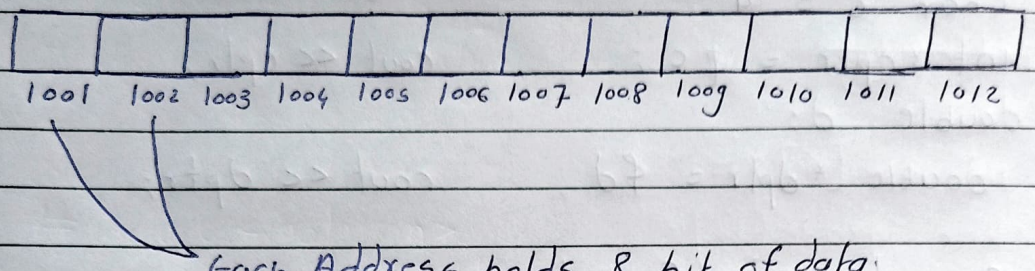
13/02/2023

Day 7

Pointers :

Address of variable

int a ; a gets some space in memory



Each Address holds 8 bit of data

```
int a;
cout << "Address of variable a " << &a <<endl;
```

A pointer variable (or pointer) is basically the same as the other variables, which can store a piece of data Unlike normal variable which stores a value (such as int, a double, a char) a pointer stores memory address.

Delaring pointers :

1) datatype *ptr ; => datatype of variable whose address is getting stored in ptr.
2) datatype* ptr ;
3) datatype *ptr ;

Indirection or Dereferencing operator (*)

The indirection operator (or dereferencing operator) operates on a pointer, a return the value stored in the address kept in the pointer variable.

* Initialize pointers

```
int a = 10
int *iptr = &a;          cout << iptr;
char c = 'a';
int *cptr = &c;          cout << cptr;
double d;
double *dptr = &d;       cout << dptr;
```

* Operations performed on pointer:
  1) Increment operators
  2) Decrement operators
  3) Addition of a constant to a pointer
  4) Subtraction of constant to a pointer
  5) Subtraction of two pointers
  6) Comparison of two pointers.

* Null pointers-
  We can initialize a pointer to 0 or NULL i.e it points nothing
  It is called a null pointer. Dereferencing a null pointer (*p) causes on STATUS_ACCESS_VIO ATION exception.

```
int * iptr = 0;
cout << *iptr << endl;
int *p = NULL;
```

* Void pointers -

The void type of pointer is a special type of pointer. In c++ void represents the absence of type. Therefore void pointers are pointers that point to a value that has no type.

```
int a = 10;
double d;
void *vptr;
vptr = &a;
vptr = &d;
*vptr;          --> // error, cannot access element
*vptr++;        --> // error, cannot increment void pointe
int *iptr = static_cast<int*>(pv);
cout << *iptr;
```

* Pointer to pointer - We can use pointer store other pointers (a variable that is storing some address)

```
int a = 10;
int *iptr = &a;
int **iiptr = &iptr;
cout << iiptr;
cout << *iiptr;
cout << **iiptr;
cout << iptr;
cout << *iptr;
```