```c
#include <stdio.h>
#define SIZE(arr) sizeof(arr)/sizeof(*arr);
void fun(int *arr, int n)
{
int i;
*arr+=*(arr+n-1)+=10;
}
void printArr(int *arr, int n)
{
int i;
for(i=0; i<n;++i)
printf("\t%d",arr[i]);
}
int main()
{
int arr[] ={10,20,30};
int size =SIZE(arr);
fun(arr,size);
printArr(arr,size);
return 0;
}
```

Giải thích từng phần

1. **Macro** `#define SIZE(arr) sizeof(arr)/sizeof(*arr);`
   - Macro này được sử dụng để tính kích thước của mảng `arr`.
   - `sizeof(arr)` cho kích thước toàn bộ mảng (trong trường hợp này, `arr` là `int[]` có 3 phần tử, nên kích thước của nó là `3 * sizeof(int)`).
   - `sizeof(*arr)` cho kích thước của phần tử đầu tiên trong mảng (`int` có kích thước `sizeof(int)`).
   - Vì vậy, `SIZE(arr)` trả về `3`, là số phần tử trong mảng `arr`.

2. **Hàm** `fun(int *arr, int n)`
   - Hàm này nhận vào một mảng `arr` và kích thước `n` của nó.
   - Biểu thức chính trong hàm là:

     c Sao chép mã

     ```c
     *arr += *(arr + n - 1) += 10;
     ```

   - Cụ thể:
     - `arr` là con trỏ trỏ đến phần tử đầu tiên của mảng (`arr[0]`).
     - `*(arr + n - 1)` là giá trị của phần tử cuối cùng trong mảng (`arr[n-1]`), tức là `arr[2]` .
     - Biểu thức `*(arr + n - 1) += 10` tăng giá trị của phần tử cuối (`arr[2]`) thêm `10` .
     - Sau đó, giá trị của phần tử đầu tiên (`*arr` hoặc `arr[0]`) cũng được tăng thêm giá trị của phần tử cuối đã được tăng.
   - Sau khi thực hiện câu lệnh này:
     - `arr[0]` = `arr[0]` + (`arr[2]` + `10`)
     - `arr[2]` = `arr[2]` + `10`

3. **Hàm** `printArr(int *arr, int n)`
   - Hàm này dùng để in các phần tử của mảng `arr` sau khi hàm `fun` đã thay đổi giá trị của nó.
4. **Trong** `main()`
   - Khởi tạo mảng `arr` với các phần tử `{10, 20, 30}` .
   - `size = SIZE(arr);` xác định kích thước của mảng là `3` .
   - `fun(arr, size);` được gọi để thay đổi các phần tử trong mảng `arr` .
   - `printArr(arr, size);` in ra mảng sau khi đã được thay đổi.

Phân tích kết quả sau khi gọi `fun(arr, size);`

Dưới đây là từng bước thực hiện biểu thức trong `fun`:

1. Trước khi gọi `fun`:

```makefile
arr = {10, 20, 30}
```

2. Bên trong `fun`:
   - `*(arr + n - 1) += 10;` tương đương với `arr[2] += 10;`, nên `arr[2]` trở thành `40` (vì `30 + 10 = 40`).
   - Sau đó, `*arr += *(arr + n - 1);` tương đương với `arr[0] += arr[2];`, nên `arr[0]` trở thành `50` (vì `10 + 40 = 50`).

3. Sau khi gọi `fun`, mảng `arr` có giá trị:

```makefile
arr = {50, 20, 40}
```

4. Hàm `printArr` in ra:

```
50    20    40
```