# Laboratorio de Programación - Labo04

## Debugging en C++

Algoritmos y Estructuras de Datos I

Departamento de Computación, FCEyN, Universidad de Buenos Aires.

# Motivación... ¿Cuál es el error?

## Ejercicio

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

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

devuelve el string “un”.

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

devuelve el string “es”.

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

devuelve el string vacío.

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

devuelve el string vacío.

# Motivación... ¿Cuál es el error?

```
1   string indice_substring(string s, char delim, int ind) {
2     int i = 0; int j = 0; int actual = 0;
3     bool encontre = false; string res = "";
4     while (i < s.size() && !encontre) {
5       if (s[i] != delim) {
6         j = i;
7         while (j < s.size() && s[j] != delim) {
8           res.push_back(s[j]);
9           j++;
10        }
11      }
12      if (actual == ind) {
13        encontre = true;
14      } else {
15        res.clear();
16        i = j + 1;
17        actual++;
18      }
19    }
20    if (encontre || ind == actual)
21      return res;
22    else
23      return "Indice fuera de rango";
24  }
```

# Debugging

## ¿Qué es un bug?

# Debugging

## ¿Qué es un bug?

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

## ¿Cómo encontramos un bug?

# 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).

# 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)?

# 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.

# 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?

# 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.

# ¿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.

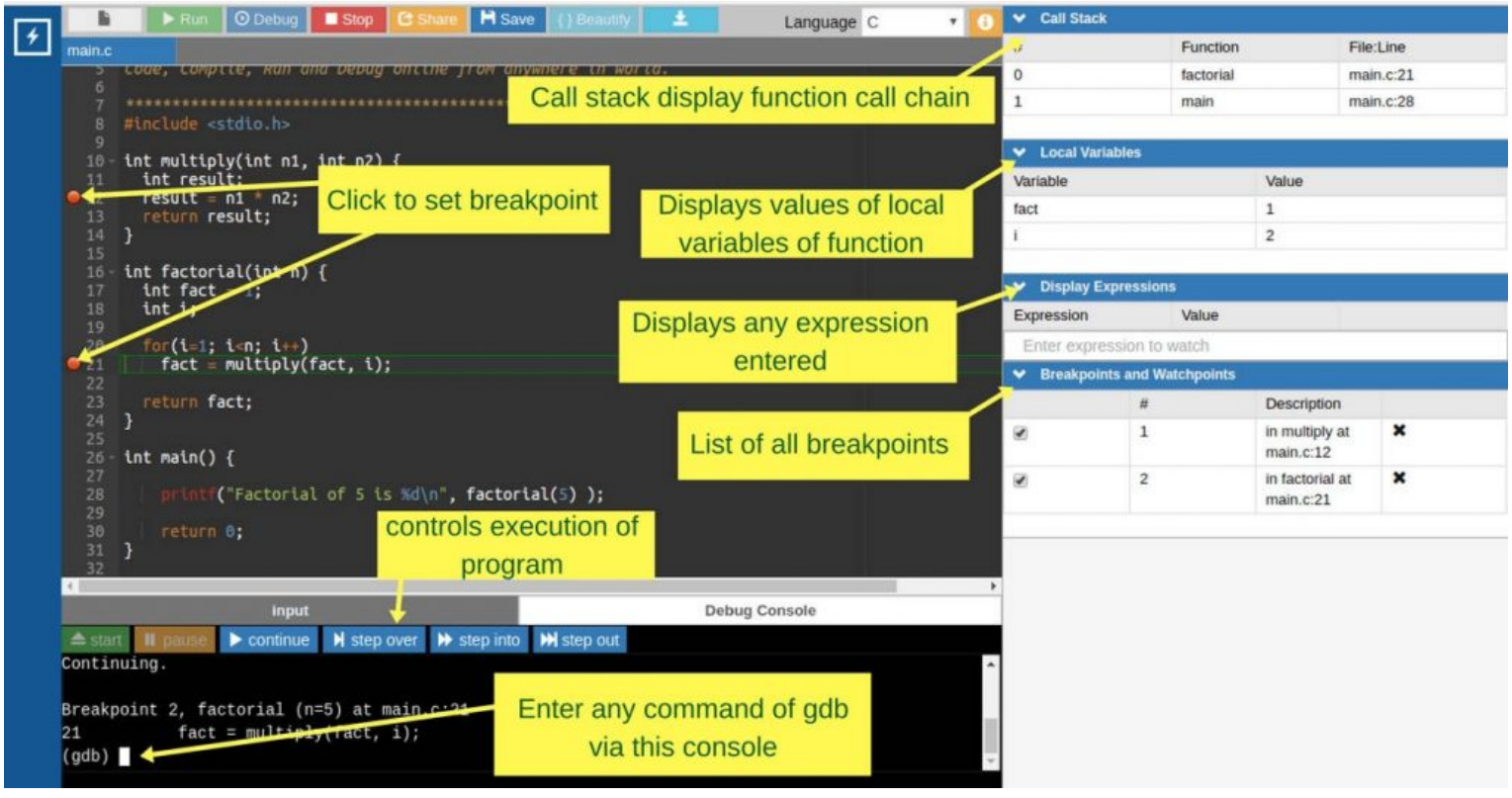
# Cómo lo usamos

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.

# IDE online



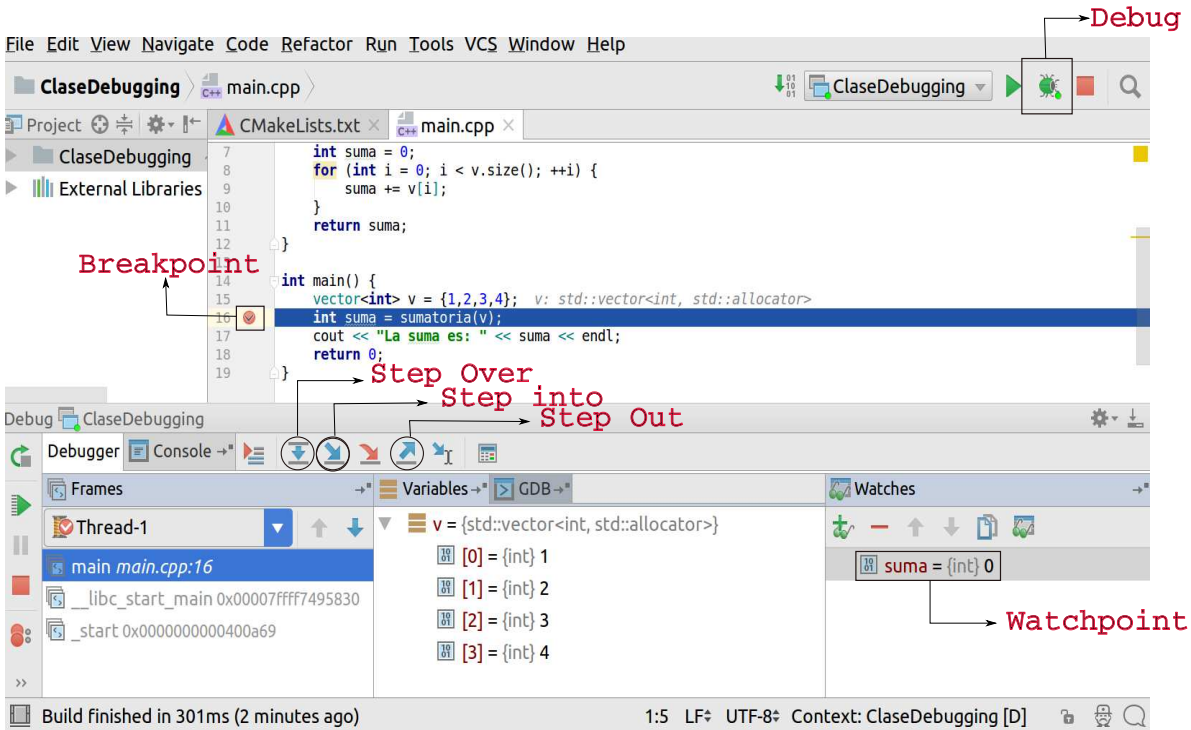

Esta imagen proviene de `https://www.onlinegdb.com/blog/brief-guide-on-how-to-use-onlinegdb-debugger/`. Podemos ver algunas de las herramientas de debugging que contiene esta IDE.

# Comandos GDB

| Comando | Description |
|---|---|
| r | Ejecutar el programa hasta un breakpoint o el final |
| b fun | Fijar un breakpoint al inicio de la función "fun" |
| b N | Fijar un breakpoint en la linea N |
| b file.c:N | Fijar un breakpoint en la linea N del archivo "file.c" |
| d N | Borrar breakpoint numero N |
| info break | Listar breakpoints |
| c | Continuar la ejecución hasta un breakpoint o el final |
| f | Ejecutar hasta finalizar la función actual |
| s | Ejecutar la próxima linea |
| s N | Ejecutar próximas N lineas |
| n | Como s pero no entra en las funciones |
| p var | Imprime valor actual de variable "var" |
| set var=val | Asigna valor "val.ª la varible "var" |
| bt | Imprime stack trace |
| q | Exit gdb |

Esta tabla proviene de `http://www.gdbtutorial.com/tutorial/commands`, donde además se pueden encontrar ejemplos simples de uso de GDB.

# En CLion




La figura muestra algunas herramientas de CLion para el debugging. Existen muchas otras para evaluación de expresiones durante el funcionamiento, breakpoints condicionales (se puede poner una guarda para detener el programa, que es muy util cuando se trabaja con iteraciones largas), y otras herramientas que pueden consultar en la guía online de CLion `https://blog.jetbrains.com/clion/2015/05/debug-clion/`.