CS F111: Computer Programming

(Second Semester 2020-21)

Lect 17: 2DArray and Strings

Dr. Nikumani Choudhury
Asst. Prof., Dept. of Computer Sc. & Information Systems
nikumani@hyderabad.bits-pilani.ac.in

BITS Pilani

Hyderabad Campus

Two-Dimensional Arrays

month

year

	0	1	2	3	4	5	6	7	8	9	10	11
0	30	40	75	95	130	220	210	185	135	80	40	45
1	25	25	80	75	115	270	200	165	85	5	10	16
2	35	45	90	80	100	205	135	140	170	75	60	95
3	30	40	70	70	90	180	180	210	145	35	85	80
4	30	35	40	90	150	230	305	295	60	95	80	30

Average Yearly Rainfall (in mm)

```
int table [5][12] = {
      {30,40,75,95,130,220,210,185,135,80,40,45},
      {25,25,80,75,115,270,200,165, 85, 5,10, 16},
      {35,45,90,80,100,205,135,140,170,75,60,95},
      {30,40,70,70, 90,180,180,210,145,35,85,80},
      {30,35,40,90,150,230,305,295, 60,95,80,30}
   };
```

OR

```
int table [5][12];
```

```
int table [5][12] =
{30,40,75, … ,80,30};
```

Two-Dimensional Array Initialization continued…

```
int table [ ][12] = {
        {30,40,75,95,130,220,210,185,135,80,40,45},
        {25,25,80,75,115,270,200,165, 85, 5,10, 16},
        {35,45,90,80,100,205,135,140,170,75,60,95},
        {30,40,70,70, 90,180,180,210,145,35,85,80},
        {30,35,40,90,150,230,305,295, 60,95,80,30}
    };
```

Need not specify the size of the first dimension when the array is completely initialized

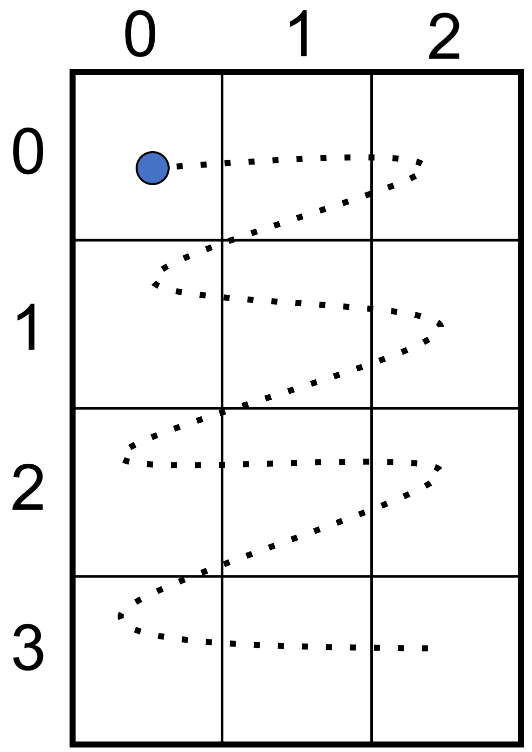
- If the values are missing in an initializer, they are automatically set to zero

```
int a[3][ ] = {2,4,6,8,10,12};
int a[ ][ ] = {2,4,6,8,10,12};
```

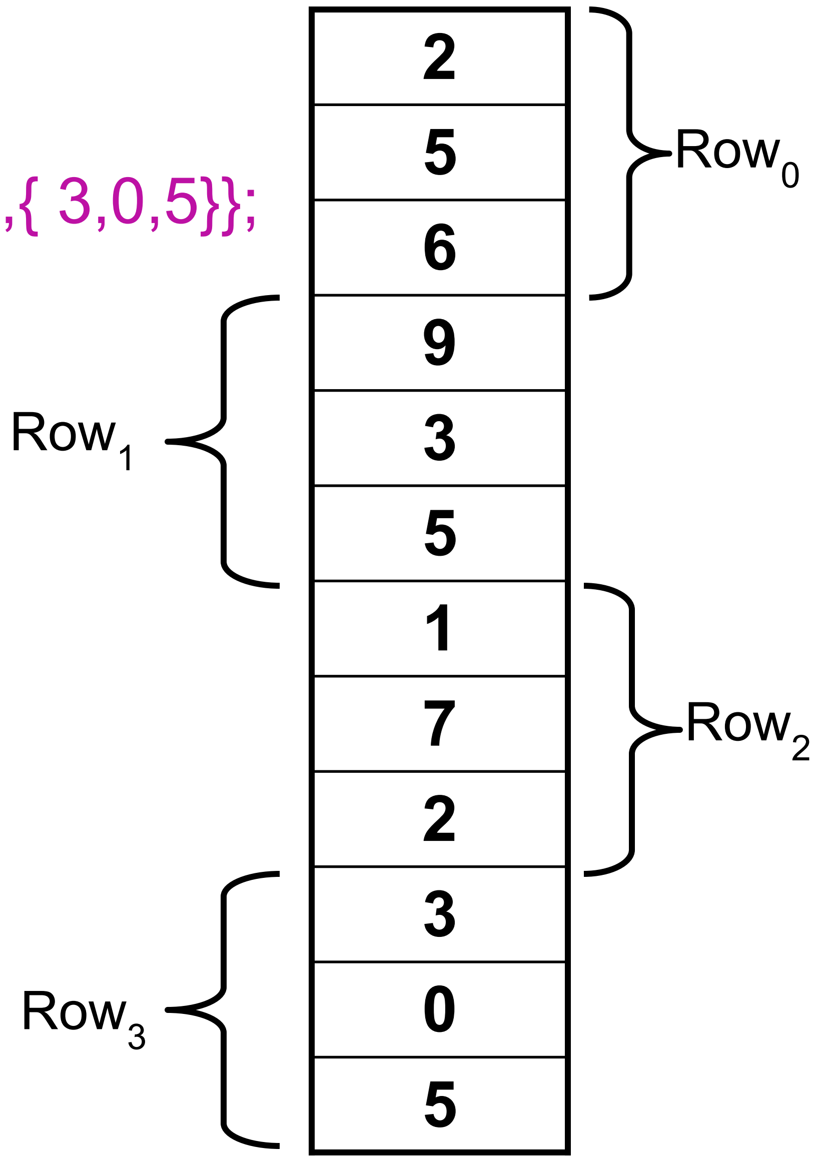
Not allowed

Memory layout

```
int table[4][3]={{2,5,6},{9,3,5},{1,7,2},{ 3,0,5}};
```



Row major order



Accessing Two Dimensional Arrays

```
int a[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};

for (i = 0; i < 3; i++)
  {
     for ( j = 0; j < 4; j++)
      {
       printf("%-4d",a[i][j]);
      }
      printf("\n");
  }
```

Output

```
1   2   3   4
5   6   7   8
9   10  11  12
```

Matrix Multiplication using 2D Arrays

```
#include <stdio.h>
void mult_matrices (int a[ ][3], int b[ ][3], int result[ ][3]);
void print_matrix(int a[ ][3]);
main(void)
  {
     int p[3][3] = {  {1, 3, -4}, {1, 1, -2}, {-1, -2, 5}  };
     int q[3][3] = {  {8, 3, 0}, {3, 10, 2}, {0, 2, 6}  };
     int r[3][3];
     mult_matrices (p, q, r);
     print_matrix (r);
  }
void mult_matrices (int a[ ][3], int b[ ][3], int result[ ][3])
{ int i, j, k;
   for(i=0; i<3; i++)
     {
        for(j=0; j<3; j++)
          {  result[i][j] = 0;
             for(k=0; k<3; k++)
           { result[i][j] += a[i][k] * b[k][j]; }
          }
     }
}
```

Passing the whole array

No. of cols in first matrix = No. of rows in second

p: mxn, q:nxp, R:mxp

$R_{11} = p_{11}q_{11}+p_{12}q_{21}+\ldots p_{1n}q_{n1}$

$R_{12} = p_{11}q_{12}+p_{12}q_{22}+\ldots p_{1n}q_{n2}$

$\ldots$

$R_{mp}=p_{m1}q_{1p}+p_{m2}q_{2p}+\ldots p_{mn}q_{np}$

```
void print_matrix(int a[ ][3])
{   int i, j;
     for (i=0; i<3; i++)
        {
            for (j=0; j<3; j++)      {
printf("%d\t", a[i][j]);      }
            printf("\n");
        }
}
```

Matrix Multiplication using 2D Arrays

```
#include <stdio.h>
void mult_matrices (int a[ ][3], int b[ ][3], int result[ ][3]);
void print_matrix(int a[ ][3]);
main(void)
  {
     int p[3][3] = {  {1, 3, -4}, {1, 1, -2}, {-1, -2, 5}  };
     int q[3][3] = {  {8, 3, 0}, {3, 10, 2}, {0, 2, 6}  };
     int r[3][3];
     mult_matrices (p, q, r);
     print_matrix (r);
  }
void mult_matrices (int a[ ][3], int b[ ][3], int result[ ][3])
{ int i, j, k;
   for(i=0; i<3; i++)
     {
        for(j=0; j<3; j++)
          {  result[i][j] = 0;
             for(k=0; k<3; k++)
           { result[i][j] += a[i][k] * b[k][j]; }
          }
     }
}
```

No. of cols in first matrix = No. of rows in second

p: mxn, q:nxp, R:mxp

$R_{11} = p_{11}q_{11}+p_{12}q_{21}+\ldots p_{1n}q_{n1}$

$R_{12} = p_{11}q_{12}+p_{12}q_{22}+\ldots p_{1n}q_{n2}$

$\ldots$

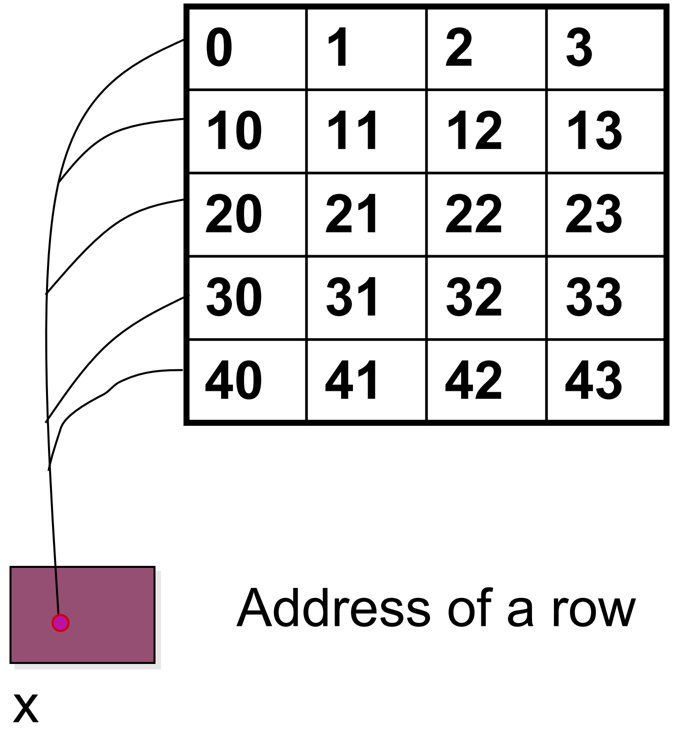
$R_{mp}=p_{m1}q_{1p}+p_{m2}q_{2p}+\ldots p_{mn}q_{np}$

```
void print_matrix(int a[ ][3])
{   int i, j;
     for (i=0; i<3; i++)
        {
            for (j=0; j<3; j++)      {
printf("%d\t", a[i][j]);      }
            printf("\n");
        }
}
```

Passing the whole array

Passing a Row

```
void print_square (int [ ]);
int main( )
{
  int table[5][4] = { {0,1,2,3}, {10,11,12,13}, {20,21,22,23},
                       {30,31,32,33}, {40,41,42,43} };
  for (int row = 0; row < 5; row++)
      print_square (table [row]);
  return 0;
}
void print_square (int x[ ])
{
   for (int col = 0; col < 4; col++)
     printf("%6d", x[col] * x[col]);
   printf("\n");
   return;
}
```



Output:

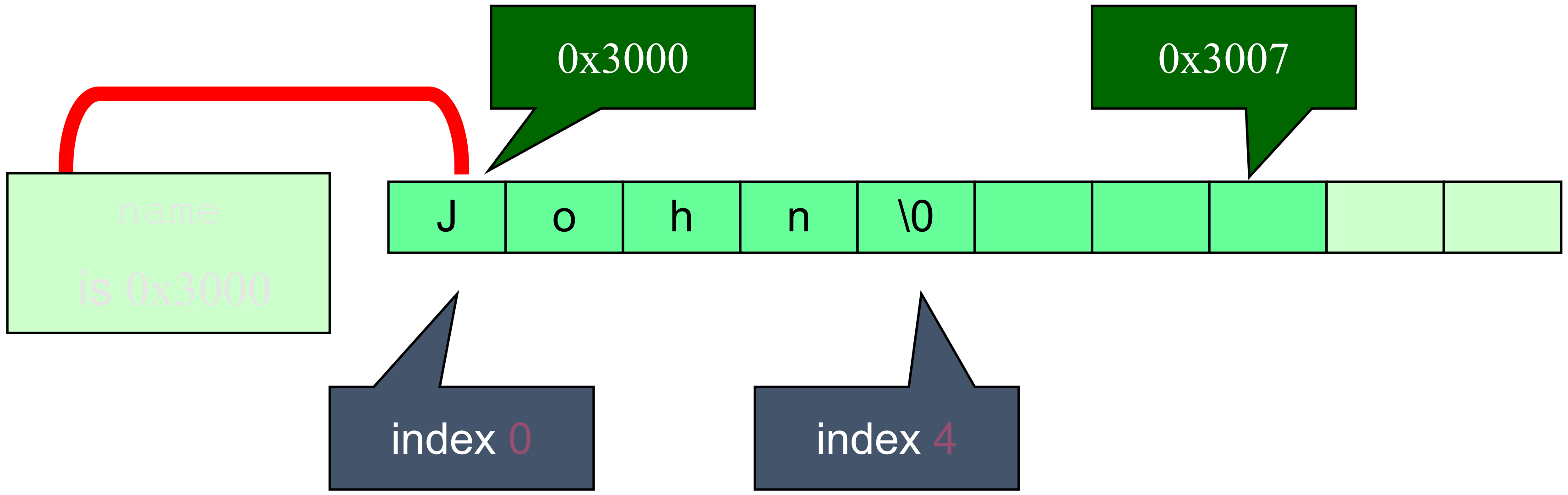
```
  0      1    4    9
100    121  …
  …
```

Character Arrays and Strings

- A string is a series of characters treated as a unit.
- No standard adopted for implementation.
- A string in C is a variable length array of characters that is delimited by the null character.

‘\0’ and ‘0’ are they equal?

A Char in a String



```
char name[8] = “John”;
int  i = 2;

printf (“Character at index %d is %c.\n”, i, name[i]);
```

Declaring and Initializing Strings

- char city [7];

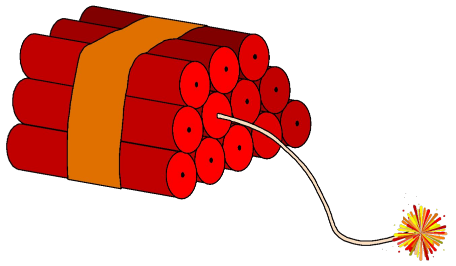
char * str3 = "BITS PILANI";

- char city [7] = “PILANI”;
- char name [12] = “BITS PILANI”;
- char name [12] = { ‘B’,‘I’,‘T’,‘S’,‘ ’,‘P’,‘I’,‘L’,‘A’,‘N’,‘I’,‘\0’};

  must

- char name [ ] = “BITS PILANI”;
- char name [15] = “BITS PILANI”;

B	**I**	**T**	**S**		**P**	**I**	**L**	**A**	**N**	**I**	**\0**	**\0**	**\0**	**\0**



Common Mistakes

- char name [10] = “BITS PILANI” — Not sufficient space
- char name [ ]; — Array size missing
- char name [12];
  name = “BITS PILANI”;
- char str1[6] = “Hello”;
  char str2[6];
  str2 = str1;

$\}$ Assignment to expression with array type
Name of string is a pointer const that cannot be used as a lvalue

Importance of ‘\0’

```
main( )
{
  char name [ ] = “John”;
  int i = 0;
  while ( i <= 3 )
    {
       printf (“%c”, name[i] );
       i++;
    }
}
```

```
main( )
{
  char name [ ] = “John”;
  int i = 0;
  while ( name[i] != ‘\0’ )
    {
       printf (“%c”, name[i] );
       i++;
    }
}
```

Without knowing the length

String Input/Output

Using scanf()

```
 char text[10];
 printf (“Enter a string: ”);
 scanf (“%s”,text);
 printf (“The string is : %s”,text);
```

Sample output:

```
Enter a string: Well
The string is: Well
----------------------------------------
Enter a string: Well done
The string is: Well
```

```
char text1[5], text2[5];
scanf (“%s %s”, text1, text2);
```

W	e	l	l	\0	?	?	?	?	?

Note: scanf() terminates its input on the first white space (blank, tab, cr, ff, and new line)

String Input/Output continued…

```
char text1[7], text2[7];
scanf ("%4s %3s", text1, text2);
```

With input: Well Done

W	e	l	l	\0	?	?

D	o	n	\0	?	?	?

Reading a Line using scanf ( ):

```
char text[80];
printf("Enter a string: ");
scanf("%[^\n]",text);
printf("The string is : %s", text);
```

Output:

```
Enter a string: Well done!
The string is: Well done!
```

String Input/Output continued…

Using getchar ( )

```
do
{
  ch = getchar ( );
  line [c] = ch;
  c++;
}
while (ch != ‘\n’);
c = c – 1;
line [c] = ‘\0’;
```

Using gets ( )

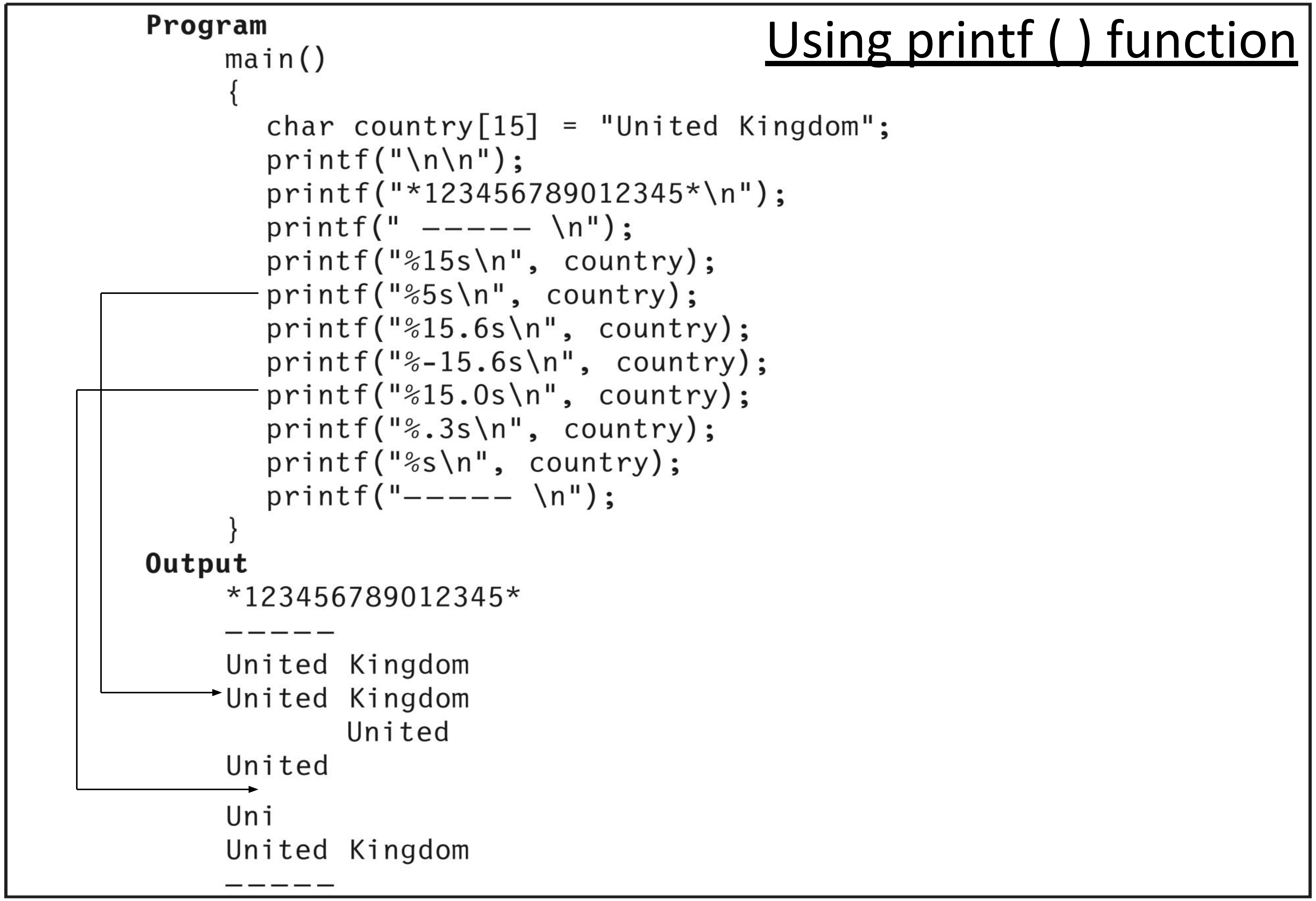
```
char line[100];
printf(“Enter a string :”);
gets(line);
printf(“The string is :%s”, line);
```

Output

```
Enter a string :BITS PILANI
The string is : BITS PILANI
```

gets() reads from stdin until an end of line or end of file is reached. **getchar**() reads a single character from stdin. Since **gets**() does not check if there is space for the line being read **in the** pointer it is passed, it is generally considered **unsafe.**

String Input/Output continued…



Using printf ( ) function

Program

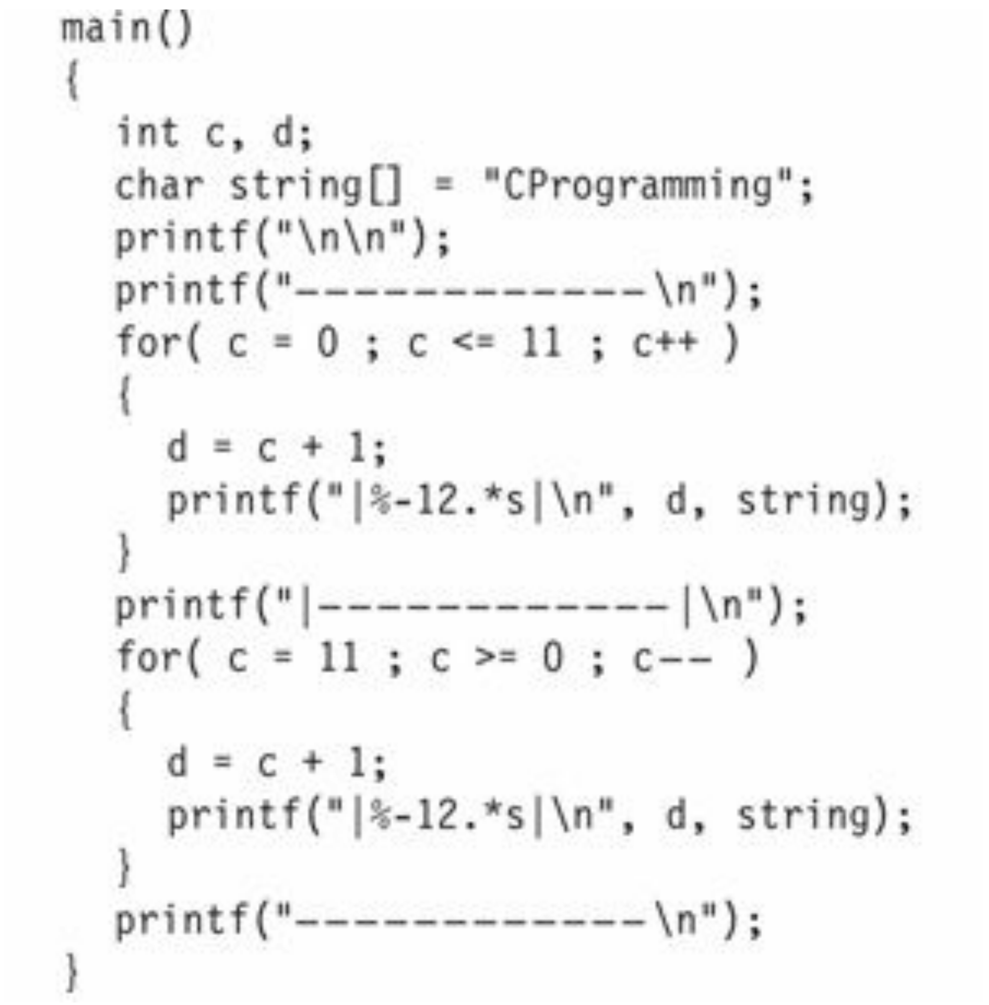
```
    main()
    {
      char country[15] = "United Kingdom";
      printf("\n\n");
      printf("*123456789012345*\n");
      printf(" ----- \n");
      printf("%15s\n", country);
      printf("%5s\n", country);
      printf("%15.6s\n", country);
      printf("%-15.6s\n", country);
      printf("%15.0s\n", country);
      printf("%.3s\n", country);
      printf("%s\n", country);
      printf("----- \n");
    }
```

Output

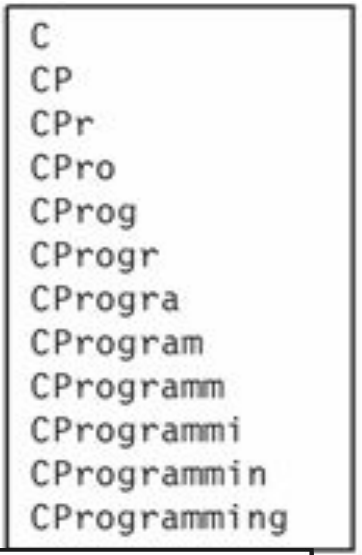
```
    *123456789012345*
    -----
    United Kingdom
    United Kingdom
           United
    United

    Uni
    United Kingdom
    -----
```

String Input/Output continued…

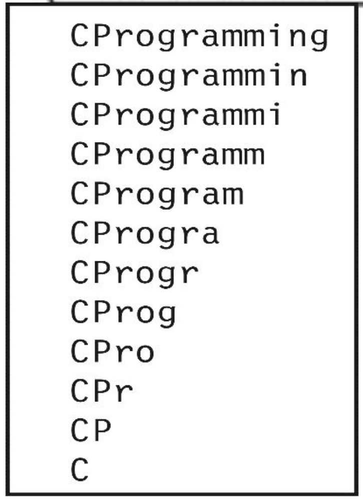
- <u>Using printf ( ) function with variable field width or precision</u>



```
main()
{
  int c, d;
  char string[] = "CProgramming";
  printf("\n\n");
  printf("------------\n");
  for( c = 0 ; c <= 11 ; c++ )
  {
    d = c + 1;
    printf("|%-12.*s|\n", d, string);
  }
  printf("|------------|\n");
  for( c = 11 ; c >= 0 ; c-- )
  {
    d = c + 1;
    printf("|%-12.*s|\n", d, string);
  }
  printf("------------\n");
}
```



```
C
CP
CPr
CPro
CProg
CProgr
CProgra
CProgram
CProgramm
CProgrammi
CProgrammin
CProgramming
```



```
CProgramming
CProgrammin
CProgrammi
CProgramm
CProgram
CProgra
CProgr
CProg
CPro
CPr
CP
C
```