

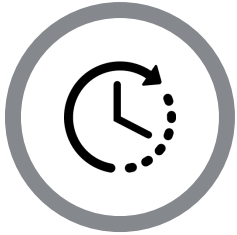
Lenguajes y Paradigmas de Programación

2022/01

Ma. Loreto Arriagada

loreto.arriagada.v@edu.uai.cl

Acuerdos de la clase



RESPETO POR EL TIEMPO
DE INICIO Y TÉRMINO



ALCEN LA MANO
PARA HABLAR



TRABAJEN SOLO
CON LO DE LA CLASE



EL FEEDBACK ES
BIENVENIDO



PARTICIPEN EN
LA CLASE



USE CONSCIENTEMENTE



EXPRESEN SUS IDEAS
LIBREMENTE

Discusión

Dijkstra vs GoTo

Problemas del GoTo

Afecta dos elementos fundamentales:

1. Probar que un fragmento de programa es correcto y
2. Describir lo que un programa ha hecho hasta ahora.


Analicemos

GoTo

```
top: if (n == 0) goto bottom;  
/* do something */  
goto top;  
bottom:
```

Estructuras de Control

```
while (n != 0) {  
    /* do something */  
}
```

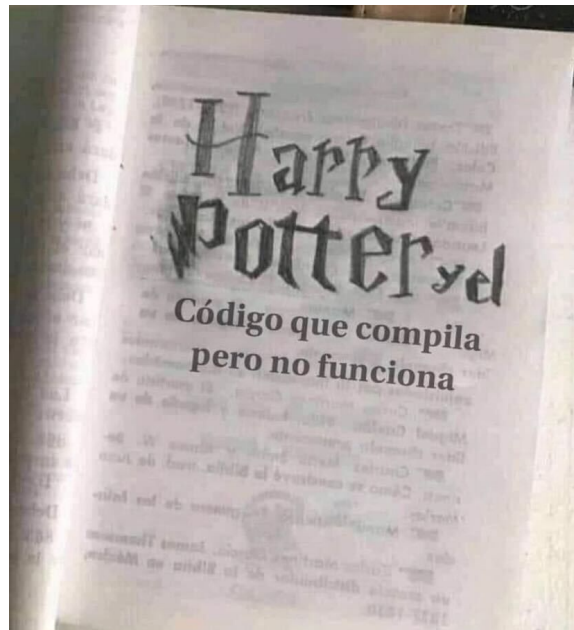


El problema no es la forma de escribirlo, es que cualquier otra instrucción puede usar el label bottom para llegar a ese mismo lugar en el código. Entonces, en este momento, debo preguntarme si el código viene del loop o de otro lado → complejidad extra

¿Qué pasa con?

- ¿Instrucciones cómo break, continue en un ciclo?
- ¿Multitasking, paralelismo?

Procedural Programming



Características

El paradigma basado en procedimientos es imperativo

→ Da instrucciones a la máquina en forma ordenada, de arriba a abajo sin saltos

Características

Las estructuras de control nos permiten modificar el flujo del programa, pero siempre de forma controlada

Estructuras de Control de Flujo

1. Bifurcaciones condicionales (if/else, switch)
2. Ciclos (for, while, do/while)

C:

```
for(int k =0; k<100; k++) {  
    if (k%2) {  
        printf("%d\n",k+1);  
    }  
}
```

Python:

```
for k in range(100):  
    if (k%2):  
        print(k+1)
```

Características

Cuando se debe ejecutar una tarea discreta (en el sentido que no se le da el control permanente del programa) se llaman **procedimientos**


Características

Las variables son explícitamente asignadas y transformadas (operador =), por lo que se puede verificar su estado en todo momento

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



```
void print_line(char* text) {  
    printf("%s\n", text);  
}
```

El control se pasa
a un
procedimiento en
forma temporal



```
int main(void) {  
    printf("Ingresa tu nombre: ");  
    char name[50];  
    scanf("%s", name);  
    print_line(name);  
    return 0;  
}
```

Instrucciones son
ejecutadas desde
el inicio del
método main en
forma ordenada,
una a una



Forma procedural en Python

```
def print_line(text):  
    print(text);  
  
def main():  
    print("Ingresa tu nombre: ")  
    name = input()  
    print_line(name)  
  
if __name__ == '__main__':  
    main()
```

C vs Python

Discusión

Actividad 1

- Ir a <https://repl.it>
- Seleccionar el lenguaje C
- Copiar el código
- Modificarlo para que la solicitud de nombre sea una función

```
#include <stdio.h>

void print_line(char* text) {
    printf("%s\n", text);
}

int main(void) {
    printf("Ingresa tu nombre: ");
    char name[50];
    scanf("%s", name);
    print_line(name);
    return 0;
}
```

```
#include <stdio.h>

void print_line(char* text) {
    printf("%s\n", text);
}

void ask_name(char*
name) {
    printf("Ingresa tu nombre:
");
    scanf("%s", name);
}
```

```
int main(void) {
    char name[50];
    ask_name(name);
    print_line(name);
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
void print_line(char* text) {
    printf("%s\n", text);
}
char* ask_name() {
    char* name =
(char*)malloc(50*sizeof(char)
);
    printf("Ingresa tu nombre: ");
    scanf("%s", name);
    return name;
}
```

```
int main(void) {
    char* name = ask_name();
    print_line(name);
    return 0;
}
```

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

void print_line(char* text) {
    printf("%s\n", text);
}

void ask_name(char** name) {
    *name =
(char*)malloc(50*sizeof(char));
    printf("Ingresa tu nombre: ");
    scanf("%s", *name);
}
```

```
int main(void) {
    char* name;
    ask_name(&name);
    print_line(name);
    return 0;
}
```

```
ask_name(name);  
print_line(name);
```

¿A qué se van pareciendo los
códigos anteriores?

```
def print_line(text):  
    print(".".join(text))
```

```
def ask_name(name):  
    print("Ingresa tu nombre: ");  
    new_name = input()  
    name.extend(list(new_name))
```

```
if __name__ == '__main__':  
    name = []  
    ask_name(name)  
    print_line(name)
```



¿Por qué tengo
que hacer estos
trucos en Python?

Paso por referencia y valor

pass by reference



`fillCup()`

pass by value



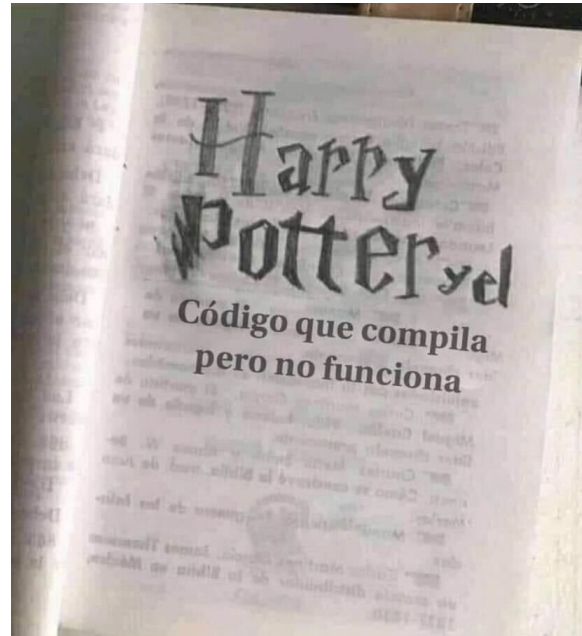
`fillCup()`

www.mathwarehouse.com

¿y Python?

Es una mezcla de ambos. Se pasa por referencia, pero cuando se cambia el valor en un método entonces se crea una copia

Volvamos al paradigma procedural



Beneficios

1. Modularidad: “simplificar” programas largos y complejos

Beneficios

2. Protección de variables: variables “viven” dentro de los procedimientos

Beneficios

3. Compartir el código: uso de bibliotecas/módulos

Beneficios

4. Fácil de leer: la sintaxis es básica

¿Qué vimos?

- Paradigma Procedural
- Ejemplos en C y Python

¿Preguntas?

**CUANDO ESCRIBÍ ESTE CÓDIGO,
SÓLO DIOS Y YO SABÍAMOS
CÓMO Y PARA QUÉ LO HICE**



AHORA, SÓLO DIOS LO SABE

Actividad

Un juego en Python

Una de las gracias de un lenguaje procedural es dividir código en métodos/funciones.

Lo que haremos ahora es aprovecharnos de eso para que cada uno trabaje en una parte de un juego.

Un juego de Python

En grupos:

En 15 minutos van a definir un juego simple en Python.

Van a crear un programa que llama a N funciones, donde N es el número de integrantes
Cada integrante hace una función POR SU CUENTA

Un juego de Python

Ejemplos:

Juego del gato:

- Crea tablero
- Jugada(símbolo jugador)
- Verifica casilla vacía
- Verifica ganador

¿Preguntas?