Love is in the air :P #Oreo #Dairy Milk. #Yumm

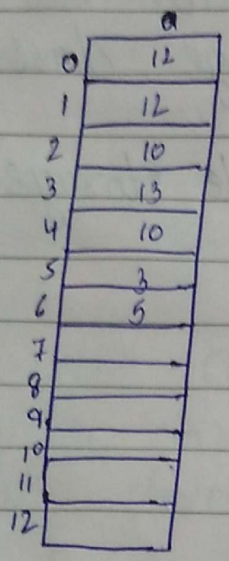
43

14/2/17

Array

Array is the group of elements

a[ ]




```
void main()
{ int a[10], n, i;
  printf("Enter the size of array");
  scanf("%d", &n);
  printf("Enter the value of array");
  for(i=0, i<n, i++)
  { scanf("%d", &a[i]);
}
getch();
}
```

```
void main()
{ int a[6] = {1, 2, 3, 4, 5, 6};
```

For initialisation u can leave a[ ] this blank otherwise not.

Date: Neelgagan
Page No. 2

2-D array example → Matrices

```
{ int a[2][2], i,j,m,n;
 printf ("Enter the size of the array");
 scanf ("%d %d", &m,&n);
 printf ("Enter the value of array a");
 For (i=0 , c[i][j]=0 i<m, i++)
  { for (j=0, j<n; j++)
  { scanf ("%d," & a[i][j]);
  }} getch();
  }
```


```
{ c[i][j] = a[i][j] + b[i][j];
  printf ("%d," c[i][i]; }}
  getch();
  }
```

```
for (i=0; i<m; i++)
{ for (j=0; j<n; j++)
 }   c[i][j] = 0;
for (k=0; k<m; k++)
{ c[i][j] = c[i][j] + a[i][k] * b[k][j];
 }
```

Multiplication of matrices

(#) Transpose of a Matrix

Functions

```
# include <stdio.h>
# include <conio.h>
 void message ();  - Function prototype
 void main ()  - Calling Function
 {
  printf ("Hi");
  message ();
  getch ();
 }
  void message ()
  { printf ("Hello world"); }
  }
```

} definition of main fn

} definition of message fn

Output
Hi
Hello world

We can call as many fns as we can.

(#)

```
 # include <stdio.h>
  # include <conio.h>
 void punjab ();
 void haryana ();
 void delhi ();
 void main ()
 {
  printf ("I am main");
    punjab ();
      haryana ();
      delhi ();
    getch ();
  }
void delhi ()
{ printf ("Kejriwal");
```

```
}
void haryana ()
{ printf ("Jaat");
}
void punjab ()
{ printf ("Udta Punjab");
  delhi ();
}
```

Output
I am main
Udta Punjab
Kejriwal
Jaat
Kejriwal

→ If program has only one fn then it is called one fn.

→ There is no limit to no. of fns in C

→ Sequence doesn't matter

→ Fn is called when there is ; after parentheses.

```
# include <stdio.h>
# include <conio.h>
void message ();
void main ()
{ printf ("Hi");
  message ();
  getch ();
}
void message
{ printf ("Hello world");
main ();
}
```

Main fn can be called within a fn

This will go into infinite loop

If a fn calls itself then it is called recurssive fn.

Passing values b/w fns

```
# in  # include <stdio.h>
      # include <conio.h>
      int calsum (int a, int b, int c);
      int calsum (int a, int b, int c);
      void main ()
      { int a, b, c, d;
      printf ("Enter the values of a, b, c");
      scanf ("%d %d %d", &a, &b, &c);
      d = calsum (a, b, c);
      printf ("%d", d);
      getch ();
      }
      int calsum (int x, int y, int z)
      { int sum;
      sum = x+y+z;
      return (sum);
      }
```

a, b, c → actual parameter

Sum is actual parameter

x, y, z → Formal parameter

If we define as (int, float, float) ~~if~~ above then it should defined in same order below.

(#)

```
int num (int n)
{ printf ("Enter any number");
  scanf ("%d", &n);
  if (n >= 10 && n <= 100)
  return (n);
  else
  return (n+32);
  }
```

return (a,b); → Invalid
return statement can give only one value at a time.

(#)
```
#include <stdio.h>
#include <conio.h>
void fun();
void main()
{ int a;
  a = 10;
  fun(a);
  getch();
}
void fun()
{ int b = 20;
  printf("%d", b);
}
```

10/3/17

Scope Rules of functions

#Holi Khelat Hai Piyaji m...
Holi khele Raghuveera.
#Happy Holi in advance.
#louisinka buddays

(#)
```
void display(int);
void main()
{ int i = 20;
  display(i);
getch();
}
void display(int j)
{ int k = 35;
  printf("%d", j);
  printf("%d", k);
}
```

Date:
Page No.

Calling convention depends on two things:
i) The order in which the arguments are passed to the fn
ii) Which fn performs the ~~cleaning~~ clean up of variables when the control returns from the fn.

i) Arguments can be passed from left to right
ii) " " " " " right to left.

14/3

```
int a = 1;
printf("%d %d %d", a, ++a, a++, ++a);
```

It goes into memory from right to left (Calling convention from right to left)
It is displayed from left to right.

Output - 3, 3, 1
a++ ⇒ 1 (First one is passed & then incremented)
++a ⇒ 3
a ⇒ 3

Using Library fns

```
~~int~~ ( int i = 20, j = 30;
printf("%d %d %d", i, j);
```

Output
20, 30, Garbage value

```
int i = 20, j = 30;
printf("%d", i, j);
```

Output
20

Date:
Page No.

1.) Return type of fn
2.) Calling fn by value or by reference
3.) Recursive fn

} Advance features of fn

1.) What do you mean by multi-dimensional array?
2.) What is function?
3.) Define call by value and call by reference.

14/3/17

(#) int i;
i = 20;

i → variable

20

65524 → Memory address or location no.

```
void main()
{ int i;
  i = 20;
printf("value of i = %u", &i);
printf("value of i = %d", i);
 getch();
}
```

Output
65524
20

address

```
{ int i, *j;
```

Pointer variable

j = &i;

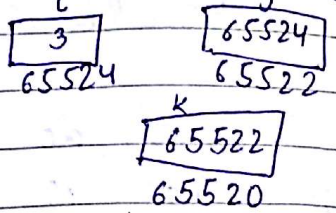
j will hold the ~~char~~ address of i

(#)

```
#include <stdio.h>
void main()
{ int i = 3;
printf("Address of i = %u", &i);
printf("Value of i = %d", i);
printf("Value of i = %d", *(&i));
getch();
}
```

*(&i) → Value of address of i

**k contains the address of pointer variable *j




```
# include <stdio.h>
void main()
{ int i=3, *j, **k;
    j = &i;
    k = &j;
printf("Address of i = %u", j);        o/p - 65524
printf("Address of i = %u", *k);       o/p - 65522
printf("Address of j = %u", &j);       o/p - 65522
  "  ("  "    "  k = %u", k);          o/p => 65522
  "  ("  "    "  k = %u", **k);        o/p => 3
```

(#)

```
# include <stdio.h>
void swapV(int x, int y);
void main()
 { int a=10, b=20;
   swapV(a,b);
   printf("a = %d  b = %d", a, b);
getch(); }
void swapV(int x, int y)
{ int t;
   t = x;
   x = y;
   y = t;
printf("x = %d y = %d", x, y);
 }
```

o/p
x=10, y=10
a=10, b=20

(#)

```
# include <stdio.h>
void swapir(int *, int *);
void main()
{ int a=10, b=20;
```

Exchange of Address

```
swapr (&a, &b);
printf ("a = %d  b = %d", a, b);
getch (); }
void swapr (int *x, int *y)
{ int t;
   t = *x;
  *x = *y;
  *y = t;
  printf ("x = %d  y = %d", x, y);
  }
```

Output

x = 20 y = 10

a = 20 b = 10

(#)

```
# include <stdio.h>
void areaperi (int, flocd *, float *);
void main ()
{
 int radius;
float area, perimeter;
printf ("Enter the radius of circle");
scanf ("%d", &radius);
areaperi (radius, &area, &perimeter);
printf ("Area = %f \n", area);
printf ("Perimeter = %f \n", perimeter);
getch ();
}
void areaperi (int r, float *a, float *p)
{ *a = 3.14 * r * r;
  *p = 2 * 3.14 * r;
 }
```

Passing Array elements in a fn

Call by value

```
void display (int);
void main ()
{ int i;
int marks = { 55, 65, 75, 56, 78, 78, 90};
for (i=0; i<=6 ; i++)
   display ( marks [i]);
getch ();
}
void display (int m)
{ printf ("%d", m);
}
```

(#)

(#)
```
void display ( int *);
void main ()
{ int i;
int marks = { 55, 65, 75, 56, 78, 78, 90};
for ( i=0, i<=6; i++)
 display ( &marks [i]);
getch ();
}
void display (int *m)
{ printf ("%d", *m);
}
```

(#)
```
void main ()
{ int i=3, *x;
float j=1.5, *y;
char k='c', *z;
printf ("Value of i = %d \n", i);
printf ("Value of j = %d \n", j);
```

A[5][2] → 2 columns
↓
5 rows

```
printf("Value of k = %d \n", k);
x = &i;
y = &j;
z = &k;
printf("Address of x = %u \n", x);
                y = %u
                z = %u
```

x++; +2
y++; +4
z++; +1

65524
65526 65520
65521
65528
65532

Passing an entire array to a fn

(#)

```
void display (int *, int);
void main()
{ int num[] = {24, 34, 12, 44, 56, 17};
    display(&num[0], 6);
  getch();
}
void display (int *j, int n)
{ int i;
for (i=0; i<=n-1; i++)
{ printf("element = %d \n", *j);
  j++;
}
}
```

↓ base address is being passed

num[i] or
*(num+i) or
*(i+num) or
i[num]

2-D Array

Pointers & 2-D Arrays

```
#include <stdio.h>
void main()
{ int s[4][2] = {{1234, 5656},
                 {1212, 3333},
```

```
    {1434, 80},
    {1312, 78} };
int i;
for(i=0; i<=3; i++)
printf("Address of %d the 1-D array = %u \n", i, s[i]);
getch();
}
```

s0,0	s0,1	s1,0	s1,1	s2,0	s2,1	s3,0	s3,1
1234	56	1212	35	1434	80	1512	78
65508	65510	65512	65514	65516	65518	65520	65522

* (s[2]+1)
65518

s[2][1] or
* (s[2]+1) or
((s+2)+1)