1. **-** void \*malloc(size\_t size) digunakan untuk mengalokasikan memori secara dinamis sebesar size byte. Nilai yang dikembalikan berupa pointer ke blok memori yang dialokasikan.

**-** void \*calloc(size\_t num, size\_t size) digunakan untuk mengalokasikan memori secara dinamis untuk num elemen dengan masing-masing berukuran size byte.

**-** void \*realloc(void \*memblock, size\_t size) digunakan untuk mengubah ukuran blok memori yang sebelumnya dialokasikan dengan malloc atau calloc.

**-** void free(void \*memblock) digunakan untuk membebaskan kembali memori yang sebelumnya dialokasikan menggunakan malloc, calloc, atau realloc.

1. void (\*ptr[3])(int) = {function1, function2, function3};

Kode tersebut mendeklarasikan **array berisi 3 pointer ke fungsi**, di mana setiap fungsi memiliki parameter int dan tidak mengembalikan nilai. Dengan cara ini, kita bisa menyimpan banyak fungsi dalam satu array dan memanggilnya secara dinamis.

1.  #ifdef ... #endif  
   Digunakan untuk memeriksa apakah sebuah macro sudah didefinisikan (#define).

 #error  
Digunakan untuk menghasilkan pesan error saat kompilasi.

 #pragma  
Memberikan instruksi spesifik kepada compiler.

 #line  
Mengubah nomor baris dan nama file yang terlihat oleh compiler.

 #ifndef  
Kebalikan dari #ifdef, digunakan untuk memeriksa apakah sebuah macro **belum** didefinisikan.

1. **- \_\_LINE\_\_**  
   Menyimpan nomor baris dalam file sumber saat ini yang sedang dikompilasi.

**- \_\_FILE\_\_**  
Menyimpan nama file sumber (dalam bentuk string) yang sedang dikompilasi.

**- \_\_DATE\_\_**  
Menyimpan tanggal (dalam format mm dd yyyy) ketika file sedang dikompilasi.

**- \_\_TIME\_\_**  
Menyimpan waktu (dalam format hh:mm:ss) ketika file sedang dikompilasi.

1. **Kelebihan Macro:**

* **Lebih cepat dieksekusi** karena macro langsung diganti oleh preprocessor sebelum kompilasi sehingga tidak ada overhead pemanggilan fungsi.
* **Tidak ada tipe data yang ketat**, sehingga bisa digunakan untuk berbagai tipe data tanpa harus membuat fungsi terpisah.
* **Dapat membuat kode singkat** untuk operasi sederhana.

**Kekurangan Macro:**

* **Tidak ada pemeriksaan tipe data**, sehingga rawan error jika digunakan dengan tipe data yang salah.
* **Meningkatkan ukuran kode (code bloat)** karena macro digandakan di setiap pemanggilannya, bukan sekali seperti fungsi.
* **Sulit untuk debug**, karena macro sudah diganti oleh preprocessor sehingga sulit melacak error pada kode asli.