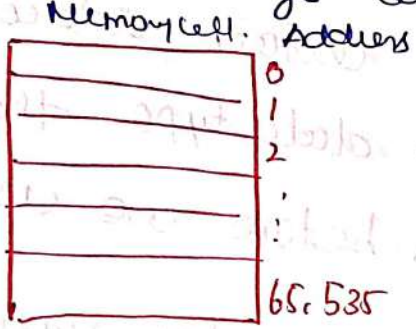
Pointers

→ It is a derived datatype in C.

→ A pointer is a variable which stores the address of another variable.

understanding Pointers:

→ A compiler's memory is a sequential collection of storage cells.



→ The addresses are numbered consecutively starting from zero. The address depends on the memory size.

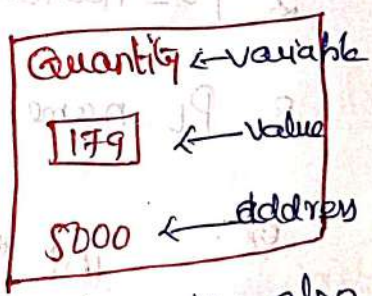
→ A computer s/m having 64K memory will have its last address as 65535.

→ Consider the following statement.

int quantity = 179.

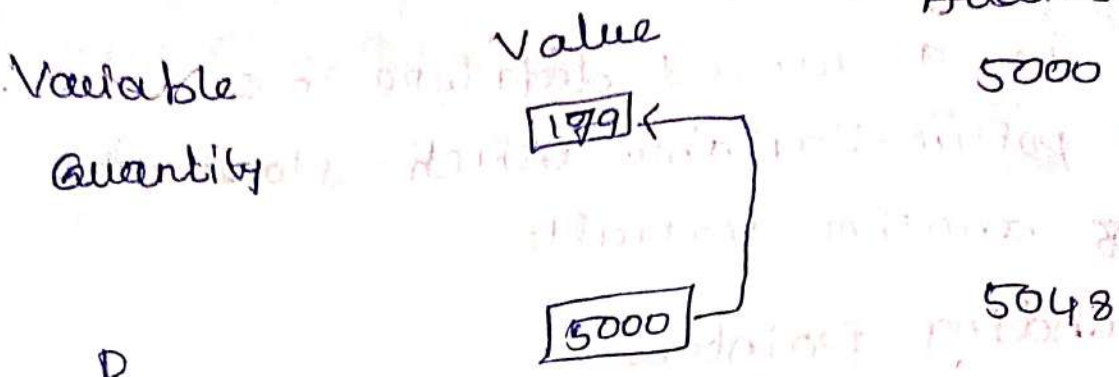
int - datatype

quantity - variable name.



→ A pointer is a variable its value is also stored in the memory in another location.

Suppose we assign the address of quantity to a variable P.



Declaration pointer variable :-

pointer variable certain address that belong to a separate data type that must be declared as pointers before we use them.

The declaration of a pointer variablesthat takes the following form.

```
data_type *pt_name;
```

1. The asterisk (*) tells that the variable pt_name is a pointer variable.
2. pt_name points to a variable of type data type
3. Pt_name need a memory location

ex:

```
int *P ; /* integer pointer */
float *x; /* float pointer */
```

Accessing The address of variable (&)

→ To access address of variable to pointer we use the unary operator (&) (ampersand) that returns the address of that variable.

For ex: The statement

p = & quantity;

Initialization of pointer variable:

→ The process assigning the address of a variable to a pointer variable is known as initialization.

→ we use the (&) address of operator to get the memory address of a variable and then store it in the pointer variable.

Ex:

```
int quantity;   pb
int *p;    /* declaration of pointer variable */
p= & quantity;  /* Initialization */
```

Ex

```
float a,b;
int x, *p;
p = &a;  /* wrong */
```

Sample programs:

```
#include <stdio.h>
#include <conio.h>
Void main()
{
    int x, y;
    int * ptr;
    x = 10;
    ptr = &x;
    y = *ptr;
    Printf ("In Value of x is %d", x);
    Printf (" %d is Stored at addr %u", x, &x);
    Printf (" %d is Stored at addr %u", *x, &x);
    Printf (" %d is Stored at addr %u", *ptr, ptr);
    Printf (" %d is Stored at addr %u", ptr, &ptr);
    Printf (" %d is Stored at addr %u", y, &y);
    *ptr = 25;
    Printf ("In Now x = %d", x);
    }
```

O/P:

The value of x is 10

10 is Stored at addr 4104.

10 is Stored at addr 4104

10 is Stored at addr 4104

4104 is Stored at addr 4106

10 is Stored at addr 4103

Now x = 25.

Pointer Expression:

→ The pointer variable can also be used in expression:

```
*p1 = *p1 + 1;
sum = *p1 + *p2;
*p1 = *p2 *2
```

```
#include <stdio.h>
int main()
{ int n1 = 2, n2 = 3, sum = 0, mul = 0, div = 1;
  int *p1, *p2;
  p1 = &n1;
  p2 = &n2;
  sum = *p1 + *p2;
  mul = sum * *p1;
  div = 9 + *p1 / *p2 - 30;
  printf ("\n sum = %d mul = %d div = %d",
                                sum, mul, div);
  return 0;
}
```

O/P:

sum = 5, mul = 10, div = -29.

Null pointers:

A pointer that doesnot point to any valid memory address is called null pointer

```
int *ptr = NULL;
```

The value of null may be taken as zero.