Asignación Dinámica de Memoria

~~Memoria Estática~~ → Espacio q' se crea al declarar variables de cualquier tipo de dato (int...);
Esta memoria en sus variables no se puede cambiar de ubicación en la ejecución y tampoco puede ser liberada manualmente.

Memoria Dinámica → Se reserva en tiempo de ejecución, su tamaño varía durante la ejecución del programa. Se usa cuando a priori no conocemos el Nº de datos/elementos a tratar:

F. malloc → Solicita una parte de la memoria del tamaño solicitado como parámetro. Devuelve un puntero a la zona de memoria concedida (esto sino está ocupada); si no puede conceder el bloque devuelve " NULL.

```
TIPO *PUNTERO;
PUNTERO = MALLOC (Tamaño en bytes);

INT *P1;
P1 = MALLOC (SIZEOF (INT));

CHAR *C1;
C1 = MALLOC (SIZEOF (CHAR));
```

Uso del malloc en C

```
INT *a;
a = MALLOC (SIZEOF (INT)); // SIZEOF (INT) Tamaño de una variable.
*a = 5;

PRINTF ("%I", *a);
RETURN 0;
```

7.1) Utilizar malloc para reservar memoria para un nombre (string)

```
CHAR NOMBRE[20], *P_NOMBRE;
INT LONGITUD;

STRCPY (NOMBRE, "ALEJANDRO"); //ALEJANDRO \0
LONGITUD = STRLEN (NOMBRE);

P_NOMBRE = MALLOC ((LONGITUD+1) * SIZEOF (CHAR));
STRCPY (P_NOMBRE, NOMBRE);

PRINTF ("NOMBRE: %S", P_NOMBRE);
RETURN 0;
```

72) RESERVAR MEMORIA CON MALLOC PARA UN ARRAY DE TIPO DOUBLE PARA 1000 Nº Y COMPROBAR SI EL PUNTERO ES VALIDO (SI EL PUNTERO DEVUELVE NULL ; NO HAY SUFICIENTE MEMORIA).

```
#DEFINE TOPE 1000

DOUBLE * P_ARRAY;
INT I;
P_ARRAY = MALLOC (TOPE * SIZEOF (DOUBLE));

IF (P_ARRAY == NULL ){
    PRINTF ("ERROR EN LA ASIGNACIÓN DE MEMORIA");
    RETURN -1; // INTENTAR RECUPERAR MEMORIA
}
ELSE{
    SRAND (TIME (NULL));
    FOR (I=0; I<TOPE; I++){
        P_ARRAY [I] = 1 + RAND () % ((TOPE+1)-1);
        PRINTF("%IF.\n", P_ARRAY [I]);
    }
}
```

FUNCION REALLOC

PERMITE AMPLIAR UN BLOQUE DE MEMORIA RESERVADA ANTERIORMENTE. SE LA LLAMA ASI:

*PUNTERO = REALLOC (PUNTERO A BLOQUE, TAMAÑO TOTAL DE NUEVO BLOQUE);

73) CREAR UN ARREGLO DE 3 ELEMENTOS, REALLOCARLO Y TRANSFORMARLO EN UN ARREGLO DE 5 ELEMENTOS

```
INT *VECTOR, *VECTOR_CONVERTIDO;
INT I;
VECTOR = MALLOC(3 * SIZEOF(INT)); // RESERVANDO MEMORIA PARA 3 ELEMENTOS
//PRIMER VECTOR
VECTOR[0] = 1;
VECTOR[1] = 2;
VECTOR[2] = 3;

FOR(I=0; I<3; I++){
    PRINTF("%I", VECTOR[I]);
}

//REALLOC()
VECTOR_CONVERTIDO = REALLOC(VECTOR, 5 * SIZEOF(INT)); //AMPLIANDO EL ARREGLO A 5
//TERMINANDO DE AGREGAR VALORES
VECTOR[3] = 4;
VECTOR[4] = 5;

FOR(I=0; I<5; I++){
    PRINTF("%I", VECTOR_CONVERTIDO[I]);
}
```

74) RESERVAR MEMORIA DINAMICA PARA UNA CADENA DE CARACTERES DE 10 ESPACIOS, LUEGO UTILIZAR REALLOC PARA AMPLIAR LA MEMORIA DINAMICA A 30 ESPACIOS.

```
CHAR *P_NOMBRE, *P_NOMBRECOMPLETO;
P_NOMBRE = MALLOC(10*SIZEOF(CHAR));
STRCPY(P_NOMBRE, "ALEJANDRO");
PRINTF("NOMBRE: %S", P_NOMBRE);
//AMPLIANDO LA CADENA DE CARACTERES A 30 ESPACIOS
P_NOMBRECOMPLETO = REALLOC(P_NOMBRE, 30*SIZEOF(CHAR));
STRCAT(P_NOMBRECOMPLETO, " MIGUEL TABOADA");
PRINTF("\n NOMBRE COMPLETO: %S", P_NOMBRECOMPLETO);

RETURN 0;
}
```

75) Función calloc

```
PUNTERO = CALLOC (NÚMERO DE ELEMENTOS, TAMAÑO DE CADA ELEMENTO);
INT *PI;
INT I;
PI = CALLOC (5, SIZEOF (INT));
FOR (I=0; I<5; I++){
   PRINTF ("DIGITE UN NÚMERO: ");
   SCANF ("%I", &PI[I]);
}
PRINTF ("\n\n");
FOR (I=0; I<5; I++){
   PRINTF ("%I", PI[I]);
}
RETURN 0;
```

76) Reservar memoria con calloc para una cadena de caracteres.

```
CHAR *C, PALABRA[50];

PRINTF("DIGITE UNA PALABRA (STRING): ");
GETS (PALABRA);
//PALABRA\0
C = CALLOC (STRLEN (PALABRA)+1, SIZEOF (CHAR));
STRCPY (C, PALABRA);
PRINTF ("PALABRA: %S", C);

FREE (C);
RETURN 0;
```