Pointers to pointers

**NULL pointer**

- **ppc:** 0x14111988
- **pc:** 0xFFA83058
- **c:** 0xFFA83D5B

```c
char c;
char *pc;
char **ppc;
c = 'A';
pc = &c;
ppc = &pc;
```

**Pointer arrays**

```c
char *pa[5];
```

- **pa[0]:** abc d
- **pa[4]:** h

**Two dimensional arrays**

```c
int a2d[2][4];
```
char *name[] = {
    "Illegal month",
    "Jan",
    "Feb",
    "Mar"
};

char aname[][15] = {
    "Illegal month",
    "Jan",
    "Feb",
    "Mar"
};

Command line arguments

argc - argument count.
argv - argument vector.

argv

$ ./hello Gerald Chennai
int main(int argc, char* argv[]) {

    // Code here

}

division function

\[ \frac{x}{y} \rightarrow \text{Quotient} \quad \frac{x}{y} \rightarrow \text{Remainder} \quad x \div y \]

e.g. 5 \rightarrow \text{div} \rightarrow Q = 2
    2 \rightarrow \text{div} \rightarrow R = 1

Design the function prototype for divide method function.
Structures

\[
\begin{align*}
\text{struct point} & \{ \\
\text{int} & \ x; \\
\text{int} & \ y; \\
\} \\
\end{align*}
\]

\( x, y \) - Members of the structure.

point - Structure tag/name.

A struct declaration - defines a type.

\[
\text{struct point} \quad \text{pt}, \ \text{p2}; \\
\text{int} \quad x1, x2; \\
\]

Accessing members

\[
\text{struct point pt;} \\
\text{pt.x} \ \text{2 used to access} \\
\text{pt.y} \ \text{x and y}. \\
\]

\( \Rightarrow \) Structure member operator.

\[
\text{printf} \left( \text{"\%.d, \%.d", pt.x, pt.y}; \right) \\
\]
Distance of $(x, y)$ from $(0, 0)$

```java
double dist, sqrt(double);

dist = sqrt((double) pt.x * pt.x + (double) pt.y * pt.y);
```

**Nested Structures**

```
struct rect {
    struct point pt1;
    struct point pt2;
}
```

```
struct rect screen;

screen.pt1.x = 1;
```

**Structures and Functions**

Legal operations on structs:

1. copying a struct
2. assigning to it as a unit
3. taking the address of a struct using &
4. accessing its members.
struct point addpoint (struct point p1, struct point p2) {
    struct point temp;
    temp.x = p1.x + p2.x;
    temp.y = p1.y + p2.y;
    return temp;
}

Pointers to structures

Analogy:
- take laptop → Service Center
  - if laptop in repair
  - OK ("small object")
- give your address
  - service provider comes to your house
It is OK to copy small structures between functions.

Not OK to copy large structures.

Use pointers for doing this!

Address of large structure.

Syntax

\[
\begin{align*}
\text{struct point} & \rightarrow x & \rightarrow \text{pt}.x \\
\text{struct point} & \rightarrow y & \rightarrow \text{pt}.y
\end{align*}
\]

Better syntax

\[
\begin{align*}
\ast pp \Rightarrow \text{structure} (pt) \\
(\ast pp) \cdot x & \Rightarrow \text{pt}.x \\
(\ast pp) \cdot y & \Rightarrow \text{pt}.y
\end{align*}
\]
Arrays of structures

struct studinfo
{
    #name;
    #unsigned int age;
}stud_detail;

struct stud_info
{
    name age
}

student [160];

160 students in CS 354 section 2.
Self-referential Structures

struct node {
    int data;
    struct node *next;
};

Example:

head

Ox1000

42 Ox2000

Ox1000

Ox2000

14 NULL
struct node {
    int data;
    struct node subnode;
}

 Illegal!

But pointer to the same structure is legal!