Programación de Computadores

Tema 12: Funciones

Carrera Ingeniería Civil en Informática
y Ciencias de la Computación
Universidad de Concepción

José Fuentes - jfuentess@inf.udec.cl

¿Por qué usar funciones?

- **Modularidad:** Permite separar nuestro código en pequeñas partes, mejorando la legibilidad
- A la hora de solucionar errores, sólo se edita la función que contiene el error
- Permite una mejor colaboración
- Nos permite reutilizar código

Componentes de una función

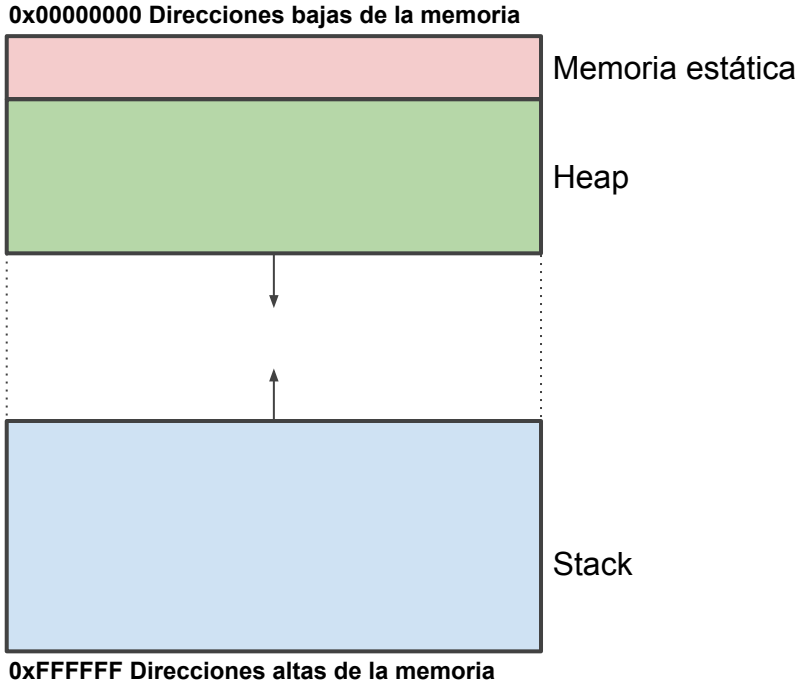
```
<tipo dato> nombre(var1, var2, … ) {
   … código …
   return <variable>;
}
```

```
double area_tri(double b, double a) {
   double area  = (b*a)/2;
   return area;
}
```

```
void print_array(int n, int A[n]) {
   if(n <= 0) return;

   for(int i=0; i < n; i++)
     printf("%d ", A[i]);
   return; // se puede omitir
}
```

Llamadas y contextos

```c
double area_tri(double b, double a) {
   double area  = (b*a)/2;
   return area;
}

int main() {
   double bs, al;
   printf("Ingrese base y altura: ");
   scanf("%lf %lf", &bs, &al);

   double ar = area_tri(bs, al);

   printf("Área resultante: %lf\n", ar);
   return 1;
}
```



Paso de parámetros

Paso por valor

```
void swap1(int v1, int v2) {
   int tmp  = v1;
   v1 = v2;
   v2 = tmp;
}
```

Paso por referencia

```
void swap2(int *v1, int *v2) {
   int tmp  = *v1;
   *v1 = *v2;
   *v2 = tmp;
}
```

```
void main() {
   int a=30, b=10;
   int c=30, d=10;
   swap1(a, b);
   swap2(&c, &d);
   printf("a:%d, b:%d\n", a, b);
   printf("c:%d, d:%d\n", c, d);
}
```

Paso de parámetros

Paso por valor

```
void swap1(int v1, int v2) {
   int tmp  = v1;
   v1 = v2;
   v2 = tmp;
}
```

Paso por referencia

```
void swap2(int *v1, int *v2) {
   int tmp  = *v1;
   *v1 = *v2;
   *v2 = tmp;
}
```

Paso por referencia (sin variable adicional)

```
void swap3(int *v1, int *v2) {
   *v1 = *v1 + *v2;
   *v2 = *v1 - *v2;
   *v1 = *v1 - *v2;
}
```

¡Un extra!

¡A practicar!

Ejemplo 1:
random_rotor.c

Ejemplo 2:
area_triangulo.c

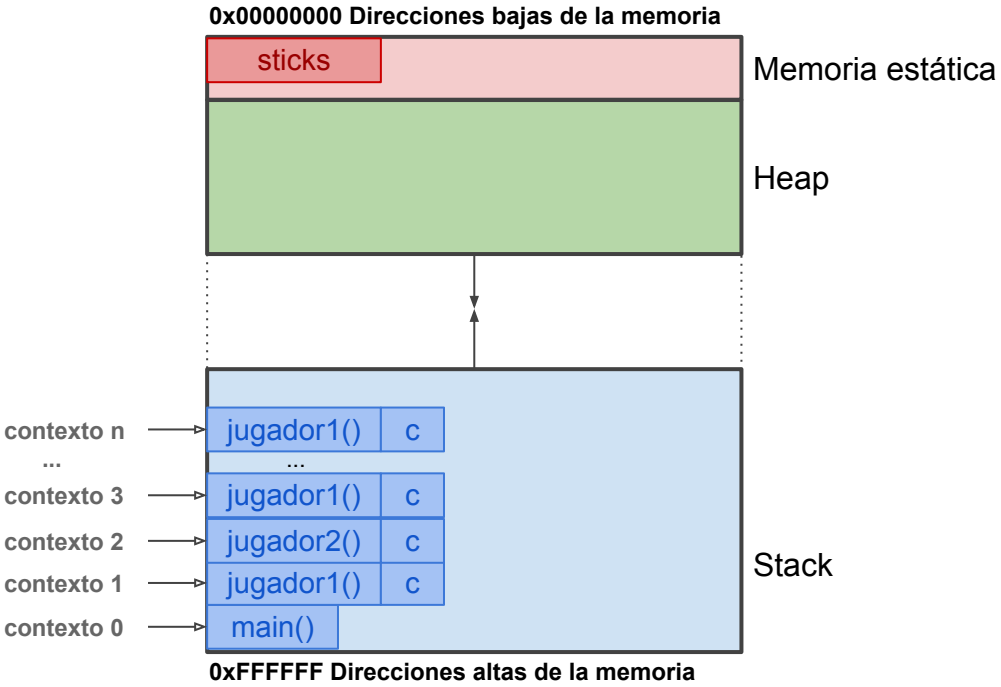
Ejemplo 3:
strlen_local.c

Ejemplo 4:
selection_sort.c

Ejemplo 5:
guardar_arreglo.c

Ejemplo 6:
Ejercicios 12-15 del listado de ensayo

Juego de la pila



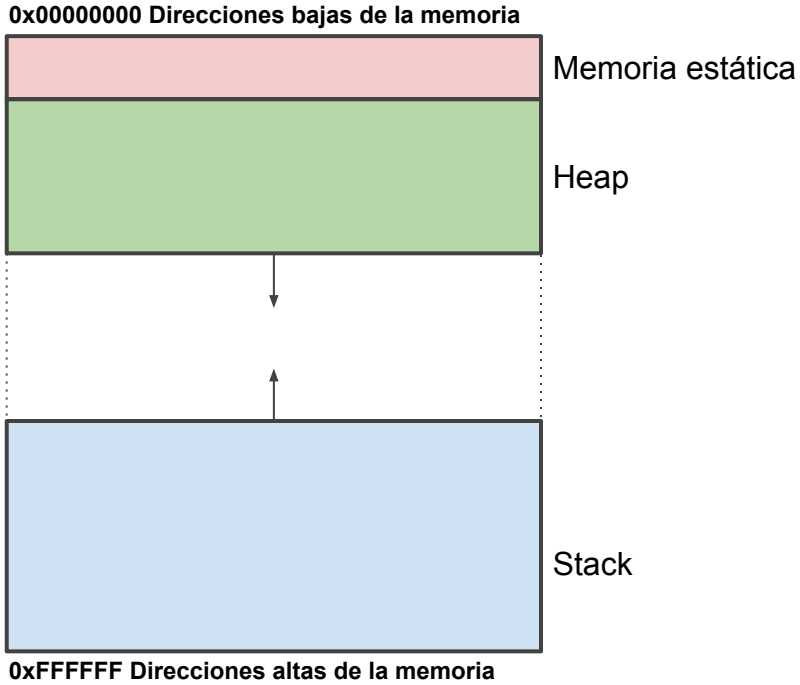
Ver: juego_pila.c

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

int sticks = 0;

void jugador1();
void jugador2();

int main (){
  sticks = 10 + rand() % 10;

  printf("¡Inicia el juego!\n");

  jugador1();

  return 0;
}

void jugador1() {
  int c;
  do {
    printf("\t[Jugador 1] Cuántos sticks sacarás (1, 2 o 3)? ");
    scanf("%d", &c);
    getchar();
  } while(c < 1 || c > 3);

  if((sticks - c) == 0) {
    printf("\t\t¡Felicitaciones jugador 1, eres el ganador!\n");
    return;
  }
  else if((sticks - c) < 0) {
    printf("\t\t¡Qué lástima jugador 1, has perdido!\n");
    return;
  }
  else
    sticks -= c;

  jugador2();
}

void jugador2() {
  int c;
  do {
    printf("\t[Jugador 2] Cuántos sticks sacarás (1, 2 o 3)? ");
    scanf("%d", &c);
    getchar();
  } while(c < 1 || c > 3);

  if((sticks - c) == 0) {
    printf("\t\t¡Felicitaciones jugador 2, eres el ganador!\n");
    return;
  }
  else if((sticks - c) < 0) {
    printf("\t\t¡Qué lástima jugador 2, has perdido!\n");
    return;
  }
  else
    sticks -= c;

  jugador1();
}
```

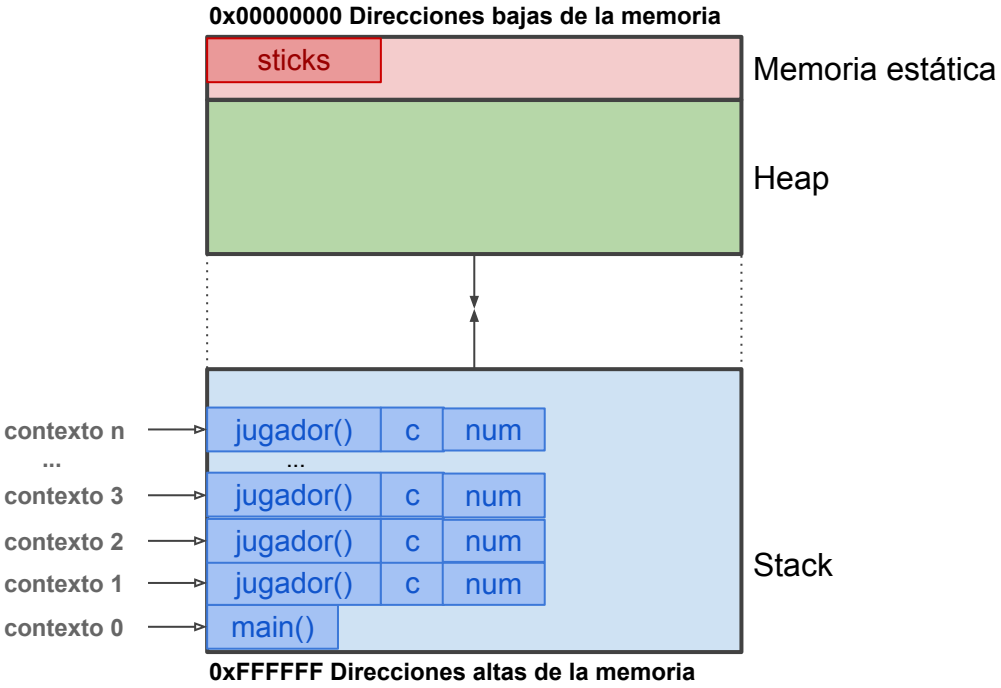
Función recursiva

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

int factorial(int n) {
   if(n == 1) return 1;
   return n * factorial(n-1);
}

int main() {
   int n;
   printf("Ingrese n: ");
   scanf("%d", &n);

   printf("El factorial de %d "
          "es %d\n", n, factorial(n));
   return EXIT_SUCCESS;
}
```

Ver: factorial.c



Juego de la pila (recursivo)



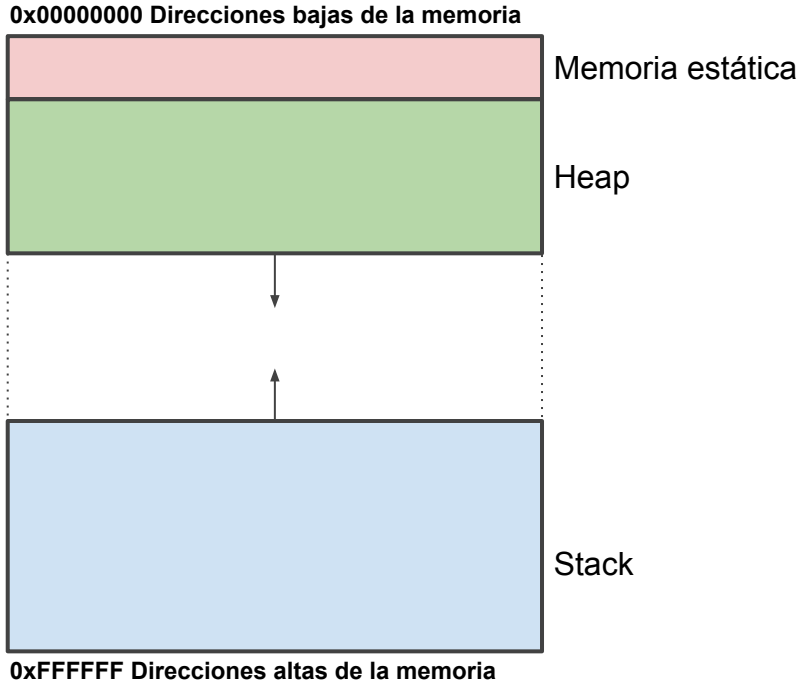
Ver: juego_pila_recursivo.c

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

int sticks = 0;

void jugador();

int main (){
  sticks = 10 + rand() % 10;

  printf("¡Inicia el juego!\n");

  jugador(1);

  return 0;
}

void jugador(int num) {
  int c;
  do {
    printf("\t[Jugador %d] Cuántos sticks sacarás (1, 2 o 3)? ", num);
    scanf("%d", &c);
    getchar();
  } while(c < 1 || c > 3);

  if((sticks - c) == 0) {
    printf("\t\t¡Felicitaciones jugador %d, eres el ganador!\n", num);
    return;
  }
  else if((sticks - c) < 0) {
    printf("\t\t¡Qué lástima jugador %d, has perdido!\n", num);
    return;
  }
  else
    sticks -= c;

  jugador(1 + (num%2));
}
```

Stackoverflow por recursión

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

unsigned int num_bytes = 0;
void funcionX(const int n, int i) {
   char arreglo[n];
   num_bytes +=n; i++;
   printf("Llamada %d: %d bytes", i, num_bytes);
   funcionX(n, i);
}

int main() {
   int n;
   printf("Ingrese cantidad de bytes: ");
   scanf("%d", &n);
   funcionX(n, 0);
   return EXIT_SUCCESS;
}
```

Ver: stackoverflow_recursivo.c



Retornando un arreglo

```
int * get_array(int n) {
   int *arreglo = malloc(n *
                sizeof(int));
   for(int i=0; i < n; i++)
      arreglo[i] = i;
   return arreglo;
}

int main() {
   int *A = get_array(10);
   for(int i=0; i < 10; i++)
      printf("%d ", A[i]);

   return EXIT_SUCCESS;
}
```

```
void get_array(int *arr, int n) {
   for(int i=0; i < n; i++)
      arr[i] = i;
}

int main() {
   int A[10];
   get_array(A, 10);
   for(int i=0; i < 10; i++)
      printf("%d ", A[i]);

   return EXIT_SUCCESS;
}
```

```
int * get_array() {
   static int arreglo[10];
   for(int i=0; i < 10; i++)
      arreglo[i] = i;
   return arreglo;
}

int main() {
   int *A = get_array();
   for(int i=0; i < 10; i++)
      printf("%d ", A[i]);

   return EXIT_SUCCESS;
}
```

```c
#include "rotores.h"

void rotar(char *rotor, int n) {
    char aux = rotor[n-1];
    for (int i = 0; i < n; i++)
        rotor[n - i] = rotor[n - i -1];
    rotor[0] = aux;
}

char *generar_rotor(int *n) {
  *n = 62;
  char *rotor = malloc(62 * sizeof(char));

  for(int i=0; i < 10; i++) rotor[i]    = 48+i;
  for(int i=0; i < 26; i++) rotor[10+i] = 65+i;
  for(int i=0; i < 26; i++) rotor[36+i] = 97+i;

  for(int i=0; i < 100; i++) {
    int a = rand() % 62;
    int b = rand() % 62;
    char tmp = rotor[a];
    rotor[a] = rotor[b];
    rotor[b] = tmp;
  }
  return rotor;
}
```

rotores.c

```c
#include <stdlib.h>

void rotar(char *, int);
char *generar_rotor(int *);
```

rotores.h

```c
#include <stdio.h>

#define BOLD "\e[1m"
#define WHITE "\e[7m"
#define BLUE "\e[44m"
#define NORMAL "\e[0m"

void imprimir_arreglo(char *, int);
void guardar_arreglo(char *, int, const char *)
char * cargar_arreglo(const char *, int *);
```

utils_array.h

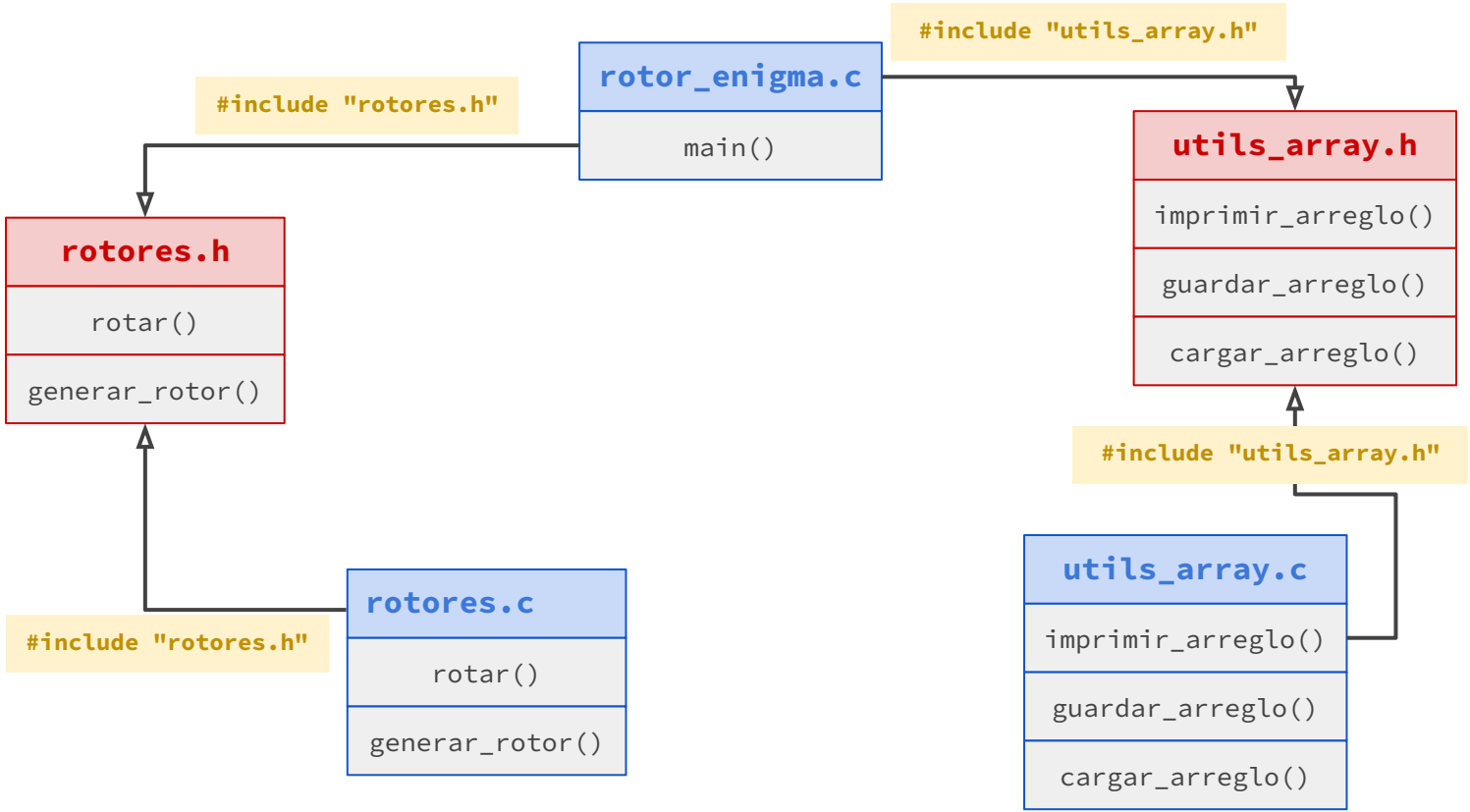
```c
#include "utils_array.h"
#include "rotores.h"

int main() {
  int n = 0;
  char *rotor = generar_rotor(&n);
  imprimir_arreglo(rotor, n);

  for(int i=0; i < 3; i++) {
    rotar(rotor, n);
    imprimir_arreglo(rotor, n);
  }

  guardar_arreglo(rotor, n, "rotor1.txt");
  int nn=0;
  char *rotor2 = cargar_arreglo("rotor1.txt", &nn);
  rotar(rotor2, nn);
  imprimir_arreglo(rotor2, nn);

  return EXIT_SUCCESS;
}
```

rotor_enigma.c

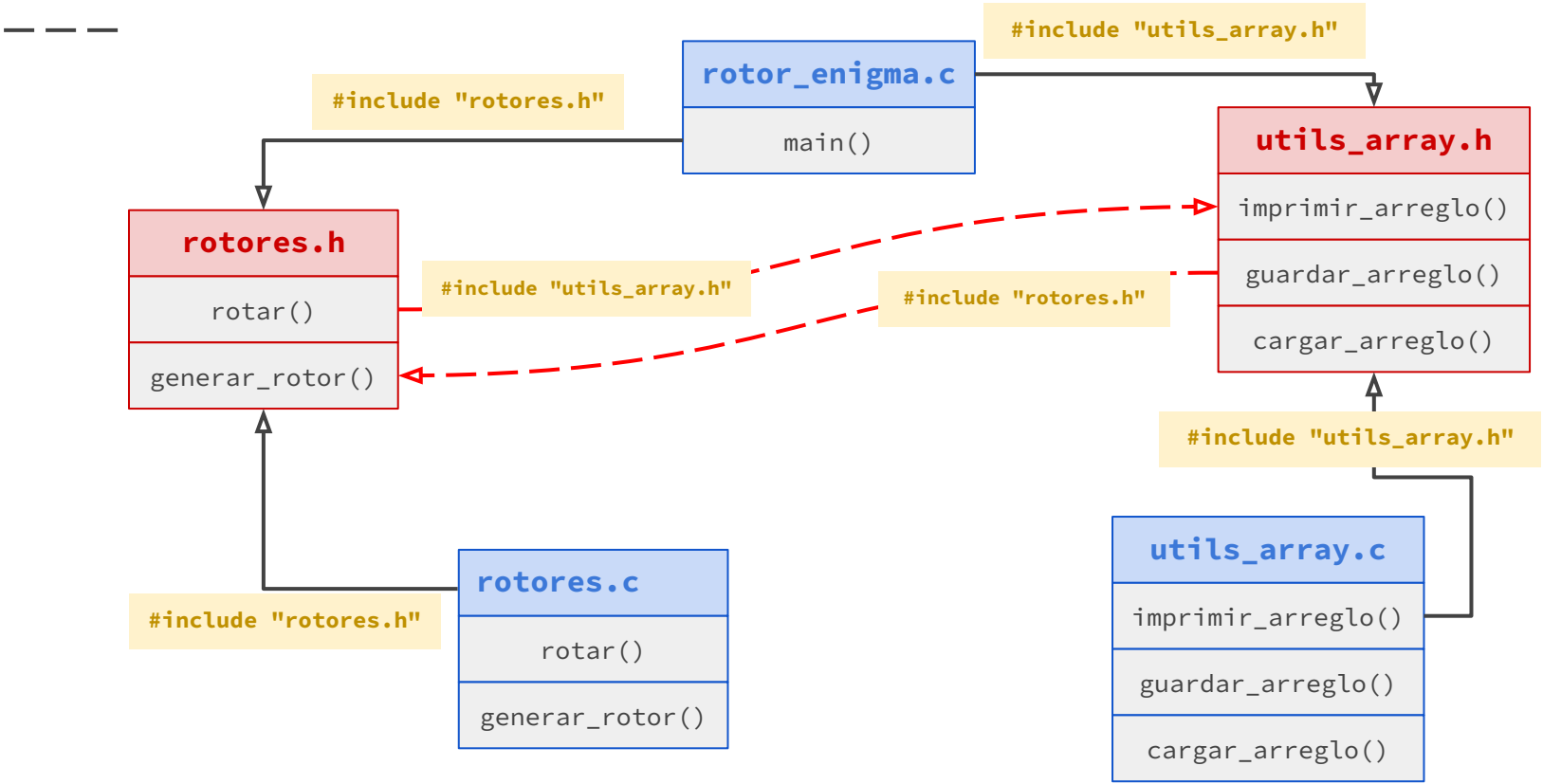
```c
#include <stdlib.h>
#include "utils_array.h"

void imprimir_arreglo(char *A, int n) {
  for(int i=0; i < n; i++)
    printf("%2d ", i);
  printf("\n");

  for(int i=0; i < n; i++)
    printf("%s%s%2c %s", BOLD, BLUE, A[i], NORMAL);
  printf("\n\n");
}

void guardar_arreglo(char *A, int n, const char *fname) {
  FILE *fout = fopen(fname, "w");

  if (!fout) {
    fprintf(stderr, "Error al abrir el archivo \"%s\".\n", fname);
    exit(EXIT_FAILURE);
  }

  fwrite(A, sizeof(char), n, fout);
  fclose(fout);
}

char* cargar_arreglo(const char *fname, int *n) {
  FILE *fin = fopen(fname, "r");

  if (!fin) {
    fprintf(stderr, "Error al abrir el archivo \"%s\".\n", fname);
    exit(EXIT_FAILURE);
  }

  fseek(fin, 0L, SEEK_END);
  *n = ftell(fin);
  fseek(fin, 0L, SEEK_SET);

  char *A = malloc(*n * sizeof(char));
  fread(A, sizeof(char), *n, fin);
  fclose(fin);
  return A;
}
```

utils_array.c

Dependencias entre archivos



Dependencias entre archivos: dependencia circular



Makefile

Contenido del archivo **Makefile**

```
CC = gcc
CFLAGS = -Wall -Wextra

rotor: rotor_enigma.o rotores.o utils_array.o
	$(CC) $(CFLAGS) -o rotor rotor_enigma.c rotores.c utils_array.c

rotor_enigma.o: rotor_enigma.c
	$(CC) $(CFLAGS) -c rotor_enigma.c

rotores.o: rotores.c rotores.h
	$(CC) $(CFLAGS) -c rotores.c

utils_array.o: utils_array.c utils_array.h
	$(CC) $(CFLAGS) -c utils_array.c
```

En la **consola**

```
make rotor
```

¡A practicar!

Ejemplo 1:
juego_pila.c

Ejemplo 2:
juego_pila_recursivo.c

Ejemplo 3:
stackoverflow_recursivo.c

Ejemplo 4:
largo_frase.c

Ejemplo 5:
invertir_frase.c

Ejemplo 6:
rotor_enigma.c/rotores.c/
utils_array.c