PreProcessorDirectives_DynamicMemoryAllocation

SUNBEAM

C Programming

Trainer : Smita Kadam

Email ID : smita@sunbeaminfo.com

PreProcessor Directives

- Any statement which starts with symbol #
- Are processed at pre-processor level
- Generates .i file with intermediate code

- **Directives:**
  - #include #define #if #else #endif
  - #ifndef #ifdef #pragma #undef
- **Operators:**

  # ##

- #define we can define symbolic constant or macro

  Syntax : #define <symbol> <replaceable text>

SUNBEAM

#include<filename>   Vs #include"filename"

- If file name is enclosed in angular bracket then file will be searched only in configured folder path of compiler where header files are available.

- If file name is enclosed in "" then file will be searched in current folder for its availability if not found then it will search in compiler's include folder.

SUNBEAM

PreProcessor Directives

- #if   #endif   #else #elseif
- Preprocessor can decide which source code can be given to compiler

Syntax:

```
#if  <expression>
     <statements>
#else
     <statements>
#endif
```

SUNBEAM

Dynamic Memory Allocation

- Dynamic Memory can be requested using functions declared in stdlib.h:
  - malloc **void * malloc (size_t);**
    - malloc requested memory is always set with default value garbage

      int *ptr = malloc(sizeof(int)) * 5);
  - calloc **void * calloc (size_t, size_t);**
    - calloc requested memory is always set with default value 0.

      int *ptr = calloc(sizeof(int),5)
  - realloc **void * realloc(void *,size_t);**

    int *x = realloc(ptr,80);

- Dynamic Memory can be deallocated using functions declared in stdlib.h
  - free **void free(void *);**

    free(p);

Dynamic Memory Allocation

Points to Note:

- Receives memory from heap section
- Returns void pointer on success and returns NULL on failure
- Programmer can request memory at runtime as and when need and becomes responsible to release memory at runtime using free() function
- Can be shrink or grown at runtime

SUNBEAM

Dynamic Memory Allocation

- Dangling Pointer:
  - Is a pointer which keeps pointing to a memory which is already deallocated. Memory once freed can be given to other process where it changes state of memory to in use mode. If not given to other process state of memory is retained in not in use mode.

- Memory Leakage:
  - If no pointer is available to access dynamic memory.

```
void fun()
{
        int *p = malloc(sizeof(int)*3);
}
```

Here on exit of function memory given for p gets released and no pointer will be available to access dynamic memory.

SUNBEAM

SUNBEAM

Thank you!