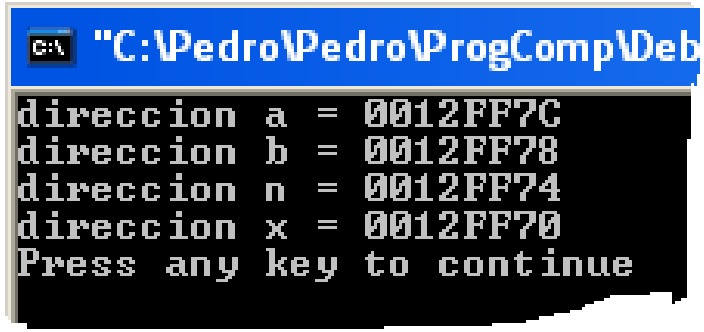
Punteros

Punteros – ejemplo operador &

```
/*****************************************************************\
* Programa: pr_oper&.c                                            *
* Descripción: Prog. que imprime direcciones de memoria con &     *
* Autor: Pedro Corcuera                                           *
* Revisión: 1.0 2/02/2008                                         *
\*****************************************************************/
#include <stdio.h>

int main()
{
  char a;
  char b;
  int  n;
  float x;

  printf("direccion a = %p\n", &a);
  printf("direccion b = %p\n", &b);
  printf("direccion n = %p\n", &n);
  printf("direccion x = %p\n", &x);

  return 0;
}
```



Argumentos de `main`

- La función principal **`main`** puede recibir argumentos que permiten acceder a los parámetros con los que es llamado el ejecutable.
- Dentro del paréntesis de main se especifican dos argumentos:
  - `int argc` : Número de parámetros.
  - `char* argv[]` : Parámetros del ejecutable.

  con lo que la definición de main queda así:

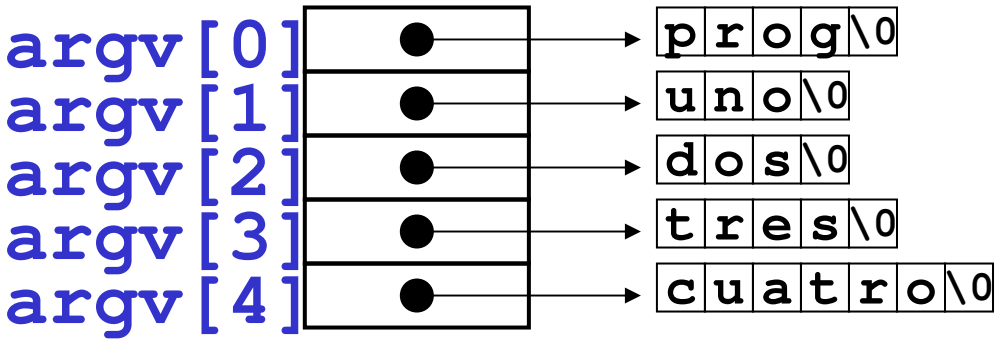
```
int main(int argc, char* argv[])
```

Argumentos de `main`

```
> cl prog.c -o prog.exe
$ prog uno dos tres cuatro
```

```
int main(int argc, char* argv[])
```

`argc=5` (incluye el comando)



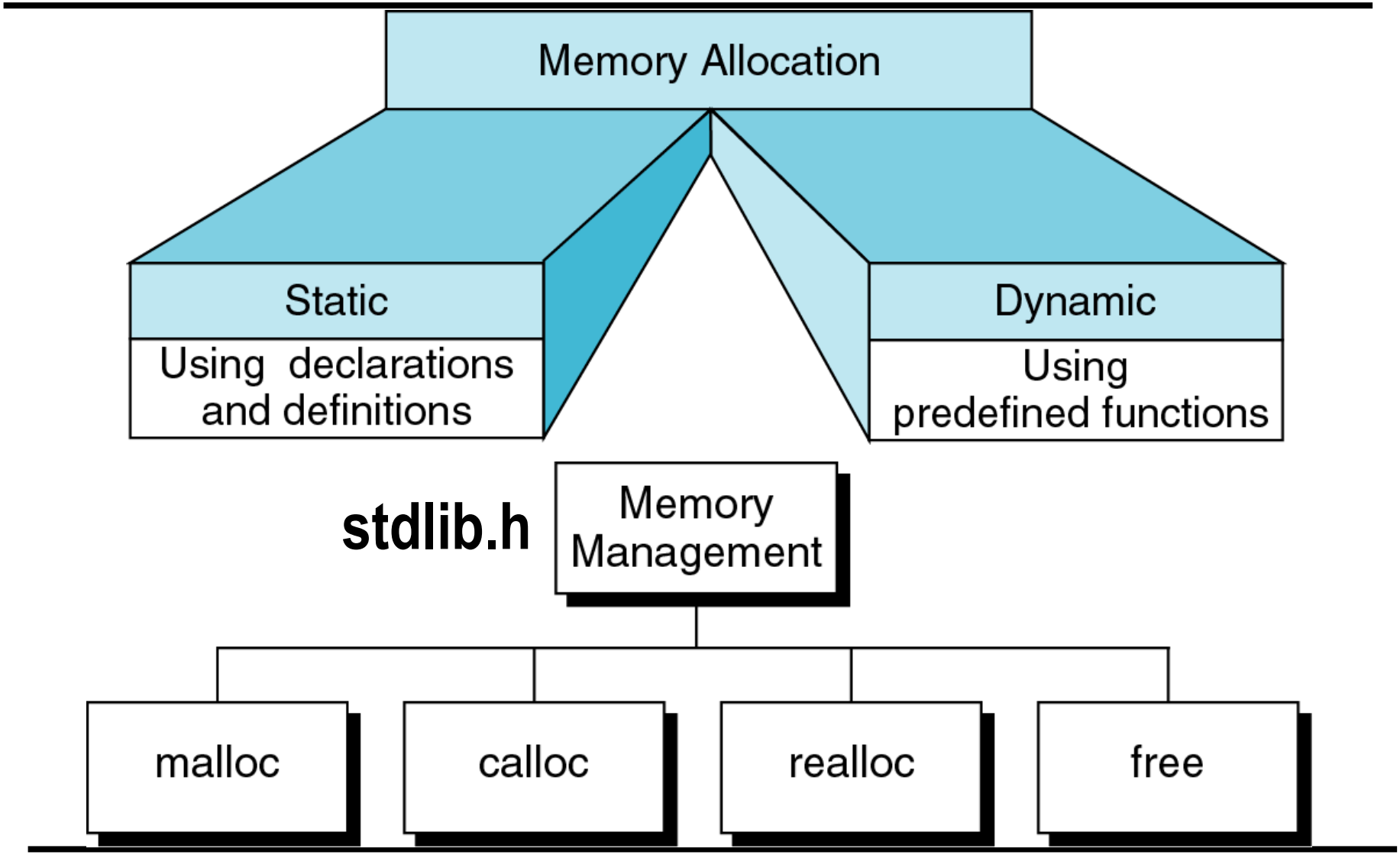
Ejemplo: imprime argumentos de main

```
/*****************************************************************\
* Programa: arg_main.c                                            *
* Descripción: Prog. que imprime el comando y los argumentos de main     *
* Uso: >progcomp arg1 arg2 arg3                                   *
* Autor: Pedro Corcuera                                           *
* Revisión: 1.0 2/02/2008                                         *
\*****************************************************************/
#include <stdio.h>

int main(int argc, char *argv[])
{
   int i=0;
   printf("Ejecutable: %s \tNumero de argumentos: %d\n",argv[0], argc);

   for(i = 1; i < argc;i++)
        printf("Argumento[%d]: %s\n",i,argv[i]);

   return 0;
}
```

Asignación dinámica de memoria

- Además de la reserva de espacio estática (cuando se declara una variable), es posible reservar memoria de forma dinámica.
- Hay funciones de gestión de memoria dinámica (stdlib.h):
  - **void *malloc(size_t)**: Reserva memoria dinámica.
  - **void *calloc(size_t)**: Reserva memoria dinámica.
  - **void *realloc(void *,size_t)**: Ajusta el espacio de memoria dinámica.
  - **free(void *)**: Libera memoria dinámica.

Memoria dinámica



Ejemplo: array dinámico

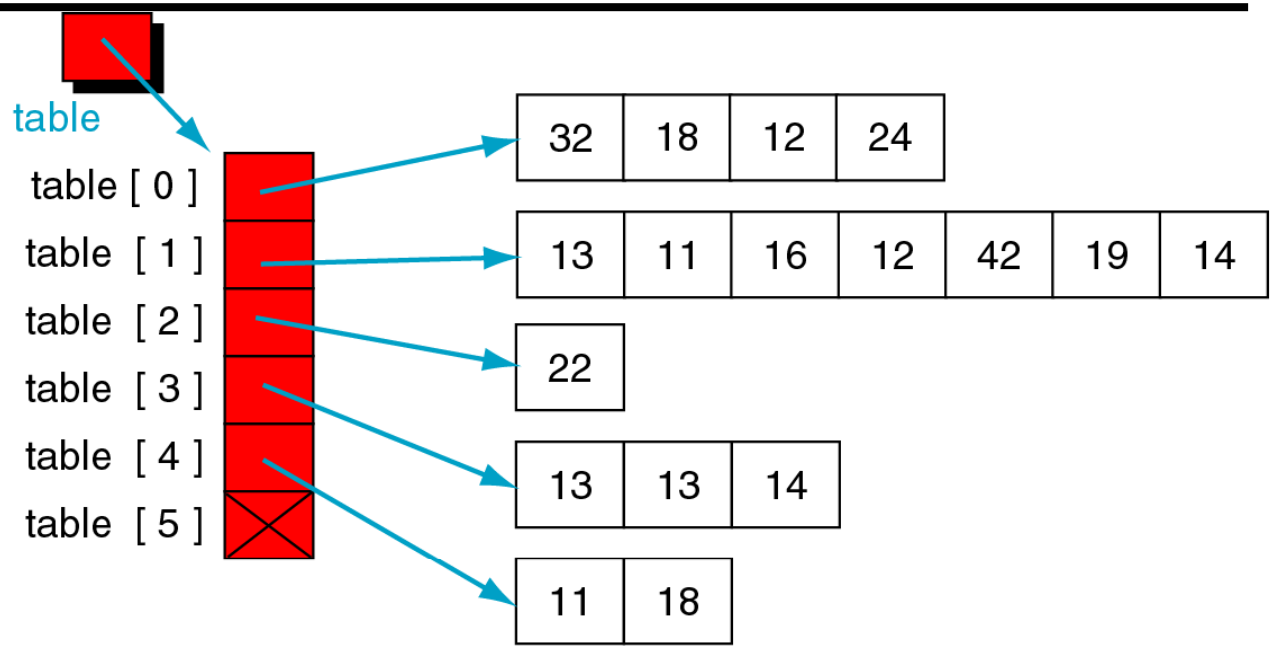
```
/*****************************************************************\
* Programa: array_dinamico.c                                      *
* Descripción: Prog. que crea una array dinamico y genera otro con los   *
*              elementos pares del primer array y lo imprime      *
\*****************************************************************/
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int dim_usu;    /* dimension del vector del usuario */
    int dim_par;    /* dimensión del vector de elementos pares */
    int n;          /* indice para los for */
    int m;          /* indice para recorrer arrya de pares */
    int *pvec_usu; /* puntero al vector introducido por el usuario */
    int *pvec_par; /* puntero al vector elementos pares (dinamico) */

    printf("Introduzca la dimension del vector: ");
    scanf(" %d", &dim_usu);

    pvec_usu = (int *) calloc( dim_usu, sizeof(int)); /*Asignar memoria vect. usuario
    */
    /*pvec_usu = (int *) malloc( dim_usu*sizeof(int));*/
    if (pvec_usu == NULL) {  /* si no hay memoria */
          printf("Error: no hay memoria para un vector de %d elementos\n", dim_usu);
    }
```

Ejemplo: array dinámico

```
    else
    {
          for (n = 0; n < dim_usu; n++) /* pedir elementos del vector */
          { printf("Elemento %d = ", n); scanf("%d", &(pvec_usu[n]));}
          dim_par = 0;
          for (n = 0; n < dim_usu; n++)
             if ((pvec_usu[n] % 2) == 0) dim_par++;
          pvec_par = (int *) calloc( dim_par, sizeof(int));
          if (pvec_par == NULL) {  /* si no hay memoria */
             printf("Error: no hay memoria para %d elementos\n", dim_par);
          }
          else
          { /* se copian los elementos pares */
                    m = 0;
                    for (n = 0; n < dim_usu ; n++)
                        if ((pvec_usu[n] % 2) == 0) { pvec_par[m] = pvec_usu[n]; m++;}
                    printf("\n----------------------------\n");
                    for (n = 0; n < dim_par ; n++)
                              printf("Elemento par %d = %d \n", n, pvec_par[n]);
          }
          free (pvec_par);
    }
    free (pvec_usu);
}
```

Matriz con número de columnas diferentes

- Se representa como un array de arrays (puntero a puntero) dinámico.




```
table = (int **)calloc (rowNum + 1, sizeof(int*));

table[0] = (int*)calloc (4, sizeof(int));
table[1] = (int*)calloc (7, sizeof(int));
table[2] = (int*)calloc (1, sizeof(int));
table[3] = (int*)calloc (3, sizeof(int));
table[4] = (int*)calloc (2, sizeof(int));
table[5] = NULL;
```