UNIT I · LESSON 03

# Pointers, Structures & Files

pointers · dynamic memory · structures · unions · storage classes · preprocessor · file I/O

**20**
questions in Unit I
C · Python · C++ / OOP

## ADDRESS POINTER BASICS

- Pointer = variable that stores an **address**
- `int *p = &x;` → `p` holds address of x, `*p` is the value of x
- `&` address-of, `*` dereference (indirection)
- Size of any data pointer = **4 B (32-bit)** / **8 B (64-bit)**, whatever it points to
- `NULL` pointer points nowhere; dereferencing it → crash
- **Wild** pointer: uninitialised; **dangling**: points to freed memory
- `void *` = generic pointer; must be cast before dereferencing

```
int x = 10, *p = &x, **pp = &p;
*p = 20;          /* x = 20              */
**pp = 30;        /* x = 30 (double ptr) */
printf("%p %d", (void*)p, *p);
```

## SCALED POINTER ARITHMETIC

- `p + 1` moves by **sizeof(*p)** bytes (int* → +4)
- Allowed: ptr ± int, ptr − ptr (same array → number of elements), compare
- **Not** allowed: ptr + ptr, ptr × int, ptr / int
- `*p++` → `*(p++)`: use value, then move pointer
- `(*p)++` → increments the **value**
- `++*p` → increments value; `*++p` → move, then value

| Declaration | Meaning |
|---|---|
| **int *p[5]** | array of 5 pointers to int |
| **int (*p)[5]** | pointer to an array of 5 ints |
| **int *f()** | function returning int* |

## SCALED POINTER ARITHMETIC (CONTD.)

| Declaration | Meaning |
|---|---|
| **int (*f)()** | pointer to function returning int |
| **const int *p** | value is read-only |
| **int *const p** | pointer is read-only |

Read declarations **right-to-left** from the name: `int *const p` = "p is a const pointer to int".

## LINK POINTERS & ARRAYS / STRINGS

- `a[i] ≡ *(a + i)`; `&a[i] ≡ a + i`
- `char *s = "SIPCOT"; while (*s) putchar(*s++);`
- `&a` has type `int (*)[n]`: `&a + 1` jumps over the **whole array**

```
int len(const char *s) {   /* own strlen */
    const char *p = s;
    while (*p) p++;
    return p - s;
}
```

- Function pointer: `int (*op)(int,int) = add; op(2,3);` (callbacks, qsort)

## STDLIB.H DYNAMIC MEMORY

| Function | Action |
|---|---|
| **malloc(n)** | n bytes, **uninitialised**, returns void* |
| **calloc(k, sz)** | k×sz bytes, **zero-filled** |
| **realloc(p, n)** | resize block (may move it) |
| **free(p)** | release block |

```
int *a = (int*) malloc(n * sizeof(int));
if (a == NULL) { /* allocation failed */ }
/* ... use a[0..n-1] ... */
free(a); a = NULL;     /* no dangling */
```

- Memory from heap; forgetting `free` → **memory leak**
- `free` twice → undefined; freeing a non-heap pointer → undefined

## SHARED MEMORY UNIONS

| struct | union |
|---|---|
| **each member has own memory** | all members share one memory |
| **size ≥ sum of members** | size = largest member (+ padding) |
| **all members usable together** | only **one** member valid at a time |

```
union u { int i; char c[4]; float f; };
/* sizeof(union u) = 4 */
```

- **enum**: `enum day {MON, TUE=5, WED};` → MON=0, TUE=5, WED=6

## STRUCT STRUCTURES

```
struct student {
    int roll;
    char name[20];
    float marks;
} s1 = {1, "Arun", 88.5};
struct student *p = &s1;
printf("%s %d", s1.name, p->roll);
```

- Members accessed with `.` (variable) and `->` (pointer): `p->roll ≡ (*p).roll`
- **Size ≥ sum of members** because of **padding/alignment**
- `typedef struct student Stu;` creates an alias
- Structures can be assigned (`s2 = s1`) but **not compared** with `==`
- Self-referential: `struct node { int d; struct node *next; };`
- **Bit fields**: `unsigned flag : 1;`

## SIZEOF PADDING EXAMPLE

| Struct (32/64-bit, 4-B int) | sizeof |
|---|---|
| `{char c; int i;}` | 8 |
| `{char a; char b; int i;}` | 8 |
| `{char a; int i; char b;}` | 12 |
| `{double d; char c;}` | 16 |
| `{char c[5];}` | 5 |

Order members **largest first** to reduce padding.

## MAIN COMMAND-LINE ARGUMENTS

- `int main(int argc, char *argv[])`
- `argc` = count **including** program name; `argv[0]` = program name; `argv[argc]` = NULL
- `./a.out x y` → argc = 3

## 4 KINDS STORAGE CLASSES

| Class | Storage | Default | Scope | Lifetime |
|---|---|---|---|---|
| **auto** | stack | garbage | block | block |
| **register** | CPU register* | garbage | block | block |
| **static** | data segment | 0 | block / file | program |
| **extern** | data segment | 0 | global (other files) | program |

- `register` variables have **no address** (`&` not allowed)
- `static` global/function → visible **only in that file** (internal linkage)
- `extern int x;` declares only, no memory allocated
- *register is only a request to the compiler

## # DIRECTIVES PREPROCESSOR

| Directive | Use |
|---|---|
| **#include** | insert a file ( `<>` system, `""` local first) |
| **#define** | macro / constant |
| **#undef** | remove macro |
| **#ifdef / #ifndef / #endif** | conditional compilation |
| **#if / #elif / #else** | conditional on expression |
| **#pragma** | compiler-specific |
| **#error** | stop with message |

- Macros are **text substitution** with no type checking
- `#define SQ(x) x*x` → `SQ(2+3)` = 2+3*2+3 = **11**; write `((x)*(x))`
- `#` stringizes, `##` pastes tokens; predefined `__FILE__ __LINE__ __DATE__`
- Include guard: `#ifndef H_ / #define H_ / … / #endif`

## POINTERS PREDICT THE OUTPUT

| Code | Output |
|---|---|
| `int a[]={10,20,30}; int *p=a; printf("%d",*p+1);` | 11 |
| same, `printf("%d",*(p+1));` | 20 |
| `printf("%d", *p++); printf("%d",*p);` | 10 then 20 |
| `char s[]="ABC"; printf("%c", *(s+2));` | C |
| `int x=5,*p=&x; *p*=2; printf("%d",x);` | 10 |
| `int a[3][2]; printf("%d", sizeof(a)/sizeof(a[0]));` | 3 |
| `printf("%d", sizeof(struct{char c; int i;}));` | 8 |

## STDIO.H FILE HANDLING

| Mode | Meaning |
|---|---|
| **"r"** | read; file must exist |
| **"w"** | write; creates or **truncates** |
| **"a"** | append; creates if missing |
| **"r+"** | read + write, must exist |
| **"w+"** | read + write, truncates |
| **"a+"** | read + append |
| **"rb", "wb"** | binary modes |

| Function | Use |
|---|---|
| **fopen / fclose** | open (returns FILE* or NULL) / close |

## STDIO.H FILE HANDLING (CONTD.)

| Function | Use |
|---|---|
| **fgetc / fputc** | character |
| **fgets / fputs** | string / line |
| **fprintf / fscanf** | formatted |
| **fread / fwrite** | binary blocks |
| **fseek(fp, off, SEEK_SET\|CUR\|END)** | move position |
| **ftell / rewind** | position / go to start |
| **feof / ferror** | end-of-file / error test |

```c
FILE *fp = fopen("data.txt", "r");
int ch, n = 0;
if (fp == NULL) return 1;
while ((ch = fgetc(fp)) != EOF) n++;
fclose(fp);            /* n = chars */
```

`EOF` is **-1**, so store `fgetc` result in an **int**, not a char.

## TRAP EXAM TRAPS

- `int *p; *p = 5;` → wild pointer write
- `int* p, q;` → only p is a pointer
- `sizeof(p)` on `int *p = malloc(40)` gives 4/8, not 40
- `p++` on an array name `a++` → error (constant)
- `malloc` returns NULL on failure; `free(NULL)` is safe
- Union: write `i`, read `f` → reinterprets bytes
- `"w"` mode erases an existing file
- Macro arguments may be evaluated twice: `MAX(i++, j)`

## DEFINITIONS TYPEDEF, ENUM & CONST

| Keyword | Example | Effect |
|---|---|---|
| **typedef** | `typedef unsigned int uint;` | alias for a type |
| **enum** | `enum color {RED, GREEN=4, BLUE};` | RED=0, GREEN=4, BLUE=5 |
| **const** | `const float g = 9.8;` | read-only after init |
| **volatile** | `volatile int flag;` | value may change outside program (hardware, ISR) |

- `typedef` is handled by the **compiler**; `#define` by the **preprocessor**

## SELF-REFERENTIAL LINKED LIST NODE IN C

```c
struct node { int data; struct node *next; };
struct node *push(struct node *head, int x) {
    struct node *t = malloc(sizeof *t);
    t->data = x;
    t->next = head;     /* insert at front */
    return t;
}
void show(struct node *h) {
    for (; h != NULL; h = h->next)
        printf("%d ", h->data);
}
```

- Insert at front O(1); search O(n); free nodes one by one

## ROWS OF POINTERS DYNAMIC 2-D ARRAY

```c
int **m = malloc(r * sizeof(int *));
for (i = 0; i < r; i++)
    m[i] = calloc(c, sizeof(int));
/* use m[i][j] ... then free rows, then m */
for (i = 0; i < r; i++) free(m[i]);
free(m);
```

- **calloc** is preferred when zero initial values are needed