



Estructuras

Heroe	
Nombre	Kratos
Vida	1000
Defensa	5000
Ataque	100
% de efectividad	60.5

```
struct Heroe{  
    char nombre[10];  
    int vida, defensa, ataque;  
    float efectividad;  
};
```

```
struct Heroe h1;
```



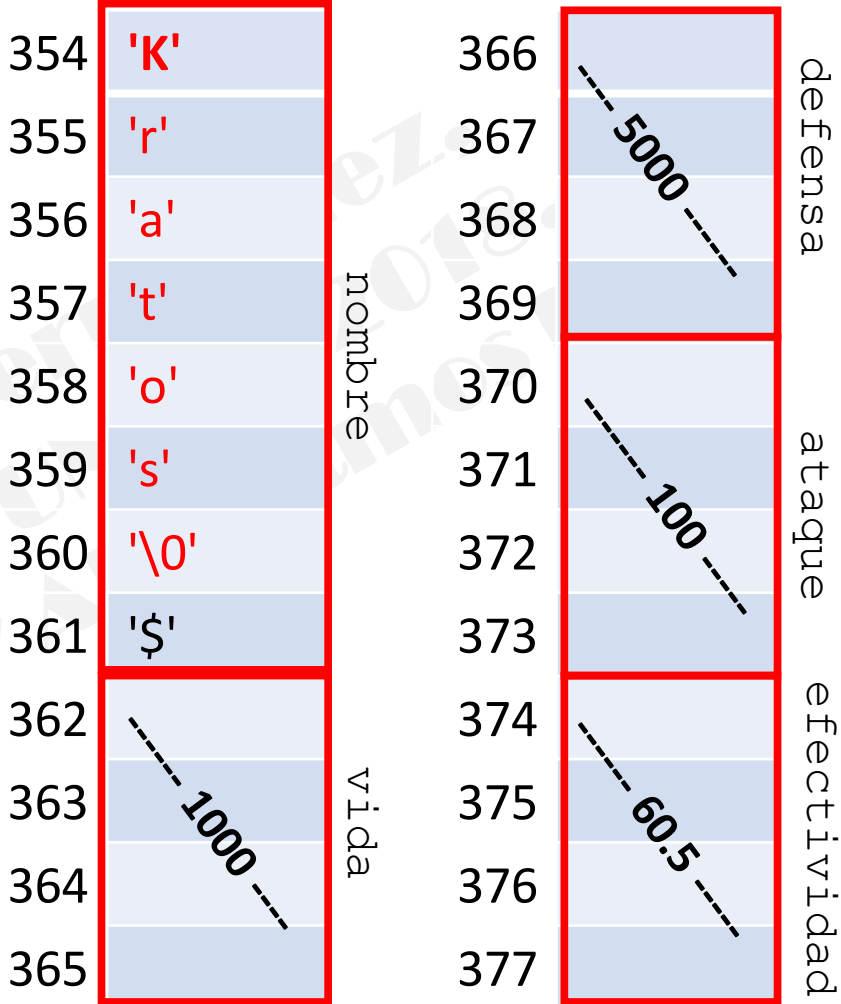
```
struct Heroe{
    char nombre[8];
    int vida,defensa,ataque;
    float efectividad;
};
```

```
int main() {
    struct Heroe h1;
    strcpy(h1.nombre,"kratos");
    h1.vida = 1000;
    h1.defensa = 5000;
    h1.ataque = 100;
    h1.efectividad = 60.5;
}
```

Tabla de símbolos main()

Símbolo	Tipo	Dirección
h1	struct Heroe	354

Heroe	
nombre	"Kratos"
vida	1000
defensa	5000
ataque	100
efectividad	60.5



```

struct Heroe{
    char nombre[8];
    int vida, defensa,
    ataque;
    float efectividad;
};

int main() {
    struct Heroe h1, h2;
    h1.vida = 1000;
    h1.defensa = 5000;
    h1.ataque = 100;
    h1.efectividad = 60.5;
    h2 = h1;
}

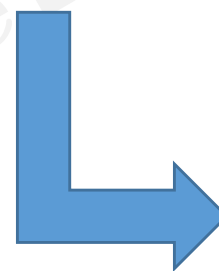
```

Tabla de símbolos main()

Símbolo	Tipo	Dirección
h1	struct Heroe	354
h2	struct Heroe	720

h1

Heroe	
nombre	"Kratos"
vida	1000
defensa	5000
ataque	100
efectividad	60.5



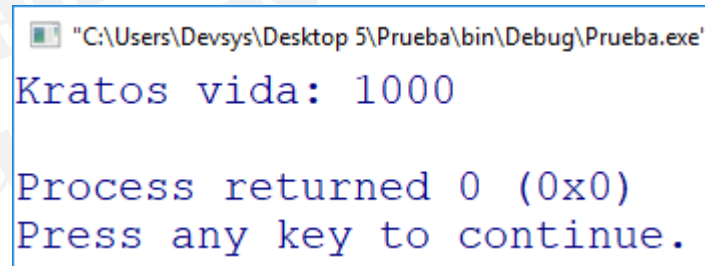
h2

Heroe	
nombre	"Kratos"
vida	1000
defensa	5000
ataque	100
efectividad	60.5

Inicialización de estructuras durante la definición de la variable

```
struct Heroe{  
    char nombre[8];  
    int vida, defensa, ataque;  
    float efectividad;  
};
```

```
int main() {  
    struct Heroe h = {"Kratos", 1000, 5000, 100, 60.5};  
    printf("%s vida: %d\n", h.nombre, h.vida);  
}
```



"C:\Users\Devsys\Desktop 5\Prueba\bin\Debug\Prueba.exe"
Kratos vida: 1000
Process returned 0 (0x0)
Press any key to continue.

Funciones que reciben estructuras (parámetros por valor)

```
struct Heroe{
    char nombre[8];
    int vida;
    int defensa;
    int ataque;
    float efectividad;
};

void imprime(struct Heroe h) {
    printf("Nombre: %s\n", h.nombre);
    printf("Vida: %d\n", h.vida);
}

int main() {
    struct Heroe hk = {"Kratos", 1000, 5000, 100, 60.5};
    imprime(hk);
    return 0;
}
```

Funciones que regresan estructuras

```
struct Heroe{  
    char nombre[8];  
    int vida;  
    int defensa;  
    int ataque;  
    float efectividad;  
};
```

```
struct Heroe aumentaVida(struct Heroe h, int inc) {  
    h.vida = h.vida+inc;  
    return h;  
}
```

```
int main() {  
    struct Heroe hk = {"Kratos", 1000, 5000, 100, 60.5};  
    hk = aumentaVida(hk, 10);  
    imprime(hk);  
    return 0;  
}
```

**Funciones que reciben estructuras
(parámetros por referencia)**



Apuntadores a estructuras

Creación dinámica de estructuras

```
struct Heroe{  
    char nombre[8];  
    int vida, defensa, ataque;  
    float efectividad;  
};  
  
int main() {  
  
    struct Heroe *h;  
    h = malloc(sizeof(struct Heroe)) ;  
}
```

Operador	Descripción	Asociatividad
++ -- () [] . ->	Post- incremento y decremento Llamada a función Elemento de vector Selección de elemento por referencia Selección de elemento con puntero	Izquierda a derecha
++ -- + - ! ~ (type) * & sizeof	Pre- incremento y decremento Suma y resta unitaria NOT lógico y NOT binario Conversión de tipo Indirección Dirección de Tamaño de	Derecha a izquierda

```
*h.vida = 100;
*h.defensa = 5000;
strcpy(*h.nombre, "Kratos");
```

Error!!!

```
(*h).vida = 100;
(*h).defensa = 5000;
strcpy((*h).nombre, "Kratos");
```

Correcto

Correcto

```
(*h) . vida = 100;  
(*h) . defensa = 5000;  
strcpy ( (*h) . nombre, "Kratos" );
```



Correcto

```
h->vida = 100;  
h->defensa = 5000;  
strcpy ( h->nombre, "Kratos" );
```

Funciones que regresan estructuras

```
struct Heroe{
    char nombre[8];
    int vida;
    int defensa;
    int ataque;
    float efectividad;
};

void aumentaVida(struct Heroe *h, int inc) {
    h->vida = h->vida+inc;
}

int main() {
    struct Heroe hk = {"Kratos",1000,5000,100,60.5};
    aumentaVida(&hk,10);
    imprime(hk);
    return 0;
}
```

Arreglos ESTÁTICOS de estructuras

```
struct Heroe{  
    char nombre[8];  
    int vida, defensa, ataque;  
    float efectividad;  
};
```

```
#define n 3
```

Define un símbolo

```
int main() {
```

```
    int k;
```

```
    struct Heroe h[n] = { {"Kratos", 1000, 5000, 100, 60.5},  
                          {"Hades", 100, 3000, 200, 70.5},  
                          {"Zeus", 500, 2000, 250, 80.5}  
    };
```

```
    for (k=0; k<n; k++) {
```

```
        printf("%-7s vida: %d\n", h[k].nombre, h[k].vida);
```

```
    }
```

```
}
```

"C:\Users\Devsys\Desktop 5\Prueba\bin\Debug\Prueba.exe"

Kratos vida: 1000

Hades vida: 100

Zeus vida: 500

Process returned 0 (0x0)

Press any key to continue.

opcional

Arreglos de estructuras

Correcto

```
(*h).vida = 100;  
(*h).defensa = 5000;  
strcpy((*h).nombre, "Kratos");
```



Correcto

```
h[0].vida = 100;  
h[0].defensa = 5000;  
strcpy(h[0].nombre, "Kratos");
```

```
(*h).vida = 100;  
(*h).defensa = 5000;  
strcpy ((*h).nombre, "Kratos");
```

Correcto



```
h[0].vida = 100;  
h[0].defensa = 5000;  
strcpy(h[0].nombre, "Kratos");
```

Correcto

```
h->vida = 100;  
h->defensa = 5000;  
strcpy(h->nombre, "Kratos");
```

Correcto

Arreglos Dinámicos de estructuras

```
struct Heroe{  
    char nombre[8];  
    int vida;  
    int defensa;  
    int ataque;  
    float efectividad;  
};
```

```
int main() {  
    int k,n;  
    struct Heroe *h;  
    printf("número de heroes? ");  
    scanf(&n);  
    h = malloc(n*sizeof(struct Heroe));  
    strcpy(h[0].nombre,"Kratos");  
    h[0].vida = 1000;  
    h[0].defensa = 550;  
    /* y el código continua...*/  
}
```

Paso de arreglos **estáticos** de estructuras a funciones

```
int main() {  
    struct Heroe misHeroes[100];  
    int n;  
    leerHeroes(misHeroes, &n);  
    listarHeroes(misHeroes, n);  
    return 0;  
}
```

```
void listarHeroes(struct Heroe h[], int n) {
    int k;
    for (k=0;k<n;k++) {
        printf("%s      vida: %d\n",h[k].nombre,h[k].vida);
    }
}

void leerHeroes(struct Heroe h[], int *n) {
    int k;
    printf("Numero de héroes? ");
    scanf("%d",n);
    for (k=0;k<*n;k++) {
        fflush(stdin);
        gets(h[k].nombre);
        scanf("%d", &h[k].vida);
    }
}
```

Paso de arreglos **dinámicos** de estructuras a funciones

```
int main() {  
    struct Heroe *misHeroes;  
    int n;  
  
    printf("Numero de héroes? ");  
    scanf("%d", n);  
    misHeroes = malloc(n*sizeof(struct Heroe)) ;  
    leerHeroes(misHeroes, n);  
    listarHeroes(misHeroes, n);  
    return 0;  
}
```

```
void listarHeroes(struct Heroe h[], int n) {
    int k;
    for (k=0;k<n;k++) {
        printf("%s      vida: %d\n",h[k].nombre,h[k].vida);
    }
}

void leerHeroes(struct Heroe h[], int n) {
    int k;
    for (k=0;k<n;k++) {
        fflush(stdin);
        gets(h[k].nombre);
        scanf("%d", &h[k].vida);
    }
}
```

Recibiendo una arreglo como un apuntador y manejándola como un arreglo

```
void listarHeroes(struct Heroe *h, int n) {  
    int k;  
    for (k=0;k<n;k++) {  
        printf("%s      vida: %d\n",h[k].nombre,h[k].vida);  
    }  
}
```

```
void leerHeroes(struct Heroe *h, int n) {  
    int k;  
    for (k=0;k<n;k++) {  
        fflush(stdin) ;  
        gets(h[k].nombre) ;  
        scanf("%d",&h[k].vida) ;  
    }  
}
```

Recibiendo una arreglo como un apuntador y manejándola como un apuntador

```
void listarHeroes(struct Heroe *h, int n) {  
    int k;  
    for (k=0;k<n;k++) {  
        printf("%s vida: %d\n", (h+k)->nombre, (h+k)->vida);  
    }  
}
```

```
void leerHeroes(struct Heroe *h, int n) {  
    int k;  
    for (k=0;k<n;k++) {  
        fflush(stdin);  
        gets(h->nombre);  
        scanf("%d", &h->vida);  
        h = h+1;  
    }  
}
```

h es una variable local, por lo que deja de existir en cuanto termina la función