



# C PROGRAMING

*Ketan Kore*

**Sunbeam Infotech**



# 2-D array

- Logically 2-D array represents  $m \times n$  matrix i.e.  $m$  rows and  $n$  columns.
  - `int arr[3][4] = { {1, 2, 3, 4}, {10, 20, 30, 40}, {11, 22, 33, 44} };`
- Array declaration:
  - `int arr[3][4] = { {1, 2, 3, 4}, {10, 20, 30, 40}, {11, 22, 33, 44} };`
  - `int arr[3][4] = { {1, 2 }, {10}, {11, 22, 33 } };`
  - `int arr[3][4] = { 1, 2, 10, 11, 22, 33 };`
  - `int arr[ ][4] = { 1, 2, 10, 11, 22, 33 };`

	0	1	2	3
0	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
1	<b>10</b>	<b>20</b>	<b>30</b>	<b>40</b>
2	<b>11</b>	<b>22</b>	<b>33</b>	<b>44</b>



# 2-D array

- 2-D array is collection of 1-D arrays in contiguous memory locations.
  - Each element is 1-D array.
- `int arr[3][4] = { {1, 2, 3, 4}, {10, 20, 30, 40}, {11, 22, 33, 44} };`

arr	0				1				2			
	0	1	2	3	0	1	2	3	0	1	2	4
	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>10</b>	<b>20</b>	<b>30</b>	<b>40</b>	<b>11</b>	<b>22</b>	<b>33</b>	<b>44</b>
	400	404	408	412	416	420	424	428	432	436	440	444
400					416				432			



# Passing 2-D array to Functions

- 2-D array is passed to function by address.
- It can be collected in formal argument using array notation or pointer notation.
- While using array notation, giving number of rows is optional. Even though mentioned, will be ignored by compiler.



# void pointer

- Void pointer is generic pointer it can hold address of any data type (without casting).
- Scale factor of void\* is not defined, so cannot perform pointer arithmetic.
- To retrieve value of the variable need type-casting.
- void\* is used to implement generic algorithms.



# Dynamic memory allocation

- Dynamic memory allocation allow allocation of memory at runtime as per requirement.
- This memory is allocated at runtime on Heap section of process.
- Library functions used for Dynamic memory allocation are
  - malloc() – allocated memory contains garbage values.
  - calloc() – allocated memory contains zero values.
  - realloc() – allocated memory block can be resized (grow or shrink).
- All these function returns base address of allocated block as void\*.
- If function fails, it returns NULL pointer.



# Memory leakage

- If memory is allocated dynamically, but not released is said to be "memory leakage".
  - Such memory is not used by OS or any other application as well, so it is wasted.
  - In modern OS, leaked memory gets auto released when program is terminated.
  - However for long running programs (like web-servers) this memory is not freed.
  - More memory leakage reduce available memory size in the system, and thus slow down whole system.
- In Linux, valgrind tool can be used to detect memory leakage.

```
int main() {  
    int *p = (int*) malloc(20);  
    int a = 10;  
    // ...  
    p = &a; // here addr of allocated block is  
lost, so this memory can never be freed.  
    // this is memory leakage  
    // ...  
    return 0;  
}
```



# Dangling pointer

- Pointer keeping address of memory that is not valid for the application, is said to be "dangling pointer".
- Any read/write operation on this may abort the application. In Linux it is referred as "Segmentation Fault".
- Examples of dangling pointers
  - After releasing dynamically allocated memory, pointer still keeping the old address.
  - Uninitialized (local) pointer
  - Pointer holding address of local variable returned from the function.
- It is advised to assign NULL to the pointer instead of keeping it dangling.

```
int main() {  
    int *p = (int*) malloc(20);  
    // ...  
    free(p); // now p become dangling  
    // ...  
    return 0;  
}
```





# Preprocessor Directives

- Preprocessor is part of C programming toolchain/SDK.
  - Removes comments from the source code.
  - Expand source code by processing all statements starting with #.
  - Executed before compiler
- All statements starting with # are called as preprocessor directives.
  - Header file include
    - #include
  - Symbolic constants & Macros
    - #define
  - Conditional compilation
    - #if, #else, #elif, #endif
    - #ifdef #ifndef
  - Miscellaneous
    - #pragma, #error



# #include

- #include includes header files (.h) in the source code (.c).
- #include <file.h>
  - Find file in standard include directory.
  - If not found, raise error.
- #include "file.h"
  - File file in current source directory.
  - If not found, find file in standard include directory.
  - If not found, raise error.





Thank you!

Ketan Kore<[ketan.kore@sunbeaminfo.com](mailto:ketan.kore@sunbeaminfo.com)>

