Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

Date
Page

* Pointer -

A pointer is a special variable that is used to store the address of some other variable.

* A pointer can be used to store the address of a single variable, array, structure, union or even a pointer.

* A pointer is a derived data type in C.

* Pointers allow C to support dynamic memory management

* Declaration of pointer variables -

syntax - data type ~~to~~ name * pointer name

(Here '*' (asterisk) before pointer indicate the compiler that variable declared as a pointer)

Eg:-
```
int *P1;          // pointer to integer type
~~int~~ float *P2;  // pointer to float type
char * P3;        // pointer to character type
```

(When pointer declared, it contains garbage value i.e it may point any value in the memory)

* int *P1; (Here the type int refers to the data type of the variable which is pointed by P1 not the type of the value of the pointer.

* Pointer ~~declation~~ declaration style -

1) int* P ;
2) int *P;
3) int * P;

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

* There are two operators are used in the pointer -

i) Address operator (&) -

Address operator is used to access the address of a variable. It can be used only with a simple variable or an array element not with a constant value.

* Indirection operator(*) -

It gives the value stored at a particular address.

Eg:- ~~main ()~~
~~{~~ ~~int a = 5;~~
~~int~~

* Initialization of pointer variable -

```
Eg:-   → int a;
         int *P;     //declaration
         P = &a      // initialization

→ or, int x, *P = &x ;  /* declaration with
                              initialization */

→ int *P = &x, x; (Not valid, because target
                   variable x should be declared
                   first.)

→ float a;
  int *P;
  P = &a; (Not valid, because we can not
           assign the address of a flot variable
           to an integer pointer.)
```

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

%x = hexadecimal format

```
float a;
float *P;
P = &a;     (It is valid.)
```

Eg:-

```
main()
{ int a=10,b=9;
  int *P, *q;
  P = &a;
  q = &b;
  c = *P;
printf("Value of a = %d\n", a);          a = 10
printf("Value of a = %d\n", *P);         a = 10
printf("Address of a = %x\n", &a);       61fec4
printf("Address of a = %x\n", P);        61fec4
printf("Address of P = %x\n", &P);       61fec0
printf("C = %d\n", c);                   a = 10
}
```

```
* P = &a, &b;
  printf("Value of a = %d\n, *P);        a = 10

* P = (&a, &b);   (Due to comma operator it prints the value of 'b')
  printf("Value of b = %d\n", *P);       b = 9

* P = &a;
  P = &b;
  printf("Value of = %d\n", *P);         b = 9
```

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

Date ____
Page ____

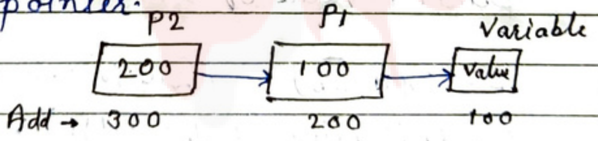
Eg:-
```
main()
int a = 10, b = 5;
 int *p, *q;
  p = &a;
  q = p;     (*q = *p) → (It is invalid, because
                          'q' is not initialize yet.)
printf("a = %d %d %d \n", a, *p, *q);
                    output - a = 10, 10, 10
```

*
```
  q = &b;
  *q = *p;   (Valid)
 printf("a = %d", *q);   output - a = 10 (not b = 5)
                                   ✓        ✗
```

* Pointer to pointer (chain of pointers) -

As we know, pointers stores the address of a variable, similarly the address of a variable pointer can also be stored in some other pointer, it is called pointer to pointer variable.

'or' Pointer within another pointer is called pointer to pointer.



Eg:-
```
  main()
  { int a = 5;
    int *P1;  int** P2;  int ***P3;
    P1 = &a;
    P2 = &P1;
    P3 = &P2;                              output
  printf("%d", a);                           5
 printf("%d", *P1);                          5
 printf("%d", **P2);                         5
 printf("%d" ***P3);                         5
} printf("%x", *P2);              address of P1 will be printed
```

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

* A two level pointer(**P2) always stores one level pointer (*P1) variable not normal variable (&a).

* Similarly, three level pointer(***P3) always stores two level(**) pointers.

Eg:-
```
main()
{
  int a = 10;
  int *p = &a;
  int **q = &p;
  int ***r = &q;          (***r = &p → Not allowed)
  printf("a = %d\n", a);                        10
  printf("a = %d\n", *p);                       10
  printf("a = %d\n", *(*q));                    10
  printf("a = %d\n", *(*(*r)));                 10
  printf("Address of q = %x\n", r);     print address of 'q'
  printf(______________________", &q);          ,,
  printf("Address of P = %x\n", q);     print address of 'p'
  printf("______________________", &P);         ,,
  printf("Address of a = %x\n",&a);     print address of 'a'
  printf ("______________________", P);         ,,
  printf ("Address of r = %x\n" &r);    print address of 'r'
```

*
```
int a = 10;
int *P = &a;
  *P = 12;
Value of a = 12
```

*
```
int a = 10
int *P = &a;
int** q = &P;
  *P = 12;
  **q = 17;
value of a = 17
```

*
```
int a = 10;
int *P = &a;
int **q = &P;
int ***r = &q;
   *P = 12;
  **q = 17;
 *** r = 78;
value of a = 78;
```

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

Date ____
Page ____

* Pointer Arithmetic -

→ Pointer addition - (Performing on array)

```
main()
{   int a[5] = {1, 4, 2, -8, 0};
     int *p = &a[0];    //or *p = a; (Both are valid.)
     int *q = &a[0];                                   output
   printf("Value is %d =\n", *p);                    → 1
        p = p + 2     // forward two position)
   printf("Value is = %d\n", *p);                    → 2
   printf("Value is = %d\n", *p + *q);               → 3
   printf("Value is = %d\n", p+q);
                                    Not allowed (Invalid)
                                     in addition.
```

* *p + *q
   ↓   ↓
   2 + 1 = 3

(In this case, we add the value which are present in pointers variable p and q.)

* p + q
(We can not add two pointers)

*

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

Date
Page

* Pointer subtraction -

```
main()
{ int a[] = {2, 4, 3, 0, -7};
  int *p = a ;
  int *q = &a[3] ;                                     output
printf ("q-p: %d", q-p);                      →         3
/* It is valid in subtraction of pointers and it
          gives the no. of elements b/w P & q */
printf (" P-q = %d", P-q );                   →       -3
printf(" Value = %d\n", *q);                  →      0  (a[3] = 0)
        q = q - 2      // Backward two position
  printf ("Value = %d\n", *q) ; → 4
```

* If P1 and P2 are both pointers to the same array, then "P2 - P1" gives the no. of elements b/w P1 & P2.

* q = q - 2 (Decrement the pointer means backward two position.)

* q = *q - 2 (Decrement the value which are present in 'q' pointer means here subtraction occures.)

If 'p' and 'q' are two pointers then -

* a = P + q ; (x)

a = *P + *q ; (✓)

a = *P + 2 ; (✓)

a = P - q ; (✓)

a = *P - *q ; (✓)

a = *P - 2 ; (✓)

a = *P * *q ; (✓)

a = P * q ; (x)

a = *P / *q ; (✓)   a = P / q ; (x)

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

Date
Page

* Pointer increment or decrement -

```
main()
{  int a[3] = {3,2, 67, 0, 56} ;
       int  *P ;
         P = a ;
     printf("%d\n", ~~*P(++~~ *(P++)) ;        3
    printf("%d\n", *P) ;          2
  printf("%d %d\n", *(P++), *P++) ;        2, 3
                          <-----------
     /* Execution process start right to left means
                 first *P++ executed then *(P++)*/
  printf("%d\n", * ++P );
  printf( ________" *P);
```

Being Pro

For more PDFs and computer notes.. search "beingpro33" on Telegram page.

Date
Page

* Void pointer-

In void pointer, we can assign different types of data types using typecast.

Eg:-
```
main()
{
  void *vp;
  int a=5; float b=1.56, char ch='c';
  vp = &a;
printf("a = %d\n", *(int*)vp);          a=5
  vp = &b;
printf("b = %f\n", *(float*)vp);        b=1.560000
  vp = &ch;
printf("ch = %c\n", *(char*)vp);        ch=c
```