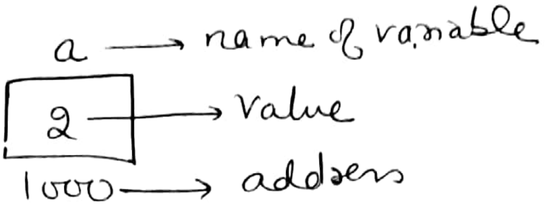
Lecture – 13

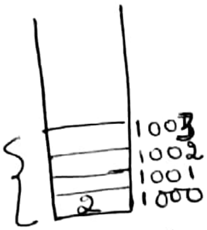
Before linke list

Pointers

int a ;

a= 2





4 byte space for a if we are using LINUX OS.

Pointers are also a variable which hold address of another variable.

* → value at (value operator)

& → address

Declaration of pointer variable

data type * variable name ;

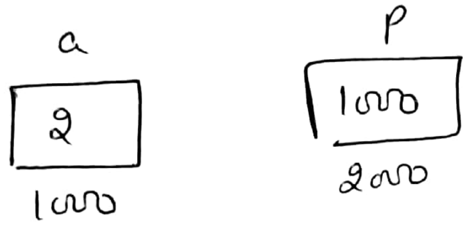
eg int * p ;

- pointers always hold address, still we define the data type at the declaration of pointer variable because with help of data type we get to know ~~that~~ ~~the address hold by pointer~~ the type of variable whose address it will hold i.e whether the variable is of int, char, float, string, [illegible] linke list & etc.

for enample

```
int a= 2 ;
int * p; // here p is a pointer variable
            that will hold addres of
            a variable integer type.

p = &a ; // assign the addres of
            variable a to p.
```



when we display

p → it will show 1000

*p → it will show 2

∴ * means value at

∴ *(1000) ⇒ $\underline{\underline{2}}$

&a → will print addres of a i.e 1000

a → will print value 2

*(&a) → will print value 2

└→ value at this location.

- How many bytes a pointer used in memory or you can say the space taken by a pointer variable ??

⇒ It depend upon the compiler

⇒ Whether the pointer variable is of int, char, float whatever, they will have same storage capacity. ∵ they had to just hold the address.

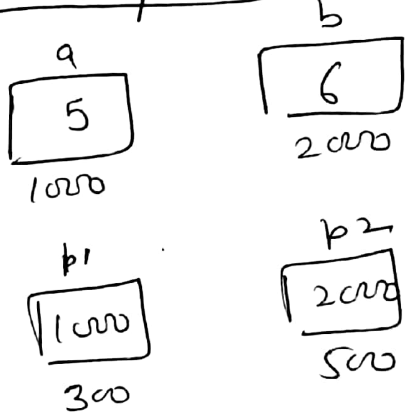
⇒ Depending upon the compiler

4 byte ⇒ If it is 32 bit compiler

8 byte ⇒ If it is 64 bit compiler.

- <u>Arithmetic operations on pointers</u>

```
int a, b;
int *p1, *p2;
a=5, b=6;
p1=&a;
p2=&b
```



sum = *p1 + *p2 // It is possible

As we apply arithmetic operation on these values.

But

p1 + p2

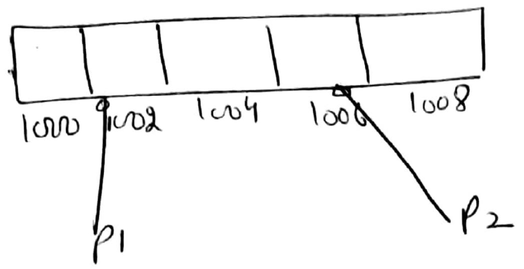
1000 + 2000

⇒ 3000

Invalid.

As we don't at this location what we have or whether this exist or not.

- If two pointers P_1 & P_2 are pointing on a array than subtraction can be performed.



$p_2 - p_1 \Rightarrow 1006 - 1002 \Rightarrow \underline{\underline{4}}$

It will give $\underline{\underline{2}}$ ~~you~~ that a result, to show that 2 integer value are stored or can be stored

- P_1++ || can be performed.

or P_1+1 || start pointing to next location.
$\underline{\underline{\text{ie}}}$ according to data type it will increment the address.

- P_1-- || can be performed.

<u>Therefore</u>, on values any arithmatic operator can be applied

<u>But</u> on address only subtraction, increment & decrement.

Dynamic Memory Allocation

int a[10];

- It will reserve memory for storing 10 elements
- If we read only 5, then our memory is wasted.
- If we want to read more than 10 than our array will fall short in size. Infact, arrays don't have any bound checking so if we read more than 10, than the 11th element may be placed at some important data.
- If we want to allocate memory at the time of execution. It can be done by standard library function called :→

  malloc() and calloc()
- For this we have to add a header file

  # include <alloc.h>.

Eg

```
int *p, n, i;
scanf("%d", &n);
p = (int *) malloc (n * size of (int));
for( i=0; i<n; i++)
printf("%d", *(p+i));
```

- malloc() function return null if memory allocation is unsuccefull otherwise it will return address of memory chunk that is allocated.

- Since malloc() return void pointer

  void pointer ⇒ size of the type is unknown i.e char, float, int.

Therefore it can be done appropriately by typecasting.

So we typecast the malloc() function according to the requirement.

eg (int *) malloc (n * size of (int))
(float *) size of (float)
(char *) size of (char))

- It did not initialize all byte position with any value.

  i.e if we print ("%d", *(p+i));

  ⇓

  It will print garbage value.

~~Rea~~ Calloc ( ) ; function

- Only format is different

  $p = (\text{int} *)\ \text{calloc}\ (\underbrace{n\ ,\ \text{size of}}\ (\text{int}))\ ;$

  ⇑

  It always need 2 argument as we have to apply comma operator.

- It intialize all the memory position by zero. [By default it initialize with 0]

  ie ~~printf~~ printf("%d", *(p+i));

  ⇓

  It will print zero's

Structures

```
struct book
{
    char name;
    float price;
    int pages;
};
```

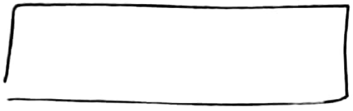
- Till now no space is allocated. as we has just declared.
- When we create object or pointers say struct book b1, b2, b3;

- Now the space will be allocated to each object.

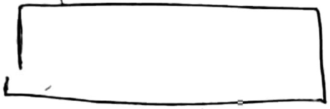
In case of window OS   7 byte { 1 byte for char
2 → Int
4 → float

In case of Linux OS   9 bytes { 1 → char
4 → Int
4 → float

ie   b1 will 7 byte or 9 byte acc. to OS



b2 will have 7 or 9



& same for b3.

- This space is adjacent memory location.

ie chunk of 7byte or 9 byte acc. to OS will have adjacent memory location.

Typedef command:-

typedef int ABC;

Now we can use ABC a;

↑ It will act as data type of integer type

typedef struct Emp
{
≡
} e1;

↑ Now it will act as data type of struct type

Now we can declare the object as

e1 b1, b2, b3;