C tokens

Keywords (words having specific purpose, can be randomly used).

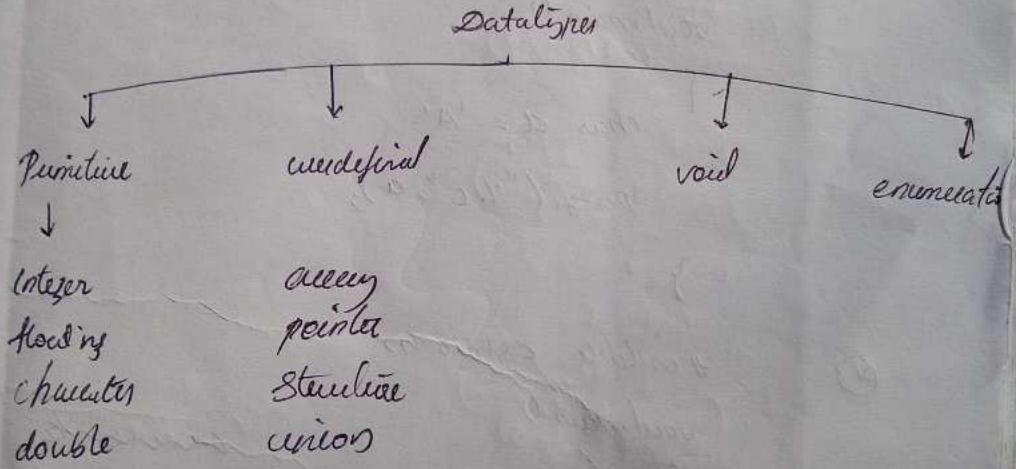
constants (const → variables cannot be changed)

Special symbols (& % \)

strings

operators

identifiers



Primitive Datatypes		Format specifier
Integer — int	4 bytes	%d
character — char	1 bytes	%c
floating — float	4 bytes	%f
double —	8 bytes	

Ascii values

0-9 ⇒ 48-57

A-Z ⇒ 65-90

a-z ⇒ 97-122

(*)
```
#include <stdio.h>
void main()
{
  int a;
  a = 10;
  printf ("%d", a);
}
```

(2)
```
#include <stdio.h>
void main()
{
  char a = 'A'
  printf ("%c", a);
}
```

(4)
```
#include <stdio.h>
void main()
{
  float a = 1.222;
  printf = ("%f", a);
}
```

%.2f ⇒ 2 decimal will display

```
#include <stdio.h>
void main ()
{
  int a,b;
  Printf(" enter two numbers");
  Scanf ("%d,%d", &a, &b);
  Printf(" sum is = %d", c);
```

→ Relation operators

<, >, <=, >=, ==, !=

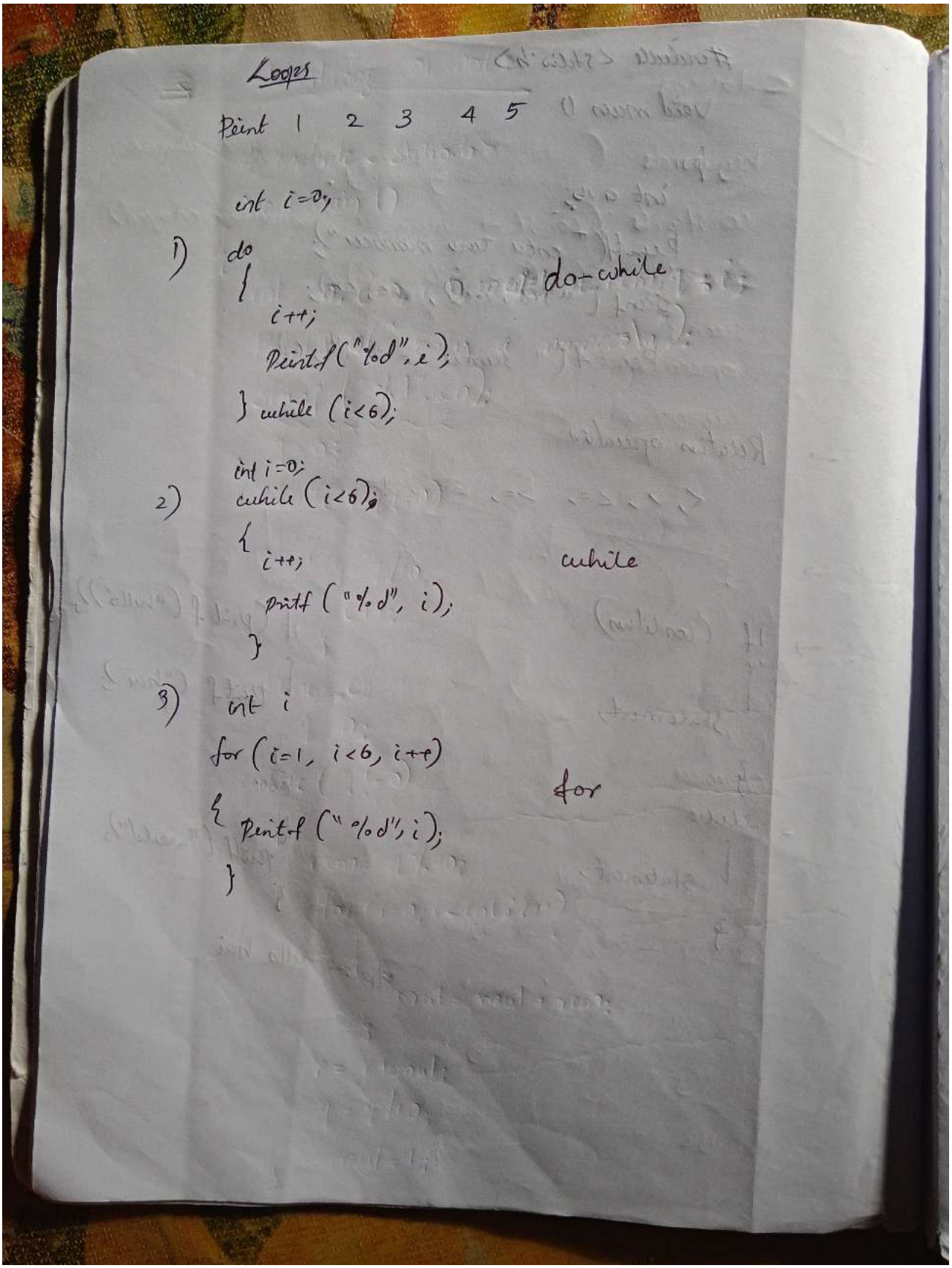
→

→ If (condition)

```
{
  statement;
}
else
{
  statement;
}
```

```
if (printf ("hello"))
{
  printf ("hai");
}
else
{
  printf ("world");
}
```

o/p :- hello hai



Loops

Print 1 2 3 4 5

```
int i=0;

1) do
   {
     i++;
     Printf("%d", i);
   } while (i<6);

int i=0;
2) while (i<6);
   {
     i++;
     printf ("%d", i);
   }

3) int i
   for (i=1, i<6, i++)
   { Printf ("%d", i);
   }
```

do-while

while

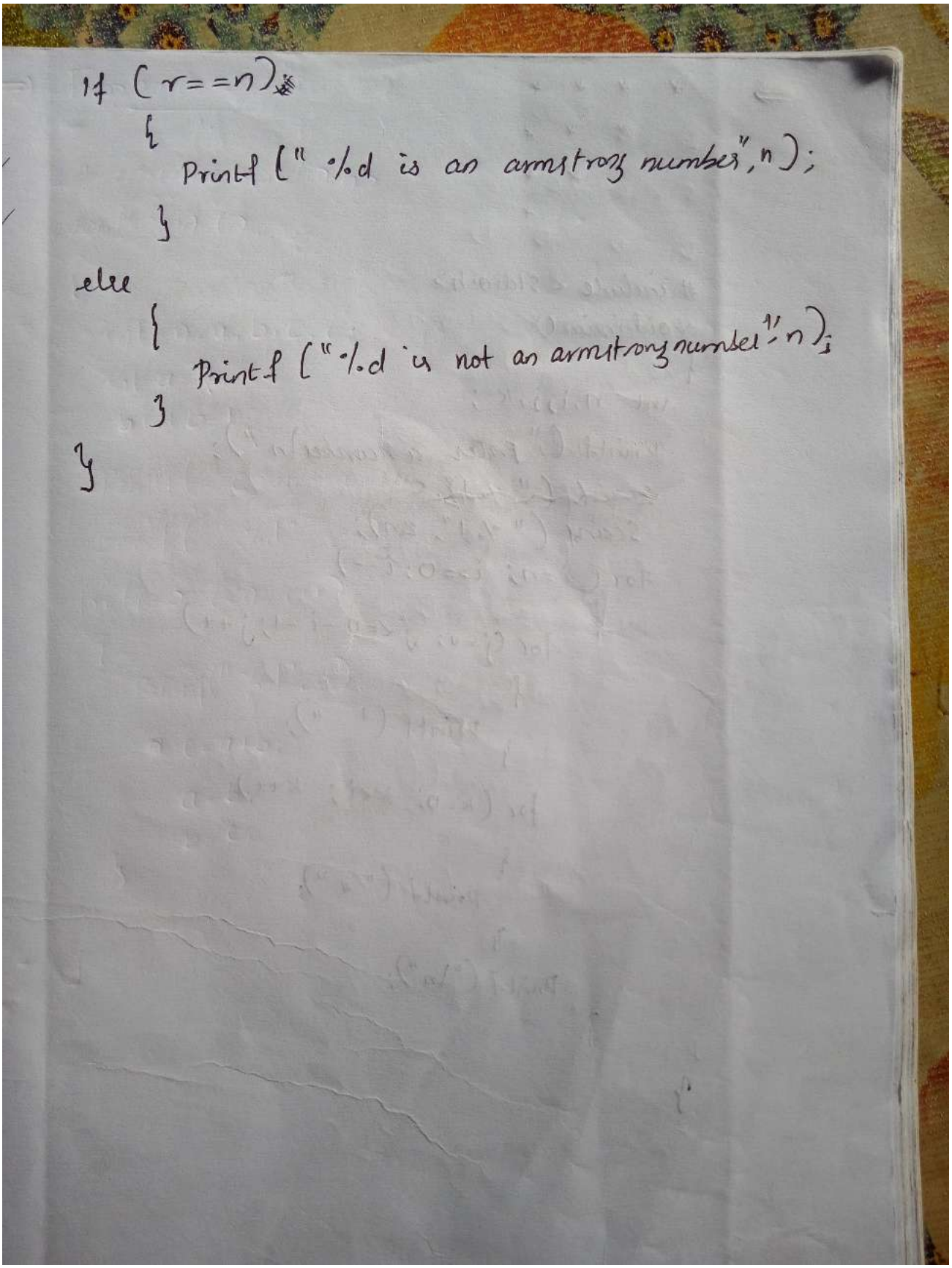
for

⇒ Fibonacci Series

```
# include <stdio.h>
void main ()
{
  int n, a, b, c, i;
  int a=0;
  int b=1;
  Printf("Enter a limit");
  Scanf("%d", &n);
for (i=0; i<n; i++)
  {
    Printf("%d", a);
    c=a+b;
    a=b;
    b=c;
  }
}
```

⇒ Armstrong or not:

```
# include <stdio.h>
Void main ()
{
int n, c=0, r=0, mul=1, cnt, rem, q, i;
Printf (" Enter a number\n");
scanf ("%d", &n);
 q=n;
cuhile (q!=0)
   {
     q= q/10;
      c++;
   }
 cnt= c;
  q=n;
cuhile (q!=0)
 {
   rem= q%10;
   for (i=0; i<cnt; i++)
     {
       mul = mul * rem;
     }
   r= r+mul;
   q= q/10;
 }  mul=1;
```



```
if (r==n)
    {
      Printf (" %d is an armstrong number", n);
    }
else
    {
      Printf ("%d is not an armstrong number", n);
    }
}
```

⇒

```
* * * *
  * * *
    * *
      *
```

```
# include <stdio.h>
void main()
{
  int n,i,j,k;
  Printf ("enter a number\n");
  scanf ("%d", &n);
  for (i=n; i>=0; i--)
     {
       for (j=0; j<=n-i-1; j++)
         {
           Printf (" ");
         }
       for (k=0; k<i; k++)
         {
           printf ("*");
         }
       Printf ("\n");
     }
}
```

⇒

```
1
2 2
3 3 3
4 4 4 4
```

```
# include <stdio.h>
void main()
{
    int n, i, j;
    printf ("enter a number \n");
    scanf ("%d", &n);
    for (i = 1; i <= n; i++)
        {
            for(j=0; j<i; j++)
                {
                    printf ("%d", i);
                }
            printf ("\n");
        }
}
```

→

A

B B

C C C

D D D D

```
#include <stdio.h>
void main()
{
  int n, i, j, K = 65;
  printf("enter a number");
  scanf("%d", &n);
  for (i=1; i<=n; i++)
    { for (j=0; j<i; j++)
      {
        printf("%c", K);
      }
      printf("\n");
      K++;
    }
}
```

⇒ A
B C
D E F
G H I J

```
#include <stdio.h>
void main ()
{
  int n,i,j,k=65;
  Printf ("enter a number");
  Scanf ("%d",&n);
  for (i=1; i<=n; i++)
   {
     for(j=0;j<i;j++)
      {
        printf ("%c", k);
        k++;
      }
     printf ("\n");
   }
}
```

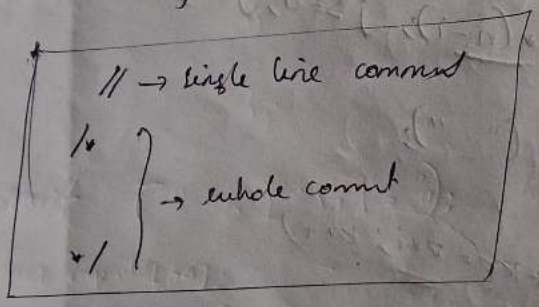
⇒

```
# include <stdio.h>
void main ()
{ int n,i,j,k;
  Printf("enter a number\n");
  scanf("%d",&n);
  for(i=0; i<=n; i++)
    { for(j=0; j<=n-i-1; j++)
      { printf(" ");
      }
      for(k=0; k<=i; k++)
      {
        printf("* ");
      }
      printf("\n");
    }
}
```

⇒

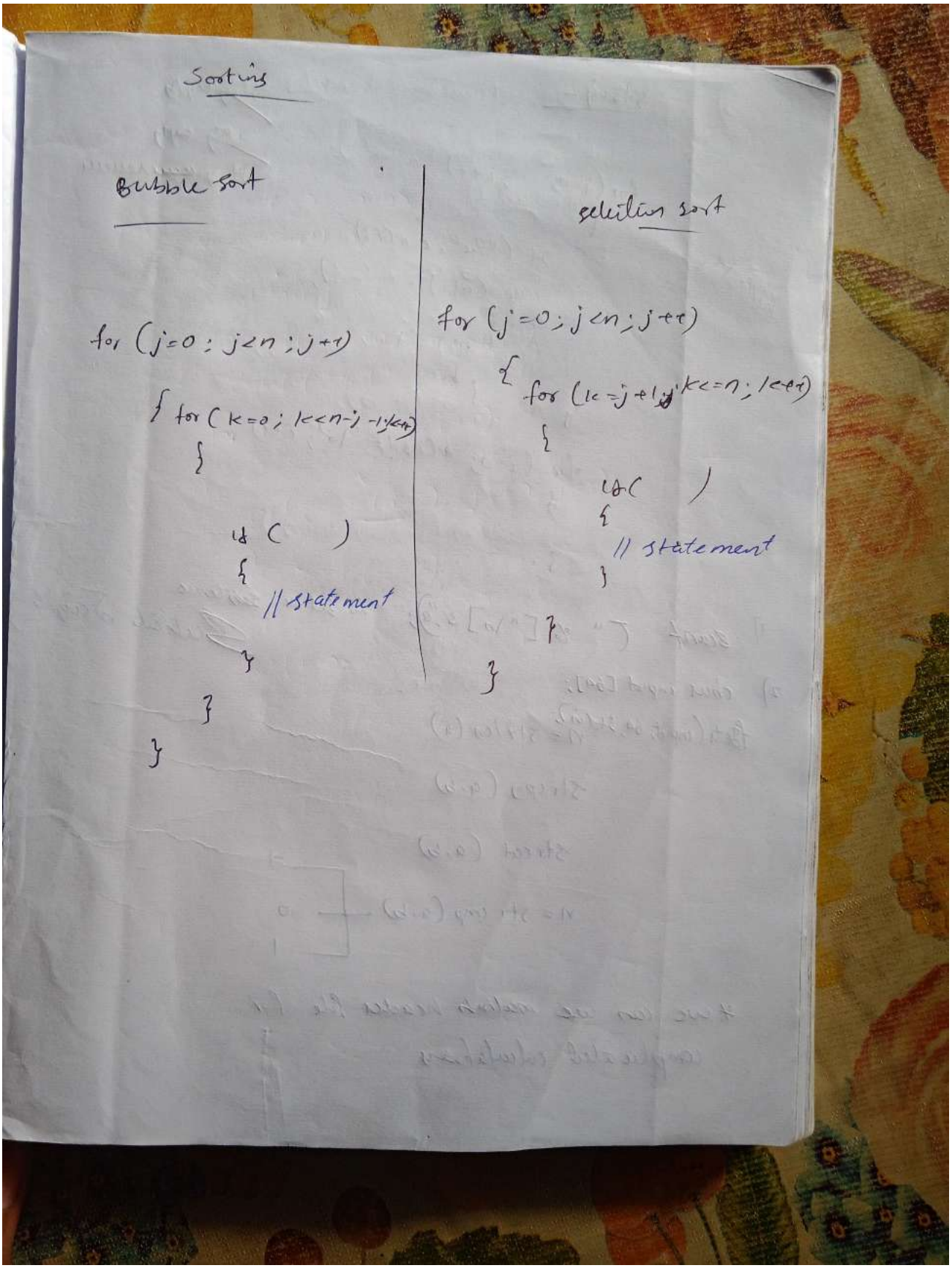
```
#include <stdio.h>
void main()
{
  int n,i,j,k;
  printf("enter a number");
  scanf("%d",&n);
 for(i=0; i<=n; i++)
 {
    for(j=2*(n-i); j>=0; j--)
    {
      printf(" ");
    }
    for(k=0; k<=2*i; k++)
    {
      printf("* ");
    }
   printf("\n");
 }

};
```

```
Switch (n)
{
    case 1:
            statement;
            break;

    case 2:
            statemat;
            break;

    Default :
            printf ("invalid entry")
}
```




```
for(i=0; i<5; i++)
{
  pt ("%d", c[i]);   // printf("%d", c[i]);
}
```

```
int c[5];
char c[5];
printf("%d", c[i]);
```



Sorting

Bubble sort

```
for (j=0; j<n; j++)
  { for (k=0; k<n-j-1; k++)
    {
      if ( )
      {
        // statement
      }
    }
}
```

selection sort

```
for (j=0; j<n; j++)
  { for (k=j+1; k<=n; k++)
    {
      if( )
      {
        // statement
      }
    }
}
```

String -

```
for(i=0;  ;i++)
{
    sf("%c", &a[i]);
    if(a[i] == '-');
    {
        a[i] = '\0';
    }
}
for(i=0; a[i] != '\0'; i++)
{
}
```

1) scanf("%[^\n]s", a); to get a sentence

2) char input[64];
fgets(input, 64, stdin);

n = strlen(a)

strcpy(a,b)

strcat(a,b)

n = strcmp(a,b) → 1, 0, -1

* we can use maths header file for complicated calculations

String copy

string reverse

hello

Suhaib Sayed

Functions

1) without argument without return type

```
Void sum ();  // fn declaration
void main ()
{
   sum ();  // fn call
}
Void sum ()   // fn definition
{
    int a,b,c;
    printf ("enter 2 nos:");
    scanf ("%d %d", &a, &b);
    c = a + b;
}
```

2). with argument without return type

```
Void sum (int , int);  // fn decl
void main ()
{
   int a,b;
   Pf ("enter 2 nos");
   sf ("%d %d", &a, &b);
   sum (a,b);   // fn call
}
Void sum (int x, int y)  // fn defn
{ int c;
  c = x+y;
Pf ("%d", c);
}
```

3) with argument with return type

```
int sum (int, int); // fn decl
void main ()
{
    int a,b,n;
    Pf ("enter 2 nos");
    sf (" %d %d", &a, &b);
    n = sum (a,b); // fn call
    pf ("%d", n);
}
int sum (int x, int y) // fn defi
{
    int c;
    c = x+y;
    return c;
}
```

4) without argument with return type

```
int sum ();
Void main ()
{
   int n;
   n= sum ();
   Pf ("%d",n);
}
int sum ()
{
  int a,b,c;
   pf ("enter 2 nos");
   sf ("%d %d", &a, &b);
   c= a+b;
   return c;
}
```

Q:- Prime or not } all 4
Q:- factorial }

```
char s[20], a[20];
int i,j,k, l=0;
for (i=0; ; i++)
{
  if (s[i] == ' ')
  {
    for (j=i-1, k=l; j>=l; j--, k++)
    {
      a[k] = s[j];
    }
    a[k] = ' ';
    l = k+1;
  }
  if (s[i] == '\0')
  {
    for (j=i-1, k=l; j>=l; j--, k++)
    {
      a[k] = s[j];
    }
    a[k] = '\0';
    break;
  }
}
```

i	j	k
0		
1		
2		
3		
4		
5	4	0
	3	1
	2	2
	1	3
	0	4
	10	6
6		
7		
8		
9		
10		
11	10	6

A
B B
C C

Recursion

```
int fact (int);
void main()
{


}
int fact (int a)
{
   int f;
   if (n==0)
    {
       return 1;
    }
   else
    {
       f=n* fact (n-1)
       return f;
    }
}
```

Pointers

datatype *name;

```
int *p;
   p=&a;
   pf ("%d", p);
   pf ("%d", *p);

int a[10], *p;
 p=a;
for (i=0; i<10; i++)
 {
     pf ("%d", (p+i));
 }
for (i=0; ; i++)
   {
     pf ("%d", *(p+i));
   }
```

2D Array

```
int a[10][10];

for (i=0; i<3; i++)
  {
      for (j=0; j<3; j++)
        {
          sf ("%d", &a[i][j]);
        }
  }
```

2D array pointer

addition

multiplication

sf → (*(p+i)+j)) pf (*(*(p+i)+j))

~~pf → (*(*(p+i)+j))~~

→ Pointer array

→ *P[];

int a, b, c, d, *P[10];

P[0] = &a;

P[1] = &b;

P[2] = &c;

P[3] = &d;

→ Function pointer

int (*P)(), a, b;

P = sum; // only function name is required

Pf ("enter 2 nos");

Sf ("%d %d", &a, &b);

s = P(a, b);

Pf ("%d", s);

Files

```
FILE *FP;
fp = fopen ("test.txt", "w");
     fprint (fp, "%s", a);
      fclose (fp);
fp = fopen ("test.txt", "r");
     fscanf (fp, "%s", a);
      fclose (fp);
```

a = getc ⇒ sf ("%c",

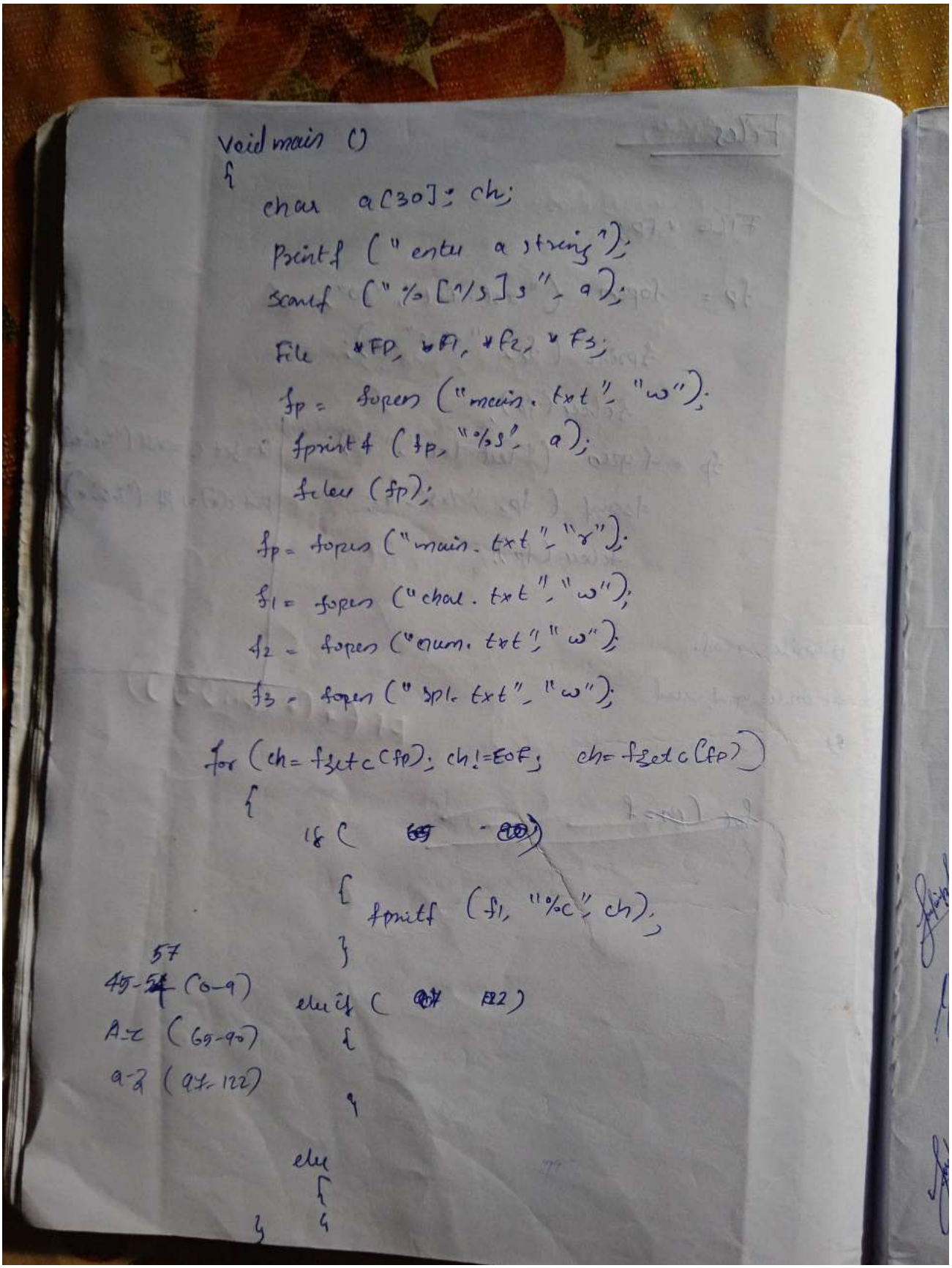
put c(a) ⇒ pf ("%c", a

1) write, read.

2) write_spiral, read

3)

~~for (i=~~ f



```
void main ()
{
    char a[30]; ch;
    Printf ("enter a string");
    scanf ("%[^/s]s", a);
    File *FP, *f1, *f2, *f3;
    fp = fopen ("main.txt", "w");
    fprintf (fp, "%s", a);
    fclose (fp);
   fp = fopen ("main.txt", "r");
   f1 = fopen ("char.txt", "w");
   f2 = fopen ("num.txt", "w");
   f3 = fopen ("spl.txt", "w");
for (ch = fgetc(fp); ch!=EOF; ch = fgetc(fp))
   {
      if (      -      )
        {  fprintf (f1, "%c", ch);
        }
57
48-57 (0-9)
A-Z (65-90)
a-z (97-122)
      else if (      122)
        {
        }
      else
        {
        }
}
```

```
fclose(f3);
fclose(f2);
fclose(f1);
fclose(fp);
```

→ Structure

```
struct student
{
    char name[20];
    int roll;
};
void main()
{
    int n,i;
    struct student s[50];
    sf("%d",&n);
    for(i=0; i<n; i++)
    {


    }

    for(i=0); i<n; i++)
    {

    }
}
```

→ Structure with pointer

```
Struct student
{ int data [illegible], *p;
};
void main ()
{
  struct students S[10]
  p=&s; S.p=&S.data;
  for (i=0; i<n; i++)
    { pf ("%d", *(p+i).data)
    }
```

if S is simple we p.data

if S[10] then we (p+i)

structure with pointer in main.

```
Struct student
{
  int data;
  char a[10];
};
void main ()
{
  struct student S, *p;
  p=&s;
  pf ("%d", p->data);
  sf ("%d", &p->data, p->a);
}
```

Structure with pointer in structure

```
#include <stdio.h>
Struct student
{
   int data[10], *p;
};
Void main ()
{
  Struct student s;
   s.p = s.data;      //&s.data if data is not an array
  int i,n;
 Pf(" enter the limit");
 sf ("%d", &n);
 Pf (" enter the elements");
 for (i=0; i<n; i++)
 {
    sf ("%d", (s.p+i));
 }
for (i=0; i<n; i++)
 { Pf ("%d", *(s.p+i));
 }
```

* Structure with pointer in main

```
#include <stdio.h>
Struct student
{
  int data;
};
void main ()
{
  Struct student s[10], *p;
  int n,i;
  p=s;
Pf("enter the limit");
Sf("%d", &n);
Pf("enter the elements");
for(i=0; i<n; i++);
{
Sf("%d", &(p+i)->data);
}
for (i=0; i<n; i++)
{
  Pf("%d", (p+i)->data);
}
```

* Structure in Structure

```
struct address
{
    int house_no;
};

struct employee
{
    char name [20]
    int ID;
    struct address a;
};
void main ()
{
    struct employee e;
    pf ("%s", e.name);
    pf ("%d", e.ID);
    pf ("%d", e.a.house_no);
}
```

* calling struct to a function

```
Struct employee
{
  char name [20];
  int ID;
};
Void main ()
{  Struct employee e;
   sf ("%s", e.name);
   sf ("%d", &e.ID);
   Print (e);
}
Void print ( Struct employee a)
{
   pf ("%s", a.name);
   pf ("%d", a.ID);
```

DMA — Dynamic Memory Allocation

1) malloc
2) calloc
3) Realloc

malloc

```
pointer = (data type *) malloc (n * size of (datatype));
p = (int *) malloc (10 * size of (int));

for(i=0; i<10; i++)
{
printf ("%d", *(p+i));
}
```

calloc

```
pointer = (datatype *) calloc (n, size of (datatype));
a = (int *) calloc (10, size of (int));
```

Realloc

```
p = (int *) realloc (p, 10 * size of (int));
```

Linked List

```
#include <stdio.h>
Struct node
 {
    int data, *next;
 }
Struct node *head = NULL, *new, *temp;
 int count = 0;
 Void add();
 void print();
 Void main()
 {
   int i,n;
   char c='y';
   while(c=='y')
   {
      Pf ("press 1. add 2. print");
      Sf ("%d", &n);
       Switch (n)
       { case 1:
                add();
                count++;
                break;
         case 2:
                print();
                break;
         default:
                Pf ("invalid entry");
       }
```

```
        Pf ("press 'y' to continue");
        sf ("%c", &c);
        }
}

Void add()
{
new = (struct node *) malloc (1* size of (struct node));
Pf ("enter data");
sf ("%d", &new -> data);
    if (head == NULL)
      {
        head = new;
        head -> next = NULL;
      }
    else
    {
      for (temp = head; temp -> next != NULL; temp = temp -> next);
        temp -> next = new;
          new -> next = NULL;
    }
}

Void print()
{
    for (temp = head; temp != NULL; temp = temp -> next)
    { Pf ("%d", temp -> data);
    }
```

Pre processor directives

macros

#include

#define

#if

#else

#endif

#if def

#if endif

#include

operator.h

```
void sum (int a, int b)
{
  pf ("%d", a+b);
}
#include <stdio.h>
#include "operator.h"
void main ()
{ int x,y;
  pf ("enter 2 nos");
  sf ("%d %d", &x, &y);
  sum (x,y);
```

define PI 3.14

area = 2 * PI * r;

③
```
# define int char
void main ()
{
  int i = 65;
  Pf ("%d", size of (i));
}
```

④
```
# define FALSE -1
# define TRUE 1
# define NULL 0

void main ()
{
  if (NULL)
  {
    puts ("NULL");
  }
  else if (TRUE)
  {
    puts ("FALSE");
  }
  else
  {
    puts ("TRUE");
  }
```

o/p → False
because if condition works only in positive values.

①
```
# define cube(x) (x*x*x)
void main ()
{
  int b=3, n;
  n = cube(b++);
  Pf = ("%d %d", n, b);
}
```

out = 60, 6

3 → b++ ∨ 4 → b++ * 5 → b++ → 6

②
```
# define TRUE 0
void main ()
{
  while (TRUE)
  { Pf ("hello");
  }
}
```

No o/p because value of TRUE is zero while works only on non-zero condition

#if (condition)
statement

#else
statement

#endif

no parenthesis with statement, semicolon

#endif

#ifdef (FALSE)
Pf ("hai");

#ifndef (FALSE)
Pf ("hello");

o/p hai

Tuenary operator (condition)? (expression 1):(expression 2)

~~a=(a>b)? Pf("a is good","b is good");~~ } if true expression 1 is shown else expression 2 is shown

```
#include <stdio.h>
Void main()
{
  int a,b;
  Printf("enter the numbers\n");
  Scanf("%d %d",&a,&b);
(a>b)? Pf("a is higher"):Pf("b is higher");
}
```

UNION

```
#include <stdio.h>
#include <std lib.h>
union student
 { int roll;
   char a;
 };
void main()
 {
   int n;
   union student u;
   n = size of (u);
   printf("%d", n);
 }
```

Here the output will be 4 because is UNION the memory allocated will be the highest memory used by the variables.

In this case int needs 4 byte and char needs 1 byte so the memory allocated will be 4 bytes.

Storage class :

Type	Initial value	memory allocation	Life	scope
auto	Garbage	Stack	block	block
Extern	0	memory segment	whole Pgm	whole Pgm
Static	0	memory segment	whole Pgm	block
register	Garbage	register	block	block

Stack downward memory allocation. (Last in First out)

heap upward " " . (First in First out)

Qualifiers

1) Signed Qualifiers:

1. signed ("%d")
2. unsigned ("%u")

2) Size Qualifiers:

1. short (2byte) // in case of controller ("%d").
2. Long (4byte) ("%ld").
3. Long Long (8byte) ("%lld").
4. double (8byte) ("%lf").

3) Volatile:

In some case of loops the compiler will done some optimization in its own risk. So it may remove some intezers. In order to avoid that we use 'volatile' key word.

```
int loop=1;
while (loop==1)
{
}
loop= a+b;
```

In normal case the compiler will remove the loop and make the while loop as 'while (1)' an infinite loop. So if we use the term loop again the code will be error.

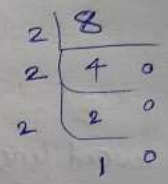
if we use volatile along with loop we can avoid this.

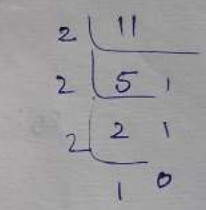
1) Constant:

Keyword const use to make a value fixed throughout the prgm. Any changes to that value will be an error.

Bitwise Operators

&	and
|	or
^	xor
<<	left shift
>>	Right shift
~	not.





① even or odd
② Count binary / parity check.
③ swaping without 3rd variable

PIC (Peripheral Interface Controller)

eve eee PIC 16F876A

eee have 12→ Base line (word length 12 bit)
16→ Mid range (" 14 bit)
17/18 High end. (" 16 bit)

word length →

Op code + operand = ascue length

Eg:- MOV A,B
op code operand

An MCU can be divided into 3

core	peripherals	special features
		mid range
ALU	Timer (0,1,2)	BOR (Brown out Reset)
instructions	GPIO	POR (Power on Reset)
memory mapping	ADC	WDT (watch Dog Timer)
Serial Oscillator	CCP	
Reset	MSSP	sleep (
	USART	

GPIO → General purpose I/O Registers
capture compare PWM (CCP)
USART (universal synchronize Asynchronize Receiver Transmitter)

Pwm (Appl:)

1) Frequency Variation

2) voltage variation

Architecture

Von-neumann (Prgm memory cpu all in one block)

Harvard (Prgm memory - cpu - Data memory) separate block.

For PIC we use Harvard architecture

* PIC 16F 876A have 14 bit 8k prgm memory
* 8 bit RISC ALU
* 28 Pin IC

PIC 16 F 876 A

family Name — 16 Ram — F Flash memory — 10bit ADC, EEPROM, USART, 28 pin

GPIO

3 Ports

Port A =>	6 bit	TRIS A	To set data direction
Port B =>	8 bit	TRIS B	
Port C =>	8 bit	TRIS C	

Blink Prgm

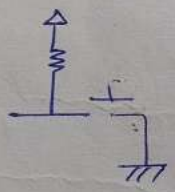
```
#include <XC.h>
#include <stdio.h>
#define _XTAL_FREQ 4000000
void main()
{
    TRISB = 0x00;
    PORTB = 0x00;
    while (1);
    {
      PORTB = 0xFF;
      __delay_ms(1000);
      PORTB = 0x00;
      __delay_ms(1000);
    }
```

~~PK2CMD~~ PK2cmd -P PIC16F876A -X -M -F

```
TRISB = 0x00;
PORTB = 0x00;
while (1)
{
    for (i=0; i<8; i++)
    {
        PORTB = 1<<i;
        __delay_ms (1000);
    }
}
```

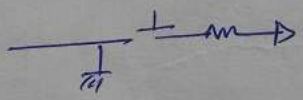
Switch configuration

1) Pull up



when switch is free current flows to controller, if switch is pressed power flows to ground

2) Pull down



here we have no guarantee that there will be a current setting the controller so it is called.

Switch2

```
for (i=0; ; i++)
{
    PortC = 1<<i;
    __delay_ms(500);
    if (RB0==0)
    {
        while (RB0==0);
        for (j=i-1; ; j--)
        {
            PORTC = 1<<j;
            __delay_ms(500);
            if (RB0==0)
            {
                while (RB0==0);
                i=j;
                break;
            }
            if (j==0)
            {
                j=8;
            }
        }
    if (i==7)
    {
        i=-1;
    }
}
```

decoder

```
char a;
a = (~ PORTB) & 0x07)        // PORTC = 1<<(~PORTB & 0x07)
PORTC = 1<<a;
```

encoder

```
for (i=a; i != 0; c=i>>1)
  {
    count ++;
  }
PORTC = count;
count = 0;
```

USART

Register Configuration

```
TXSTA = 0x24;
RCSTA = 0x90;
```

SP BRG → Baud Rate Generator

Transmission

```
Void main ()
{
    TxSTA = 0x24;
    RCSTA = 0x90;
    SPBRG = 25;
    while (1)
    {
        TxREG = 'a';
        while (TRMT==0);
        -- delay_ms (300);
    }
}
```

```
char a[10];
while (1)
{ for (i=0; a[i]!='\0'; i++)
  {
    TxREG = a[i];
    while (TRMT==0);
    -- delay_ms (500);
  }
}
```

Reception

```
Void main ()
{
  TxSTA = 0x24;
  RCSTA = 0x90;
  SPBRG = 25;
  char a;
  while (1)
  {
    while (RCIF==0);
    a = RCREG;
    TXREG = a;
    while (TRMT==0);
  }
}
```

```
for (i=0; ; i++)
{
  while (RCIF==0);
  a[i] = RCREG
  if (a[i]=='\r')
  {
    a[i]='\0';
    break;
  }
}
```

ADC conversion

```
int x;
char a[20];
ADCON 0 = 0x41;
ADCON 1 = 0x80;
TXSTA = 0x24;
RCSTA = 0x90;
SPBRG = 25;
while(1)
{ CHS2=0; CHS1=0; CHS0=0;
  GO=1;
  while (GO==1);   || while (ADIF==0);
                   ADIF=0;
  x = (ADRESH<<8) | ADRESL;
  sprint f (a, "ADC= %d", x);
  tx(a);
  --delay_ms (500);
}
```

In General case we can't use PORT A for switches because they are defaulted to analog values. In order to make them as switch configure ADCON1 = 0x07 or 0x06 during the set control bit configuration.

Timer

- used to make the time accurate
- there 3 timers
  - timer 0
  - timer 1
  - timer 2

1 machine cycle equals the time taken to complete an instruction set

(instruction set includes
- fetch
- decode
- execute
- storing.

155

$$T = \frac{1}{f_{ox}} = \frac{1}{4 \times 10^6} = 0.25\ \mu s$$

Timer value : me

1 : 8

$1\ mc = 0.25 \times 4 = 1\ \mu s$

$250 \times 8\ \mu s = 2000\ \mu s$

storing.

timer value varies at 8 machine cycles. we set the timer value upto 250 so when the time value becomes 250 it will be 2000 μs. time contains this 500 times we set 1 second delay.

```
void main ()
{
    TRISB = 0x00;
    OPTION_REG = 0x00;
    PORTB = 0x00;
    int i;
    while (1)
    {
        PORTB ^= 0xFF;
        for (i=0; i<2000; i++)
        {
            while (TMR0 <= 250);
        }
    }
}
```

use 0010 0001

TICON = 0x21

T2CON = 0x05

we use option-REG for Timer 0 we have T1CON & T2CON for Timer1 & Timer 2 respectively

~~The last 3 bits~~ in OPTION-REG ~~indicate the ratio b/w Timer value & machine cycle~~

OPTION-REG last 3 values for prescale value

T1CON 5-4 for prescale value

T1CON : 1-0 Prescale 6-3 postscale

Digital clk

```
int Rx()
{
 char x[10]; int x,y;
  for(i=0; ;i++)
  {
   while (RI==0);
   x[i]= SBUF;
    address = x[i];
    while (ran==0);
     if (x[i] == '\r')
     { x[i]='\0';
       break;
     }
  }
  y= atoi (x);
   return y;
}

tx (a);
hr= Rx();
tx (b);
min = Rx ();
tx (c);
sec = Rx ();
```

Timer Interrupt

Timer 1 $2^{16} = 65536$

```
For (i=0; i<1000; i++)
{
  while (TMR0IF==0);
    TMR0=5;
}
```

Timer ~~0~~ 1 Interrupt flag.

Timer 1 can store data ~~Largest of~~ In some case we need the time datas for calculation. then we Timer 1.

we can limit the final value of the Timer in Timer 2 using keyword → PR2 at the configuration step.

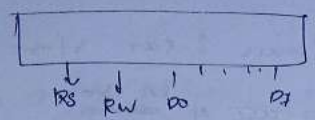
Eg:- PR2 = 250;

the final value for the register will be 250.

65536 / 250 → 65286

65286 / 250 → 65536

LCD



RS ⇒ For command mode & datamode

RS0 → command mode

RS1 → data mode.

RW → Read & write

D0-D7 → Digital pins

0x02 = returns home

0x28 = 4 bit mode

0x38 = 8 bit mode

0x01 = LCD clear

0x80 = 1st row 1st column

0xC0 = 2nd row 2nd colum

0x0C = Display on cursor off

PORTC for LCD

RC0 = RS

RC1 = RW

RC2 = E

RC4 : D4

: = :

RC7 D7.

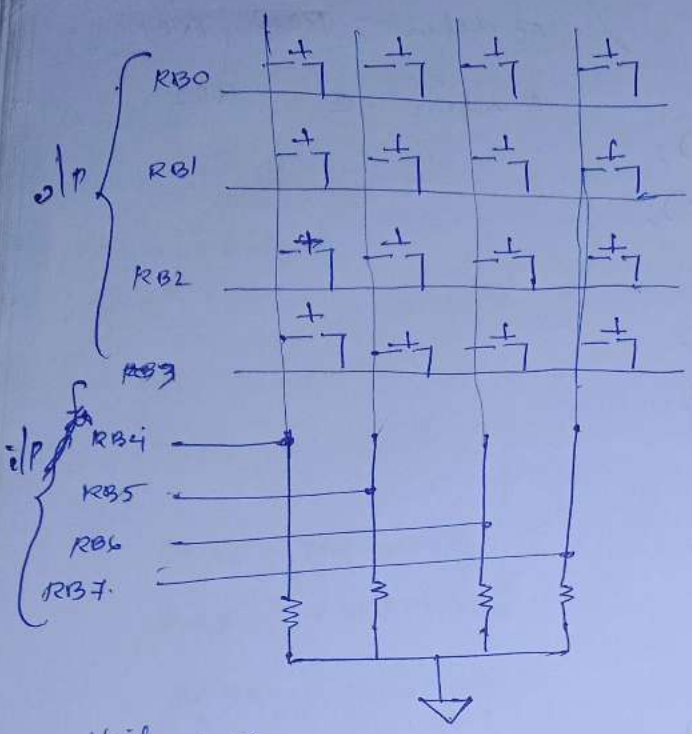
```
Void main ()
{
    TRISC = 0x00;
    command (0x02);
    Command (0x0C);
    commnd (0x28);
    Commnd (0x01);
    Commnd (0x80);
    while(1)
    {
      data('a');
    }
}
void command (char a)
{
  char b;
  RS=0;
  RW=0;
  b= a & 0xF0;        0x0F;
  PORTC = PORTC & 0x00;
  PORTC = PORTC & 0x0F;
  PORTC = PORTC | b;
  E=1;
  __delay_ms (25);
  E=0;
  __delay_ms (25);
```

```
#define RS RB0
#define RW RC1
#define E  RC2
```

```
b= a & 0x0F;
PORTC = PORTC & 0x0F;
PORTC = PORTC | b << 4;
E=1;
__delay_ms (25);
E=0;
```

if RS=1 then it will be data

Keypad




```
Voidmain()
{
 TRISB = 0xF0;
 TRISC = 0x00;
 init(); //LCD command
  char a;
 while(1)
 {
  a = keypad();
   data(a);
 }
}
```

```
char keypad ()
{ while (1)
  {
  __delay_ms (100);
  RB0 = 0;
  RB1 = RB2 = RB3 = 1;
  if (RB4 == 0)
    { while (RB4 == 0);
      return '7';
    }
```

continue under the while loop until all conditions were completed.

RB0	7	8	9	/
RB1	4	5	6	*
RB2	1	2	3	-
RB3	C	0	=	+

command

1) duty selection

2) pic16f876a
pk2cmd -pPIC16F876 -x -m -f
balance of folder name.

```
}
```

SPI, I2C

PWM

$$PWM_{period} = ((PR2)+1) \times 4 \times T_{osc} \times TMR_2 \text{ prescale value}$$

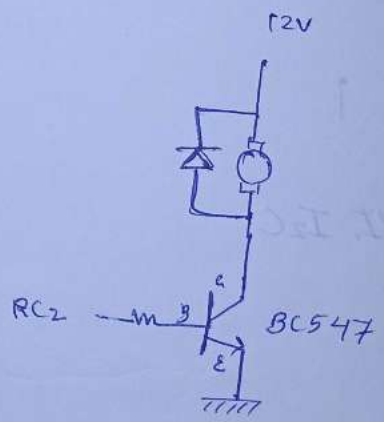
Prescale 1:4

PR2 = 249

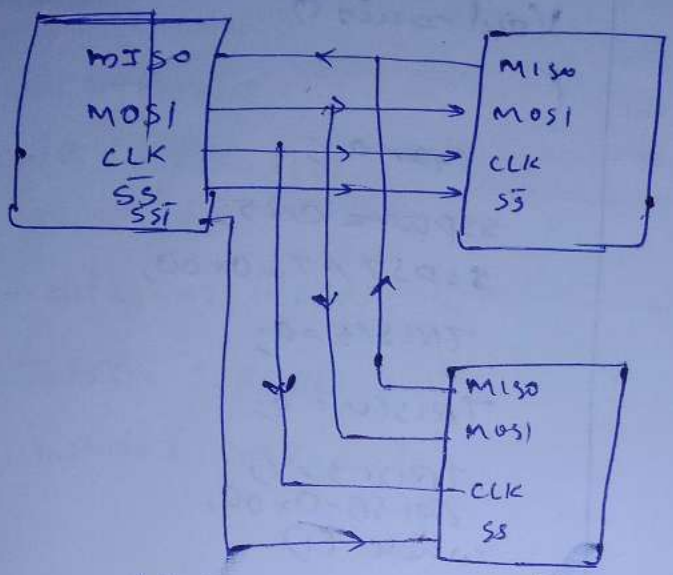
$F_{osc} = 4 \times 10^6$ MHz $(249+1) \times 4 \times 0.25 \times 4$

$= 1000.45$

```
Void main()
{
TRISC2 = 0;
T2CON = 0x04
CCP1CON = 0x0F;
PR2 = 249;   int i;
while (1)
  {
    for(i = 0; i<=255; i++)
    {
      CCPR1L=i;
    }  --delay_ms(20);
```



SPI



one of the

* Fastest communication (50MBPS)
* wire length maximum upto 10m.
* No use of complex addressing mode.
* Cannot check parity and can't find the data stream is completed.
* No: Pins required is high.
* USART → Full Duplex.
* SPI / I2C → half Duplex.

SDO = RC5
SDI = RC4
SCK = RC3
SS = RA5

Master

```
Void main ()
{
  char a = 0x01;
  SSPCON = 0x20;
  SSPSTAT = 0x00;
  TRISC5 = 0;
  TRISC4 = 1;
  TRISC3 = 0;
  TRISB = 0x00;
while (1)
{
  PORTB = ~a;
  SSPBUF = a;
while (SSP1F == 0);
  a = a<<1;
  if (a==0)
  {
    a = 0x01;
  }
  --delay_ms (500);
}
}
```

Slave

```
Void main ()
{
  char a;
  SSPCON = 0x25;
  SSPSTAT = 0x00;
  TRISC5 = 0;
  TRISC4 = 1;
  TRISC3 = 1;
  TRISB = 0x00;
while (1)
{
while (SSP1F == 0);
  a = SSPBUF;
  PORTB = a;
}
}
```

Slave to master

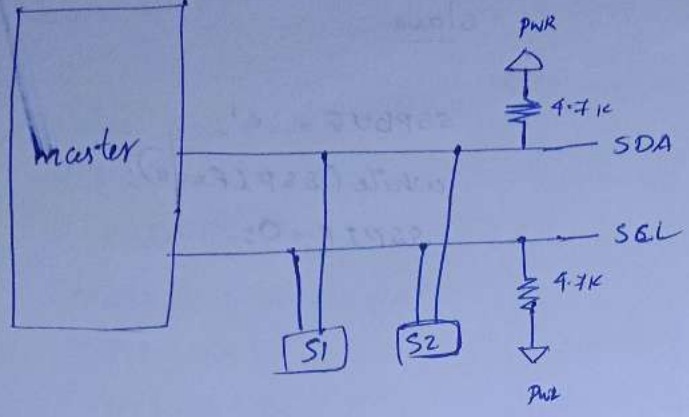
Master

```
SSPBUF='A';
while (SSPIF==0);
  SSPIF= 0;
--delay-ms (200);
 TxREG= SSPBUF;
  while (TRMT==0);
```

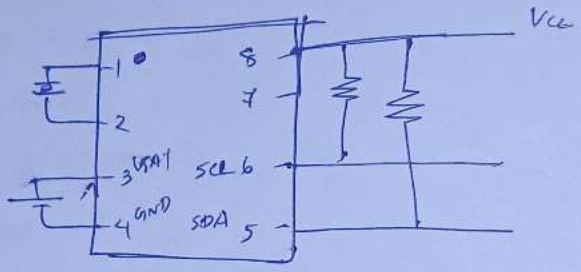
slave

```
SSPBUF = 'C';
while (SSPIF==0);
 SSPIF = 0;
```

I2C

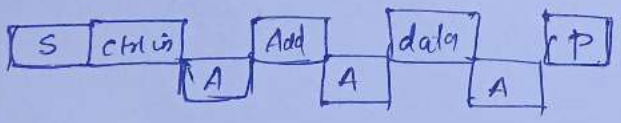


DS1307



64 byte — 8 byte time of data,

56 byte NV SRAM



⇒ write mode.

To start the data transmission the SDA must be low and the SCL must be high. It is based on controller.

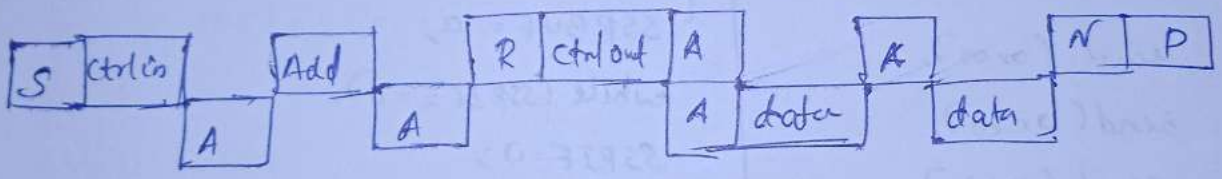
Ctrl in → D0 → Read
D1 → write
} b 51307

P → SDA high
scl high

A → recipient device the bus low.

Add → slave address.

Read mode.



```
void main()
{
  SSPCON = 0x28;
  SSPSTAT = 0x80;
  SSPADD = 9;
  TXSTA = 0x24;
  RCSTA = 0x90;
  SPBRG = 25;

  TRISC4 = TRISC3 = 1;
  Start();
while (send(0xD0))
 { start(); }

 send (0x00);
 send(0x20);
 send (0x34);
 send (0x02);

 Stop();
while()
{
  Start();
while (send(0xD0))
 { start();}
 send (0x00);
 restart();
 send (0xD1);
 hr = read;
 ack;
 min = read;
 ack();
 hr = read();
 nack();
 stop();
 }
}

Void start()
 {
  SEN = 1;
  while (SSPIF==0);
  SSPIF = 0;
 }
int send (char a)
{ SSPBUF = a;
 while (SSPIF==0);
 SSPIF=0;
 return (ACKSTAT);
 }
Void stop()
  {
    PEN = 1;
    while (SSPIF==0);
      SSPIF = 0;
    }
Void restart()
  {
    RSEN = 1;
    while (SSPIF==0);
    SSPIF = 0;
  }
```

```
char read ()
{
  RCEN=1;
 while (SSPIF==0);
   SSPIF=0;
  return (SSPBUF);
}
Void ack ()
{
  ACKDT = 0;
  ACKEN = 1;
  while (SSPIF==0);
   SSPIF=0;
}
Void nack()
{
  ACKDT=1;
  ACKEN=1;
 while (SSPIF==0);
  SSPIF=0;
}
```

PDF Created Using

Camera Scanner

Easily Scan documents & Generate PDF

https://play.google.com/store/apps/details?id=photo.pdf.maker

LPC 2148 — ARM7 TDMI-S

PORT 0

32 pins

0-31

PORT 1

16 pins

16-31

PORT 0 · 0-15 ⇒ PINSEL0

16-31 ⇒ PINSEL1

PORT 1 16-31 ⇒ PINSEL2

IODIR = Data Direction (1 o/p 0 i/p)

IOPIN = Status of a Pin (1 on 0 o/f)

IOSET = 1 ON

IOCLR = 1 off

```
#include <LPC21xx.h>
main ()
{
  PINSEL0 = 0x00000000;
  PINSEL1 = 0x00000000;
  PINSEL2 = 0x00000000;

  IODIR0 = 0xFFFFFFFF;
  IODIR1 = 0xFFFFFFFF;
  while (1)
   {
     IOSET0 = 0xFFFFFFFF;
     for(i=0; i<50000; i++);

     IOCLR = 0x FFFF FFFF;
     for (i=0; i<50000; i++);

   }
}
```

```
IODIR1 = 0xFFFFFFFF;

IOPIN0 = 0x00000000;
```

```
IODIR0 = 1<<21;
IOSET0 = 1<<21;
IOCLR0 = (1<<21);
```

Single LED blink;

```
if ((IOPIN0&(1<<25)) ==0)
{
  while (1)
  {
  }
}
```

UART

Fosc = 12 MHz

CCLK = M * Fosc

= 60 MHz

	PLL(EN)	PLL(CON)
MSEL	5	1
PSEL	2	1

$PCLK = \frac{CCLK}{4} = \frac{60MHz}{4} = 15MHz$

$U_0DLL = \frac{PCLK}{16 * Baud\ rate}$

$= \frac{15 \times 10^6}{16 \times 9600} = 97.$

if PLL'n disabled U.DLL is 19

```
main()
{
PINSEL0 = (1<<0) | (1<<2);
U0LCR = 0x83;
U0DLL = 97;
U0LCR = 0x03;
while(1)
 {
   U0THR = 'a';
   while ((U0LSR & (1<<6))==0);
 }
}
```

Receive

```
while ((U0LSR & (1<<0))==0)
a = U0RBR;
```

ADC

```
main ()
{
  int x;
  char a[5];
  PINSEL0 = (1<<0) | (1<<2) | (1<<10) | (1<<11);
  U0LCR = 0x83;
  U0DLL = 97;
  U0LCR = 0x03;
  AD0CR = 0x00200380;
  while (1)
  {
    AD0CR |= 1<<24;
    while ((AD0GDR & (1<<31)) == 0);
    x = (AD0GDR >> 6) & (0x3FF);
    sprintf (a, "Adc0 = %d", x);

    Lcd(a);
  }
}
```

Multiband
vary at the
8 bits of ADC

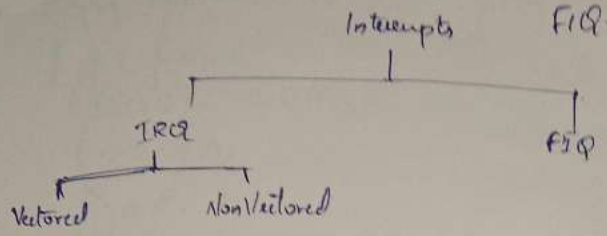
Timer

```
main ()
{
    PINSEL0 = 0x00000000;
    PINSEL1 = 0x00000000;

    IODIR 0 = 0xFFFFFFFF;
    IOPIN 0 = 0x00000000;

    T0PC=0;
    T0TC = 0;

    T0PR= 14999;
    T0MR0 = 1000;

    T0MCR = 2;

    T0TCR = 2;
    T0TCR = 1;

    while (1)
    {
      while ( T0TC != T0MR0);

        T0TC = 0;

        IO0IN0 = 0x FFFFFFFF;

      }
}
```

~~tpc21isp~~ Timer.hex /dev/ttyUSB0 38400 12000.

Interrupts

IRQ- Interrupt Request Queue.
FIQ- Fast Interrupt Query.



External Interrupt 0

```
main()
{
 PINSEL0 = (1<<2) | (2<<3);
 PINSEL2 = 0x 0000 0000;

 IODIR1 = 0xFFFFFFFF;

 IOPIN1 = 0x 00 00 00 00;

VICINTEnable = (1<<14);

VICVECT vect Addr = (unsigned) intr;

EXTMODE=1;
EXTPOLAR=1;
 while(1)
 {

 }
}
```

```
__irq void intr()
{
 if (EXTINT==1)
 {
  IOPIN1 ^= 0xFFFFFFFF;
  EXTINT=1;
 }
 VICVect Addr = 0;
}
```

(21:3) (6:7) 14, 15 2, 2.

Uart Interrupt

```
main ()
{
  PINSEL0 = (1<<0) | (1<<2);
  U0LCR = 0x83;
  U0DLL = 97;
  U0LCR = 0x03;
  U0IER = 1;
  VICIntSelect = 0;
  VICIntEnable = (1<<6);
  VICDefVectAddr = (unsigned) Uart;
  while (1);
}
__irq void uart()
{
  a = U0RBR;
  U0THR = a;
  while((U0LSR & (1<<6)) == 0);
  VICVectAddr = 0;
}
```

if((U0IIR & 4) ==

Timer Interrupt

```
__irq void timer ()
{
  if (T0IR==1)
  {
    IOPIN0 = 0xFFFFFFFF;
    T0TC = 0;
    T0IR = 1;
  }
  VICVectAddr = 0;
}
```

```
__irq void timer 0
{
  if (T0IR==1)
  {
    IOPIN0 = 0x FF FF FF FF;
    T0TC = 0;
    T0IR = 1;
  }
  VICVectAddr = 0;
}
```

```
main()
{
  PINSEL0 = 0x 00000000;
  PINSEL1 = 0x 00 00 00 00;
  IODIR0 = 0xFF FF FF FF;
  IOPIN0 = 0x00 00 00 00;
  T0PC = 0;
  T0TC = 0;
  T0PR = 14999;
  T0MR0 = 1000;
  T0MCR = 1;
  T0TCR = 2;
  T0TCR = 1;
  VICIntSelect = 0;
  VICIntEnable = (1<<4);
  VICDefVectAddr = (unsigned) timer;
  while (1);
}
```

Vectored Interrupt

```
main ()
{
PINSEL0 = (1<<0) | (1<<2);
PINSEL2 = 0x00000000;
IODIR1 = 0xFFFFFFFF;
U0LCR = 0x83;
U0DLL = 97;
U0LCR = 0x03; U0IER=1;
IOPIN1 = 0x00000000;
VICIntSelect = 0;
VICIntEnable = (1<<6)|(1<<4);
VICVectCntl0= 0x26;
VICVectAddr0 = (unsigned) uart;

VICVectCntl1 = 0x24;
VICVectAddr1 = (unsigned) timer;

T0PR = 14999;
T0MR0 = 1000;
T0TC = 0;
T0MCR = 1;
T0TCR = 2;
T0TCR = 1;
while (1);
}
```

```
__irq void uart ()
{
a = U0RBR;
U0THR = a;
while ((U0LSR & (1<<6)) == 0);
VICVectAddr = 0;
}

__irq void timer ()
{
IOPIN1 = 0xffffffff;
T0IR = 1;
T0TC = 0;
VICVectAddr = 0;
}
```

LPC 1768 ARM CORTEX — UM10360

```
main()
{
    LPC_SC -> CLKSRCSEL = 0x00;      // } configurations
    LPS_SC -> PLL0CON = 0x00;        //
                                     LPC____
                                     structure name

    LPC_PINCON -> PINSEL0 = 0x00000000;
    LPC_PINCON -> PINSEL1 = 0x00000000;

    LPC_GPIO0 -> FIODIR = 0x FFFFFFFF
    LPC_GPIO0 -> FIOPIN = 0x00000000;
    while (1)
    {
        LPC_GPIO0 -> FIOPIN ^= 0xFFFFFFFF;
    }
}.
```

using switch

```
if ((LPC_GPIO1 -> FIOPIN & (1<<0)) == 0)
{
    while ((LPC_GPIO1 -> FIOPIN & (1<<0)) == 0);
    {
```

Timer

PCLKSEL0 = (1<<

```
main ()
{
  LPC_SC -> CLKSRCSEL = 0x00;
  LPC_SC -> PLL0CON = 0x00;
  LPC_SC -> PCONP = 1<<1;
  LPC_SC -> PCLKSEL0 = (1<<2);

  LPC_PINCON -> PINSEL0 = 0x00000000;
  LPC_PINCON -> PINSEL1 = 0x00000000;

  LPC_GPIO0 -> FIODIR = 0xFFFFFFFF;
  LPC_GPIO0 -> FIOPIN = 0x00000000;

  LPC_TIM0 -> PC = 0;
  LPC_TIM0 -> TC = 0;
  LPC_TIM0 -> PR = 3999;
  LPC_TIM0 -> MR0 = 1000;
  LPC_TIM0 -> MCR = 2;
  LPC_TIM0 -> TCR = 2;
  LPC_TIM0 -> TCR = 1;
while (1)
{
  while (LPC_TIM0 -> TC != LPC_TIM0 -> MR0);
  LPC_TIM0 -> TC = 0;
  LPC_GPIO0 -> FIOPIN ^= 0xFFFFFFFF;
}
```

308 UART

```
main()  // Transmission
{
 int i;
 char a[20] = "qwerty";
 LPC_SC → CLKSRCSEL = 0x00;
 LPC_SC → PLL0CON = 0x00;
 LPC_SC → PCONP = 1<<3;
 LPC_SC → PCLKSEL = 0 1<<6;
 LPC_PINCON → PINSEL0 = ~~0x04~~ (1<<4) | (1<<6);
 LPC_UART0 → LCR = 0x83;
 LPC_UART0 → DLL = 26;
 LPC_UART0 → LCR = 0x03;
 while (1)
  {for (i=0; a[i]; i++)
 { LPC_UART0 → THR = a[i];
  while ((LPC_UART0 → LSR (1<<6)) == 0);
   }
  }
}
// Receiver
  while (1)
   { for (i=0; ; i++)
    {
     while ((LPC_UART0 → LSR (1<<0)) == 0);
     a[i] = LPC_UART0 → ~~LSR~~ RBR ;
     if (a[i] == '\r')
    } { a[i] = '\0'; break;}
   }
```

1<<4 | 1<<6
1<<3 PCONP
(7:6) PCLKSEL
~~U0FCR = 1<<0~~
U0DLL = 26

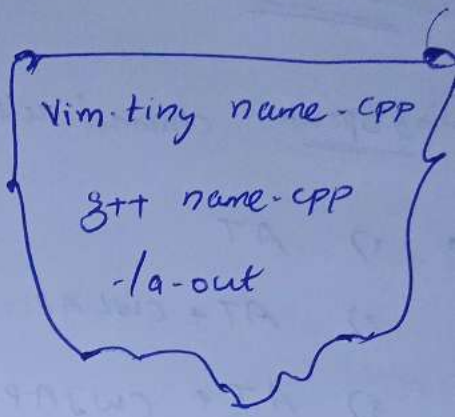
ADC

* 12 Bit ADC Converter
* Results on the 4-15 bits of ADGDR register
* Channel selection at 0→7 bits of ADCR register.
* Start bit 24 of ADCR
* Done bit 31 of ADGDR.

```
main()
{
 int i;
 char a[16];
LPC_SC → CLKSRCSEL = 0x00;
LPC_SC → PLL0CON = 0x00;
LPC_SC → PCONP = (1<<3) | (1<<12);
LPC_SC → PCLKSEL0 = (1<<6) | (1<<24);
LPC_PINCON → PINSEL0 = (1<<4) | (1<<6) | (1<<9);
LPC_UART0 → LCR = 0x83;
LPC_UART0 → DLL = 26;
LPC_UART0 → LCR = 0x03;
LPC_ADC → ADCR = 0x00200001;
while(1)
 {
  LPC_ADC → ADCR | = 1<<24;
  while((LPC_ADC → ADGDR & (1<<31)) == 0);
  i = (LPC_ADC → ADGDR >> 4) & (0x00000FFF);
  sprintf(a, "ADC0 = %d", i);
  tx(a);
 }
}
```

CPP

```
#include <iostream.h>
using namespace std;

int main()
{
    cout << "hello";
}
```



Datatypes

under primitive class

1) int
2) float
3) char
4) string
5) boolean.

IOT

ESP2866

ThingSpeak channel selection

value send

1) AT
2) AT+CWLAP
3) AT+CWJAP="name","password"
4) AT+CIPSTART="TCP","api.thingspeak.com",
5) GET /link address/ port.
6) AT+CIPSEND=( ) (length of 5th command)

Put all this commands in defined function.

Data send.

ATMEGA 128

7 ports (A - G)

In G only 5 ports

16MHz. oscillator

DDRx → Data Direction (1 o/p, 0 i/p)

PORTx → Data Register

Pin Xn → Pin select Specific Data.

```
#include <avr/io.h>
#include <util/delay.h>
void main()
{
  DDRA= 0xFF;
  while (1)
  {
    PORTA= 0xFF;
    _delay_ms (500);
    PORTA = 0x00;
    _delay_ms (500);
  }
}
```

Switch

```
DDRB = ~(1<<0);
if ((PINB&(1<<0))==0)
  while ((PINB & (1<<0))==0);
```

UART

Transmitter

```
Void main ()
{
  UCSR0A = 0;
  UCSR0B = 0x18;
  UCSR0C = 0x06;
  UBRR0L = 103;   // we use 8 bit
                  so least bit register is selected

  while (1)
  {
   UDR0 = 'a';
   while ((UCSR0A & (1<<5)) == 0);
  }
}
```

Receiver

```
Void main ()
{
  UCSR0A = 0;
  UCSR0B = 0x18;
  UCSR0C = 0x06;
  UBRR0L = 103;
  char a;
  while (1)
  {
  while ((UCSR0A & (1<<7)) == 0);
   a = UDR0;
}
}
```

ADC

10 bit conversion Prescalar factor (128)
ADC needs clock slow (100–200 KHz)

```
Void main ()
{
 int x;
 char a[20];

 ADCSRA = 0x87;
 ADMUX = 0x00;   // channel selection.

  UCSR0A = 0;
  UCSR0B = 0x18;
  UCSR0C = 0x06;
   UBRR0L = 103;

 while(1)
  {
    ADCSRA |= 1<<6;
    while((ADCSRA&(1<<4))==0);
    X = ADCL|(ADCH<<8);
    sprintf(a,"ADC = %d", x);
    tx(a);
  _delay_ms(500);
  }
}
```

RTOS

blinq

```
#includes < includes.h >
  OS_STK  S1[128];

Void blinq ()
{
 while (1)
  {
    PORTA = 0xFF;
    OSTimeDlyHMSM(0,0,0,500);
    PORTA = 0x00;
    OSTimeDlyHMSM(0,0,0,500);
  }
}

Void main ()
{
  DDRA = 0xFF;

  TIMSK = 0x01;
  TCCR0 = 0x07;        } imp time configuration

  OSInit();
  OSTaskCreate(blinq, 0xFF, &S1[127], 5);

  OSStart();
}
```

blinq, 0xFF, &S1[127], 5

- blinq → function name
- 0xFF → non zero value
- &S1[127] → address of local variable count from 0-127 or 0-255 (depending on requirement)
- 5 → priority

```
#include <includes.h>
OS_STK s1[128];
OS_STK s2[128];
Void bling()
{
 while(1)
 {
  PORTA = 0xFF;
  OSTimeDlyHMSM(0,0,0,500);
  PORTA = 0x00;
  OSTimeDlyHMSM(0,0,0,500);
  OSTaskResume(5);
 }
}
Void uart()
 {
  char b[20]= "qweety";
  int i;
  while(1)
  {
   for(i=0;b[i];b++)
   {
    UDR0= b[i];
    while((UCSR0A&(1<<5))==0);
   }
  OSTimeDlyHMSM(0,0,0,500);
  OSTaskResume(6);
  OSTaskSuspend(5);
 }
}
```

```
Void main()
 {
 DDRA = 0xFF;
UCSR0A = 0;
UCSR0B = 0x18;
UCSR0C = 0x06;
UBRR0L = 103;
TIMSK = 0x01;
TCCR0 = 0x07;
 OSInit();
OSTaskCreate(bling,0xff,&s1[127],6);
OSTaskCreate(uart,0x8f,&s2[127],5);
 Os Start();
}
```

Semephore Mutex, Mail box, Queue

	Semaphore	Mutex
Approach	Signalling Mechanism	Locking mechanism
Type	Its an integer	its an object
Functioning (important)	Multiple prgm threads can access a finite instance of resources	Allows multiple prgm threads to access a single resource bit with a restriction of not accessing simultaneously
Access Permission	can be accessed by any process	the process which has acquired the mutex lock can only release it
Types	binary, counting	No types
	wait until the resource count is greater	waits until the lock is released.

mail box

```
OS_STK S1[128] S2[128], S3[128];
OS_EVENT *mailData1, *mailData2;
int *err;
Void adc1();
Void adc2();
int main()
{
  Vaut initialiser l'ADC, configurer la
  TCCR0 = 0x07;
  TIMSK = 0x01;
  OSInit();
  OSTaskCreate(adc1, 0xFF, &S1[127], 4);
  OSTaskCreate(adc2, 0xFF, &S2[127], 5);
  OSTaskCreate(Vaut, 0xFF, &S3[127], 6);
  mailData1 = OSMboxCreate(1);
  mailData2 = OSMboxCreate(1);
  OSStart();
}
Void adc1();
{ while(1)
  { // adc code;
    OSMboxPost(mailData1, b);
    OSMTimeDlyHMSM(0,0,1,0);
  }
}
```

```
Void ade2().
{ while (1)
  {// ade 2 code;
   OSMboxPost(mailData2, d);
   OSTimeDly(HZ/10);
  }
}

Void vat()
{ char *out1, *out2; int i;
  while (1)
    {
      out1 = OSMboxPend(mailData1, 0, &err);
        tx();
      out2 = OSMboxPend(mailData2, 0, &err);
    } tx();
}
```

Embedded Linux & Linux Drives

ls.

cd Desktop

cd /sys.

ls

cd class.

cd gpio

echo 17 > export

cd gpio17.

echo out > direction.

echo 1 > value

echo 0 > value

LED blink

```
# include <stdio.h>
# include < unistd.h>
# include < stdlib.h >
void main ()
{
 int i;
 FILE * fp;
 fp= fopen ("/sys/class/gpio/export", "w");
 fprintf (fp, "17");
 fclose(fp);
fp= fopen ("/sys/class/gpio/gpio17/direction", "w");
 fprintf (fp,"out");
 fclose (fp);
while(1)
 {
 fp= fopen ("/sys/class/gpio/gpio17/value", "w");
 fprintf (fp, "1");
 fclose (fp);
 sleep(1);
 fp= fopen("/sys/class/gpio/gpio17/value", "w");
 fprintf ( fp,"0");
 fclose (fp);
 }
}
```

Loadable Kernel Module

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE ("GPL");
MODULE_AUTHOR ("SUHAIB_SYD");
MODULE_DESCRIPTION ("A Simple Loadable Kernel Module");
MODULE_VERSION ("0.1");
static __init int hello__init (void)
 {
 Printk (KERN_INFO " module installed successfully\n");
  return (0);
 }
static void hello_exit(void)
{
 printk( KERN_INFO " Module removed successfully\n");
}
module_init (hello__init);
module_exit (hello__exit);
```

Steps (do in administrator account)

① make a makefile with name Makefile and type (obj-m=hello.o)

② sudo su, lsmod

③ enter into the directory

④ make -C /lib/modules/$(uname -r)/build M=$PWD modules

⑤ open new terminal and type

tail -f /var/log/kern.log

⑥ in kernel space terminal type (console window)

insmod filename.ko

modinfo filename.ko

⑦ type rmmod filename.ko (module removed)

Commands

sudo su → for administrative privilege

lsmod → list the modules

make → kernal compilation,

tail →

insmod → for installing module

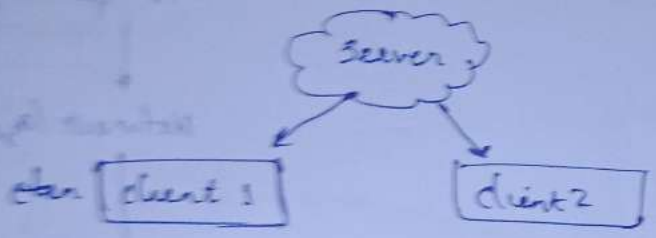
mod info → for getting the information of modules

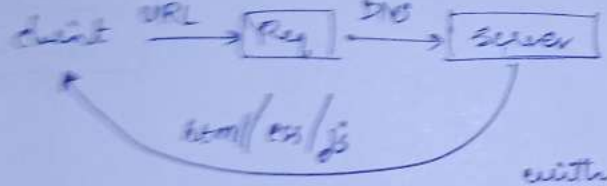
rmmod → for removing module

Networking

LAN
MAN
WAN

→ Client - Server Model





DNS → Domain Name s/m.

DNS search through server with the help of IP address.

Rendering through DOM (Document Object Model).

Network topology

1) Star
2) Mesh
3) Ring
4) Bus

OSI (Open s/m Interconnection)

application
↓
Presentation
↓
session
↓
Transport layer
↓
Network layer
↓
Data link layer
↓
Physical layer

TCP/IP

application
↓
Transport layer
↓
Network layer
↓
Network Interface

IP address

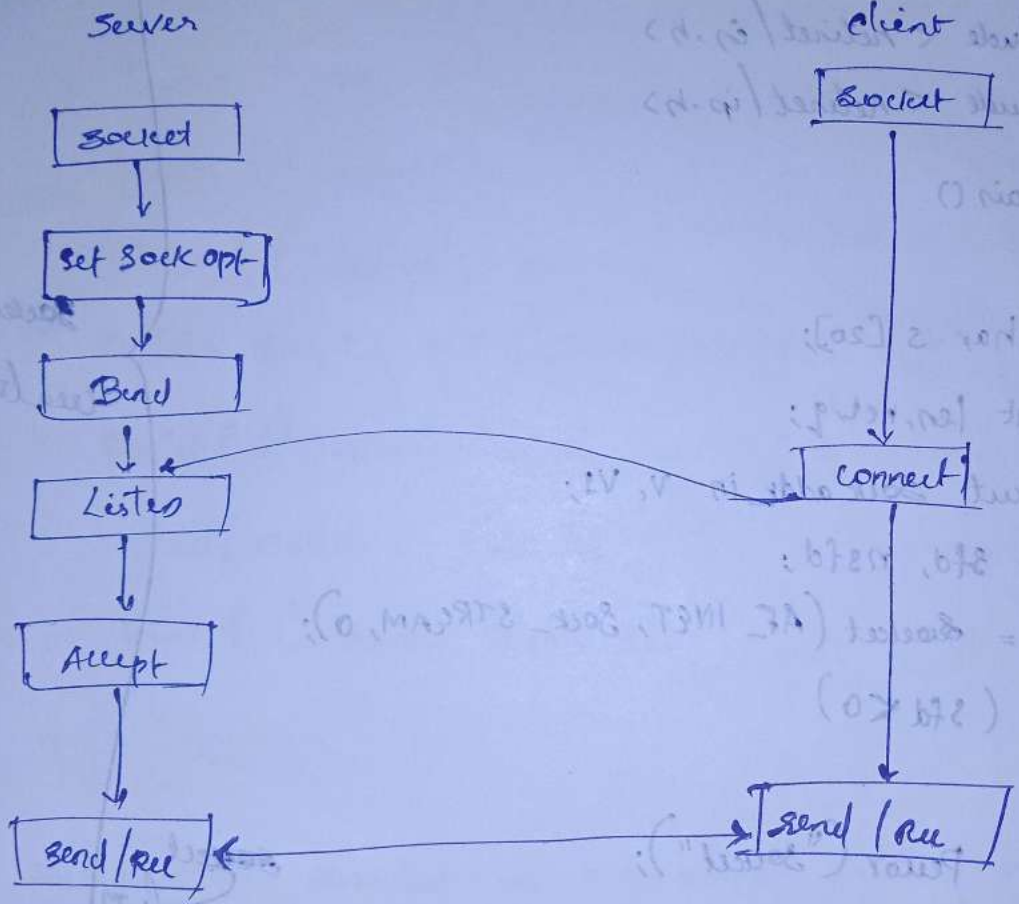
* address of a network
* MAC address is the address of a s/m on a network.

IPV4 IPv4 → 32 (numeric)
IPv6 → 128 (alpha numeric)

address Resolution protocol

Neighbour Discovery is used in case of IPv6.

Socket Programming



Socket Programming

Server side

```
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdio.h>
#include <netinet/in.h>
#include <netinet/ip.h>

Void main()
{
    char s[20];
    int len, ret, q;
    Struct sockaddr_in V, V1;
    int sfd, nsfd;
    sfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sfd < 0)
    {
        Perror("socket");
        return;
    }
    Perror("socket");
    Printf("sfd=%od\n", sfd);
//////////////////////////////////////////
    V.sin_family = AF_INET;
    V.sin_port = htons(4000);
    V.sin_addr.s_addr = inet_addr("0.0.0.0");
```

socket creation

socket creation

```
len = size of (V);
ret = bind (sfd, (struct sockaddr *)&V, len);
Perror ("bind");
```

Binding

```
listen (sfd, 5);
perror ("listen");
Printf ("before\n");
nsfd = accept ( sfd, (struct sockaddr *) &V1, &len);
Printf ("after\n");
read(nsfd, s, sizeof (s));
Printf ("data = %s\n", s);
}
```

Listen

sockaddr_in structure is a prebuild structure on the header file <sys/socket.h> so we don't want to define or declare it

socket function contains 3 parameter

AF_INET → IPV4	SOCK_STREAM → TCP	0 → IP (only 1 value)
AF_INET6 → IPV6	SOCK_DGRAM → UDP	

htons → used to give port number.

Socket programming client side

```
void main()
{
    header file and socket creation is same as the previous
    code.

    V.sin_family = AF_INET;
    V.sin_port = htons (9000);
    V.sin_addr.s_addr = inet_addr("127.0.0.1");
    len = size of (v);
    ret = connect (std, (struct sockaddr *) &v, len);
    if(ret < 0)
    {
        Perror ("connect");
        return;
    }
    Perror("connect");

    Printf("enter the data\n");
    scanf("%[^\n]s", s);
    write (std, s, strlen(s)+1);
}
```

Python

```
a=10
Print(a)
```

o/p → 10

```
a,b = 10,15
Print("a= b=" %(a,b))
```

o/p ⇒ a=10 b=15

```
Print("a= ", a)
```

o/p ⇒ a= 10

```
if __name__ == '__main__': main()
a,b = 10,15
Print("a=%d b=%d" %(a,b))
```

```
a=int(input("enter a number"))
```

Arithematic

+ − * / % // ~ floor division

(/ → normal value, % → remainder, // → Quotient)

Logical

** expond + unary +ve − unary -ve

and or not

membership

in not in

identity

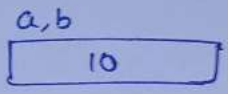
is is not

To finish virtual mode enter " exit() "

memory allocation

a = 10

b = 10



- if we assign multiple identifiers with same value no new memory is allocated only the name tag is changed.

conditional statement

1) if elif else

2) Statement 1 if ( ) else statement 2

```
if ( ):
    # statement
else :
    # statement
```

loop

* while (i<5):
    Print (i)
    i+=1

* for i in range (5)
    Print ("%d\n" %i)

- for i in range (0,5,2):
    print ("%d\n")

(0, 5, 2)

Starting index, final index, ~~step~~. offset (~~comma~~ sep)

Str = 'hello'

Str[0] = 'h'

Str[1:5] = 'ello'

Str[1:] = 'ello'

Str1 = "world"

Str = Str + Str1 = 'hello world

Str * 3 = 'helloworld helloworld hello world'.

format

x = 42

Y = 73

Print ('the number is {}'.
format (x,y))

or

print (f'the number is {x}')

→ join():

a, b = 'abc', '123'

c = a.join(b)

'1abc 2abc 3'

→ split():

a = "1x2x3x4"

~~b = a.split('x')~~ # ['1', '2', '3', '4']

b = a.split('x')

→ len():

str1 = "world"

len(str1) # 5

→ max():

min() :

- the first element of list.

Task:

String length

compare two strings

concatenate two strings

String copy

find characters numbers, special characters in a string.

→ a = 'hello' ;

a[::-1] = 'olleh'

→ a = "hello"

a.capitalize # Hello.

a.find (e) # 1 (return index value)

~~a.replace(e,a)~~ #

a.replace(o, a) # hella

List — can modify

```
list = ['abcd', '123', 2.23,]
tiny list = ['123', 'John']

Print (list)
Print (list[0]) → 1st element
Print (list[1:3]) → starting from 1st to ending in 3rd index
Print (list * 2) → 2 times printing
Print (list + tiny list) → concatenated list

l = [1, 2, 3]
l.append(1) # [1, 2, 3, 1]
l.count(1) # 2 one time checked the occurence of '1' in the list
l.insert(index, value)
    l.insert(0, 55) # [55, 1, 2, 3, 1]
l.remove(1) # [55, 2, 3, 1] # removes '1' from the first position
del (l[1]) # [55, 3, 1]
```

Keyword argument

```
def sum(a,b):
    print(a+b)
sum(a=10, b=20)
```

even though we changed the order of argument passing the value will be passed to the function correctly.

Default argument

```
def mul(a,b=1):
    return(a*b)
n = mul(2)
print(n)
```

n=2 a=2, b=1

```
n = mul(2,2)
print(n)
```

n=4 a=2, b=2

Variable length argument

```
def mul(a,*b):
    return(a*b)
n = mul(2,3,4,5,6)
print(n)
```

b is a tuple

then the 2 will be stored to 'a' and other values will be stored to 'b' as a tuple.

lambda function

```
Sum = lambda a,b,c : a+b+c
```

Similar to the macros in C

Higher order functions.

```
def first(a,b):
    def second(c):
        return (a+b+c)
    return(second)
add = first(1,2)
X = add(3)
```

X=6.

recursive functions

```
def fact(n):
    if n==1:
        return n
    else:
        f= n* fact(n-1)
```

Variable Length Argument

```
def see
    def see (*b):
        b= list (b)
        n= len(b)
        for i in range (0,n):
            Print (b[i])
    l= []
    a= input("enter the commands\n")
    l= a.split(',')
    see(l).
```

Python modules

1) import math
   ~~os.factorial (5)~~ math.factorial(5)

2) import math as abc
   abc. factorial (5)

3) from math import *
   factorial (5)

4) from math import factorial
   factorial (5)

5) from math import factorial as abc
abc(5)

~~For loop~~

Loops

While

```
def main():
    x = 0
```

1) # define a while loop

```
while (x<5):
    print(x)
    x = x+1
```

2)
define a for loop

```
for x in range(5,10):
    print(x)
```

3)
use a for loop over a collection

```
days = ["mon", "tue", "wed", "thur", "fri"]
for d in days:
    print(d)
```

4)
use break nd continue

```
for x in range(5,10)
    * if (x==7): break
    * if (x%2==0): continue
        print(x)
```

5)
using the enumerate() fn to get index

```
days = ["mon", "tue", "wed", "thur", "fri"]
for i,d in enumerate(days):
    print(i,d)
```

ole
1) 0, 1, 2, 3, 4
2) 5, 6, 7, 8, 9
3) mon, tue, wed....
4) * 5,6
   * 5, 7, 9
5) 0 mon
   1 tue
   2) wed
   4) thu
   5) fri

Class

Syn # class class name():

```
class myclass():                                  → main class
    def method1(self):                            Self is necessary on all class creators it should be the first argument of functions.
        print("my class method1")
    def method2(self, some string):
        print("my class method 2:" + some string)

class another class(my class):                    → derived class
    def method2(self):
        print("another class method 2")
    def method1(self):
        my class.method1(self)
        print("another class method1")

def main():
    c = my class()                                → Defining object
    c.method1()
    c.method2("This is a string")
    c2 = another class()
    c2.method1()
```

by using _ in front of a variable in a class we can't set it private. to access we have to use method

Eg:- self.type()

→ date, time & date time class

all are classes.

```
from datetime import date
     "    "    time
     "    "    datetime
```

→ Calendar

```
import calendar
```

→ Files

```
def main():
    f = open("text file.txt", "w+")      → file open and create
    f = open("textfile.txt", "a")        → append data
    f.close                              → closing the file
    f = open("textfile.txt", "r")
    if f.mode == 'r':
        contents = f.read()
        print(contents)
```

```
for i in range(10):
    f.write("This is line\n")
(adding data to file)
```

To read line by line

```
if f.mode == 'r':
    fl = f.readlines()
    for x in fl:
        print(x)
```

* type()

→ help to find the datatype of any variable

String type

```
a = 8
b = 9
x = f'seven{a}_{b}'
Print (`x is {}. format (x)}
Paint (type (x)).
```

o/p ⇒ x is seven 8 9
<class 'str'>

Numeric type

1) Float 2) integer

```
from decimal import *
a = Decimal ('.10')
b = Decimal ('.30')
x = a+a+a-b
print ('x is {} . formed (x)}
print (type (x))
```

o/p → x is 0.00

without importing Decimal this would work accurately

Boolean type

Values greater than zero will considered as True values
Zero, None, empty string all are then considered as false

sequence type

Eg: tuple and list, dictionaries
list can varied tuple is fixed, dictionaries can vary.
sequencing starts with zero index.

type(), id()

* type() function is like class it represents the class
* id() functions are like objects it help to set the unique number for each object

Ternary operators

hungry = True
x = 'Feed the bear now it is hungry else' do not feed'
Print (x)

statement if Condition else statement 2

Boolean operators

and, or, not, in, not in, is, is not

Generators & Decorators

Sets

* Set allows no duplicate
* if we can sort the values in set.
* we can done all set operation in the data type

Constructing an object
1) Default
2) parameter

class Animal:

def __init__(self, type, name, sound):

these arguments are only available after creating the objects

Class method

handling exceptions

```
def main():
    try:
        statement
    except error type:
        statement
```

built in functions

Numeric

Eg: int, float, abs, divmod (both quotient), complex

String functions

Eg: repr, str, ascii, chr, ord

containers

Eg: len(), reversed(), sum(), max(), min(), any(), zip()

Objects related functions

Eg:- type(), isinstance(), id()

Python OOPS

```
class Book:                          # creating the class
    def __init__(self, title):       # initializing fnctn to create properties of object
        self.title = title

b1 = Book("Beauty and world")        # creating object
b2 = Book("a world not peace")

print(b1)                            # prints class and property
print(b1.title)
```

Instance methods and attributes

- '_' is used to privatise the attributes
- attributes are identical as member variable
- type() can be used to identify the class of instance
- is instance() can be used for checking whether the instance is a part of the mentioned class.
- Instance attributes and method are part of the instance when it is created

* instance methods work on specific objects and class methods works on entire class

* Now we discuss static methods - they don't modify ~~class~~ class or instance method.

class Book:

```
                                    # create class method
    @classmethod
    def set booktype(cls):
        statement

    @staticmethod                   # create static method
    def set book list():
        statement

    def set price(self):            # create instance methods
        statement
```

inheritance:

Eg: check on pc folder.

abstract base class.

Eg: check on folder

→ using multiple inheritance

Python uses method resolution order when we call an attribute ~~in the order mentioned~~

```
class A():
    statement
class B():
    statement
class C(A,B):
```

→ multiple inheritance

when we call an attribute by C which is common in A and B then the method resolution comes in action - Python will check in C first then go to A and finally to B. If it is found it will stop there and will not check further

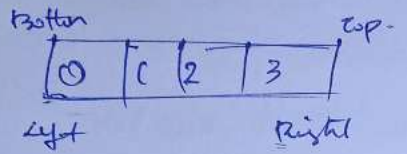
* It is not used widely because of its complexity

Data structures in Python

* concrete implementations of Abstract Data Type (ADT) that organize and retrieve data stored in memory

Stack

* can only add or remove at the top
* follows last in first out



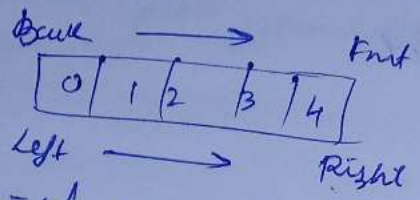
* It has limited access only at the ends
* They are recursive data structures.

```
class stack:
    def __init__ (self):
        self.items = []
imp def push(self, item):
        pass
imp def pop (self):
        pass.
    def peek (self):
        pass
    def size (self):
        pass
    def is_empty (self):
        pass
```

General syntax

1) push ⇒ append i/p to stack and return
2) pop ⇒ remove from stack and return the last added i/p which is at the top
3) peek ⇒ return the last i/p which is at the top
4) size ⇒ it checks the size of the stack using len()
5) is_empty ⇒ return a Boolean value based on wheather the stack is empty or not

* An ADT that stores items in order in which they were added
* items in a queue are added to the back and removed from front
* follows FIFO



* Limited access only from front

Basic Syntax

```
class Queue:
    def __init__(self):
        self.items=[]
    def enqueue(self,item):
        pass
    def dequeue(self):
        pass
    def peek(self):
        pass
    def size(self):
        pass
    def is_empty(self):
        pass
```

* enqueue: insert elements to the nth position (zero)
self.items.insert(0,item)
* dequeue: remove elements from the nth position

peek, size and is_empty are same as in the stack.
here peek shows the first added item or the nth item
self.items[-1]

____ => Double ended Queue

* An ADT that resembles both stack and a queue
* items in a deque can be added to and removed from both the back and front (both FIFO and LIFO)

Basic syntax

```
class Deque:
    def __init__(self):
        self.items = []
    def add_front(self, item):
        pass
    def add_rear(self, item):
        pass
    def remove_front(self):
        pass
    def remove_rear(self):
        pass
    def peek_front(self):
        pass
    def peek_rear(self):
        pass
    def size(self):
        pass
    def is_empty(self):
        pass
```

=> self.items.insert(0, item)

=> self.items.append(item)

=> self.items.pop(0)

=> self.items.pop()

=> self.items[0]

=> self.items[-1]

=> return len(self.items)

=> return self.items == []

return a boolean value by comparing our deque with an empty deque

Linked List

Single Linked List

```
class SLLNode:
    def __init__(self, data):
        self.data = data
        self.next = None

    def set_data(self):
        return self.data   # return self.data attribute

    def set_data(self, new_data):
        self.data = new_data   # replacing the current
                               value of self.data attribute
                               with new_data parameter

    def set_next(self):
        return self.next   # return the self.next attribute

    def set_next(self, new_next):
        self.next = new_next   # replacing the current value
                               of self.next attribute with
                               new_next parameter
```

Double Linked List

```
class DLLNode:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.previous = None.

    def get_data(self):
        return self.data

    def set_data(self, new_data):
        self.data = new_data

    def get_next(self):
        return self.next

    def set_next(self, new_next):
        self.next = new_next

    def get_previous(self):
        return self.previous

    def set_previous(self, new_previous):
        self.previous = new_previous
```

creating single linked list and its method

single linked list code in the previous page
+

```
class SLL:
    def __init__(self):
        self.head = None

    def __repr__(self):
        return ("SLL object: head = {}".format(self.head))

    def is_empty(self):  # return true if the linked list is empty otherwise false
        return self.head is None

    def add_front(self, new_data):  # add a node whose data is the new_data argument to the front of linked list
        temp = SLLNode(new_data)
        temp.set_next(self.head)
        self.head = temp

    def size(self):  # Traverses the linked list and returns an integer value representing the number of nodes in the linked list
        size = 0
        if self.head is None:
            return 0
        current = self.head
        while current is not None:
            size += 1
            current = current.get_next()
        return size
```

```
def search(self, data):
    if self.head is None:
        return "Linked list is empty".
    current = self.head
    while current is not None:
        if current.get_data() == data:
            return True
        else:
            current = current.get_next()
    return False
```

Traverse the linked list and return true if data searched for is present in one of the nodes, otherwise it returns false.

⇒ Removing a node from single linked list

```
def remove(self, data):
    if self.head is None:
        return "Linked list is empty"
    current = self.head
    previous = None
    found = false
    while not found:
        if current.get_data() == data
            found = true
        else:
            if current.get_next() == None:
                return "A node with that value is not present"
            else:
                previous = current
                current = current.get_next()
```

same line as while loop

```
if previous is ~~None~~None:
    self.head = current.get_next()
else:
    previous.set_next(current.get_next())
```

Double linked list and its methods

Double linked list code :

```
class DLL:
    def __init__(self):
        self.head = None
    def is_empty(self):
        return self.head is None
    def size(self):
        size = 0
        if self.head is None:
            return 0
        current = self.head
        while current is not None:
            size += 1
            current = current.get_next()
        return size

    def search(self, data):
        if self.head is None:
            return "Linked list is empty"
        current = self.head
        while current is not None:
            if current.get_data() == data:
                return True
            else:
                current = current.get_next()
        return False
```

```
def add_front(self, new_data):
    temp = DLLNode(new_data)
    temp.set_next(self.head)
    if self.head is not None:
        self.head.set_previous(temp)
    self.head = temp

def remove(self, data):
    if self.head is None:
        return "Linked list is empty"
    current = self.head
    found = False
    while not found:
        if current.get_data() == data:
            found = True
        else:
            if current.get_next() is None:
                return "Node with that data is not present"
            else:
                current = current.get_next()
    if current.previous is None:
        self.head = current.get_next()
    else:
        current.previous.set_next(current.get_next())
        current.next.set_previous(current.get_previous())
```

Python Dictionaries

```
sal_info = dict()
print(sal_info)
sal_info = {'A': 1011, 'B': 1010, 'C': 1000, 'D': 1001}
print(sal_info)
if ('D' in sal_info):
    print(sal_info['D'])
else:
    print("not found")
for location in sal_info:
    print(sal_info[location])
for location in sal_info:
    print location
```

o/p

```
# {}
# print whole data
# print 1001
# print all values
# print all keys
```

⇒ Basic operations and methods

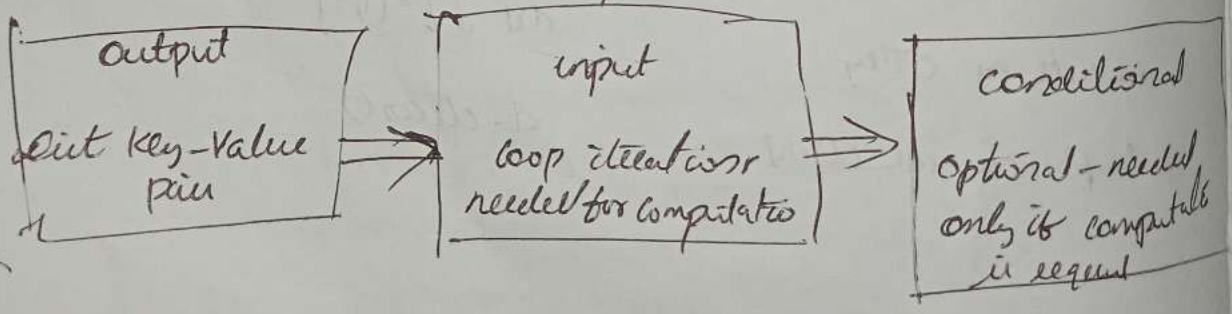
- change an entry — `d["11"] = "two"`
- Delete an entry — `del d["IV"]`
- Remove all items — `d.clear()`

Python dictionary Basic Methods

* Access a value from the dict	d.get(key, default msg)
* List all keys	d.keys()
* List all values	d.values()
* List both keys and values	d.items()
* maximum value in dict	d.max()
* minimum "	d.min()
* Return the value associated with key	d.pop(key, default)
* Access and return a random key-value pair	d.popitem()
* Return a sorted list	d.sorted()
* make a shallow copy of dict	d.copy()

~~using python lists within a dictionary~~

Python dictionary : comprehension



Sorting

1) Bubble sort
2) merge sort
3) quick sort

} example codes on pc folder

Decorators

A decorator is a callable that takes another fnctn as an argument, extending the behavior of that function without explicitly modifying that function

```
def my_decorator(func):
    def wrapper():
        return 'F-I-B-O-N-A-C-C-I'
    return wrapper

@ my_decorator
def pfib():
    return 'Fibonacci'

print(pfib)
```

Decorators with *args and **kwargs.

```
from functools import wraps

def decorator(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        # do something
        result = func(*args, **kwargs)
        # do something
        return result
    return wrapper

@decorator
def func():
    pass
```

Class and decorator in Python

```
from functools import update_wrapper

class count:
    def __init__(self, func):
        update_wrapper(self, func)
        self.func = func
        self.cnt = 0

    def __call__(self, *arg, **kwarg):
        self.cnt += 1
        print(f'Current count: {self.cnt}')
```

```
        result = self.func(*args, **kwargs)
        return result.
```

```
@count
def fib(n):
    if n<2:
        return n
    else:
        return fib(n-1) + fib(n-2)

fib(4)
```

all function

```
all(
    condition(item)
    for item in iterable
)
```

syntax

~~Gives a true if all condition are satisfied iterable are~~

if all of the items are true or true worthy

for loop

~~Gives~~ a true if all items are satisfied the condition

any function

```
any(
    condition(item)
    for item in iterable
)
```

if any of the items are true or true worthy

enumerate function

```
for item in enumerate(list, starting index)
    print(item)
```

zip function

Syntax : zip(*iterators)

Parameter :
python iterables or containers (list, string)

Return value :
returns a single iterator object, having mapped values from all the containers

Sorted function

Sorted(iterable, *, key=None, reverse=False)

Variable argument list in functions

def addition(*numbers) Synta

Eg: def addition(base, *args):
    result=0
    for arg in args
        result += arg
    return result

```
def main():
    print(addition(5,10,15,20))   # pass different argument
    print(addition(1,2,3))
    myNums = [5,10,15,20]         # pass an existing list
    print(addition(*myNums))

if __name__=="__main__":
    main()
```

Lambda functions

```
def celsius_to_farenheit(temp):
    return (temp * 9/5) + 32
def farenheit_to_celcius(temp):
    return (temp-32) * 5/9
def main():
    cTemps = [10, 12, 34, 100]
    fTemps = [32, 65, 100, 212]

    print(list(map(lambda t: (t-32)*5/9, fTemps)))
    print(list(map(lambda t: (t*9/5)+32, cTemps)))

if __name__=="__main__":
    main()
```

C++

```
#include <iostream>
using namespace std;
void main ()
{
    string name;
    cin >> name
    cout << name << "hello" << endl;
}
```

i/p subscribe

o/p => subscribe hello

datatype

```
<datatype> <variable name>;
```

int, float, char, string, bool

array

```
string playername [2] = { "tom", "jerry" }
```

if statement

```
if (condition is true)
{
    // statement;
}
else
{
    // statement;
}
```

conditional operator

<, >, ==, <=, >=, !=

Switch case

```
Switch (n)
{
    case 1:
        Statement;
        break;
    case 2: Statement
        break;
    :
    :
    case n:
        Statement;
        break;
    default:
        Statement
        break;
}
```

loops

```
for(declaration; condition; iteration)
{
    Statement
}
```

```
int m=0;
while(m<5)
{
    Statement
    m++;
}
```

```
int m=0;
do
{
    Statement
    m++;
} while (m<5);
```

```
int array={1,2,3,4};
for( int i : array)
{
    statement
}
```

#exclusive for c++

Functions

< Return Data Type > Function Name (parameter list)

{

function body code

}

By using 'static' along with a data type inside a function, this will help to avoid removing the current value when the function called again. ⇕

Pass by reference value

```
Void swapNums (int &x, int &y)
{
    int z = x;
    x = y;
    y = z;
}

int main ()
{
    int a=10, b=20;
    Cout << " Before swap: " << \n";
    cout << a << b << "\n";
    swapNums (a, b);
    Cout << "after swap" << "\n";
    cout << a << b << "\n";
    return 0;
```

Pass by value

```
Void foo (int y)
{
  std:: cout << "y=" << y << "\n";
}
int main ()
{
  foo(5); // 1st call
  int x{6};
  foo (x); // 2nd call
  foo (x+1); // 3rd call
  return 0;
}
```

o/p y=5
y=6
y=7

⇕

Otherwise the value will be initialised again

Files

Read mode

```
# include <iostream>
# include <fstream>
  using namespace std;

void main()
{
    ifstream instream;
    instream.open("pangram.txt");
    string pangram;
    if (!instream.fail())
    {
        while(instream >> pangram)
        {
            cout << "the pangram is:" << pangram << endl;
        }
    }
    instream.close()
}
```

write mode

creating details from one file to another

```
# include <iostream>
# include <fstream>
  using namespace std;
void main()
{
      ifstream in stream;
      instream.open("pauphrase.txt");      // existing file
      ofstream out stream;
      out stream.open("Newpauphrase.txt");
      int user Guess = 0;
      string pau phrase;
while(!instream.fail())
{
  while (instream >> pauphrase)
  {
    cout << "the pauphrase is :" << pau phrase << endl;
    cout << "what is your answer \n";
    cin >> user Guess;
    if(user Guess == pauphrase.length())
    {
      cout << "congrati";
    }
```

// enter new file. creates & append data to an existing file use (ios::app)

```
        else
            {
                cout << " sorry try again later\n";
                out streams << pau phrase << endl;
            }
    } }
    instream - close()
    outstream - close()
}
```

User defined data structure

1) Struct:

Eg:

```
Struct Students.
{
    string name;
    float GPA;
};
Students S1
int main()
{
    Students S1;
    S1.name = " Subash";
    S1.float GPA = "4.0";
    Cout << S1.name << endl;
    cout << S1.GPA << endl;
return 0;
```

array ⇒ student S[10];
S[0].name =
S[1].name =

2) class (copy note)

3) pointer (copy note)

4) Linked list

sizeof

```
int main()
{
    int x = 5;
    size_t y = sizeof(x);
    printf("size of x is %zd\n", y);
    return 0;
}
```

typeid

```
struct A { int x; };
struct B { int x; };

A a1;
A a2;
B b1;
B b2;
```

```
int main()
{
    if(typeid(a1) == typeid(b1))
    { puts("same"); }
    else
    {
        puts("different");
    }
    return 0;
}
```

Templates

Template in C++ is defined as a blueprint or formula for creating a generic class or a function. To simply put, you can create a single function or single class to work with different data type using templates

Template in C++ works in such a way that it's get expanded at compile time, just like macros and allows a function or class to work on different data types without being rewritten

1) Function template
2) class template

Function template syntax

```
template<class type> return type func-name(parameter list)
{
    // body
}
```

class template syntax

```
template <class type>
class class_name
{
    // body
}
```

Function template

```
#include <iostream.h>
using namespace std;
template <class X>   // can replace class keyword by "typename"
X func (X a, X b)
{
  return a;
}
int main()
{
cout <<func(15.8)<<endl;
cout<<func('P','q')<<endl;
cout <<func(7.5, 9.2)<<endl;
return 0;
}
```

o/p 15, P, 7.5

Each type are sent into respective functions.

class template

```
#include <iostream.h>
using namespace std;
template <class c>
class A
{
  private:
    c, a, b
  public:
    A (cx, cy)
    {
      a = x;
      b = y;
    }
void show ()
{
cout << "the addition of" <<a" "<<b
  <<"b"<<"is" <<add() <<endl;
}
c add()
{
  c c=a+b;
  return c;
}
};
int main () {
A add int(8, 6);
A add float (3.5, 2.6)
A addint. show();
cout endl;
A add double. show()
return 0;
}
```

PDF Created Using

Camera Scanner

Easily Scan documents & Generate PDF

https://play.google.com/store/apps/details?id=photo.pdf.maker