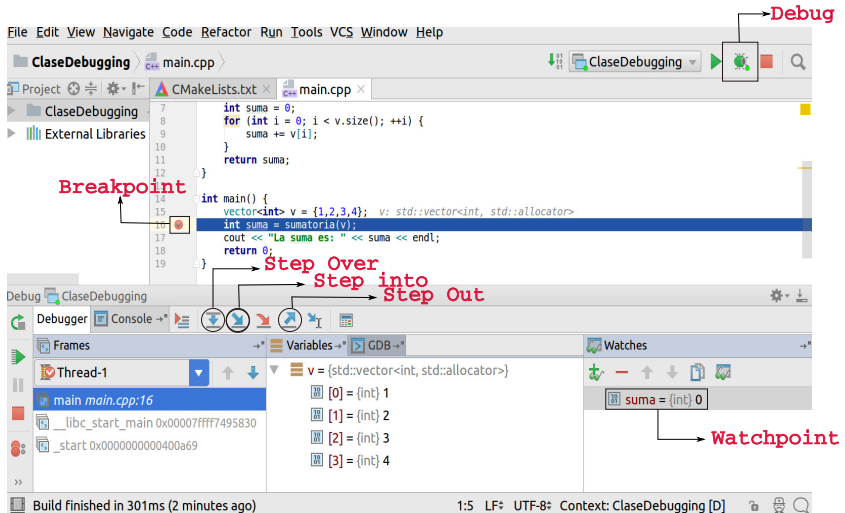
# Cómo lo usamos

1. Compilamos en modo debug. Esto introduce en nuestro programa símbolos especiales para ser usados por el debugger.
2. Corremos el programa con el debugger.

Conceptos fundamentales:

- ▶ *Breakpoint*: Suspender la ejecución del programa en una línea en particular.
- ▶ *Step Over*: Ir hacia la siguiente línea.
- ▶ *Step into*: Meterse dentro de una función.
- ▶ *Step out*: Salir de una función.
- ▶ *Watchpoint*: Hacer el seguimiento de una variable durante el transcurso de una función/programa.
- ▶ *Stacktrace/frames*: Ver en orden todas las funciones que fueron invocadas hasta el momento.

# En CLion



# Asserts

- ▶ Es una función para indicarle al programa que aborte la ejecución cuando no se cumple una condición.

```
1 | #include <assert.h>
2 | int primero(vector<int> v){
3 |     assert(v.size() > 0);
4 |     return v[0];
5 | }
```

Obtenemos el error:

```
int primero(std::vector<int>): Assertion 'v.size() > 0' failed.
```



