“C Programming”

Conditional Expression

conditional Operator: consists of two symbols (? and :)

Syntax: `expr1 ? expr2 : expr 3`

Example: `i > j ? i : j`

```
int i, j, k;
i=1;
j=2;
k = i > j ? i : j;          /* k is 2*/
k = (i >= 0 ? i : 0) + j; /*k is 3*/
```

Conditional expression

Using if-else

```
if ( i > j)
    printf(“%d\n”, i);
else
    printf(“%d\n”, j);
```

Using Conditional Operator

```
printf(“%d\n”, i > j ? i : j);
```

3. Control Statements

break

```
#include <stdio.h>
 int main()
{
        int i;
        for(i=0;i<10;i++)
        {
                if(i==5)
                {
                        printf("\nComing out of for loop when i = 5″);
                        break;
                }
                printf("%d ",i);
        }
}
```

Jumps

Output:

0 1 2 3 4

Coming out of for loop when i = 5

continue

- A **continue statement** jumps to the point just before end of loop body
- **Syntax: continue;**

```
n=0;
sum=0;
while(n<10) {
    scanf("%d", &i);
    if(i == 0)
        continue;
    sum += i;
    n++;
    /* continue jumps to here */
}
```

Output : 1 2 4 5

goto

- A **goto statement** causes an unconditional jump from the goto to a statement **in the same function.**statement MUST have a **label**
- Used to break the loops that are deeply nested, break works for only for a single level of loop
- break, continue and return are like restricted goto, sufficient for most scenarios
- **Rarely used – Not recommended to use(unless cannot do without it)**

Syntax:

```
goto label;
…..
mylabel : statement;
```

```
example:
while(..) {
   switch(…) {
        …
        goto loop_done;
        // break won't work here
        …
   }
}
loop_done : …
```

Quiz 5

1. How can you make the loops to run infinite number of time ?
2. How do you come out of infinite loops?
3. What is the difference between while loop and do-while loop?

Class Room Work

1. Write a program to print the multiplication table of a given number (upto 10) using for loop.
2. Write a program to find the number of digits in a given integer number. Use While loop or Do While loop.

Class Room Work

Solution: */* Multiplication Table */*

```
/* C program to find multiplication table up to 10. */

#include <stdio.h>
int main()
{
    int n, i;
    printf("Enter an integer to find multiplication table: ");
    scanf("%d",&n);

   for(i=1;i<=10;++i)
    {
        printf("%d * %d = %d\n", n, i, n*i);
    }
    return 0;
}
```

Class Room Work

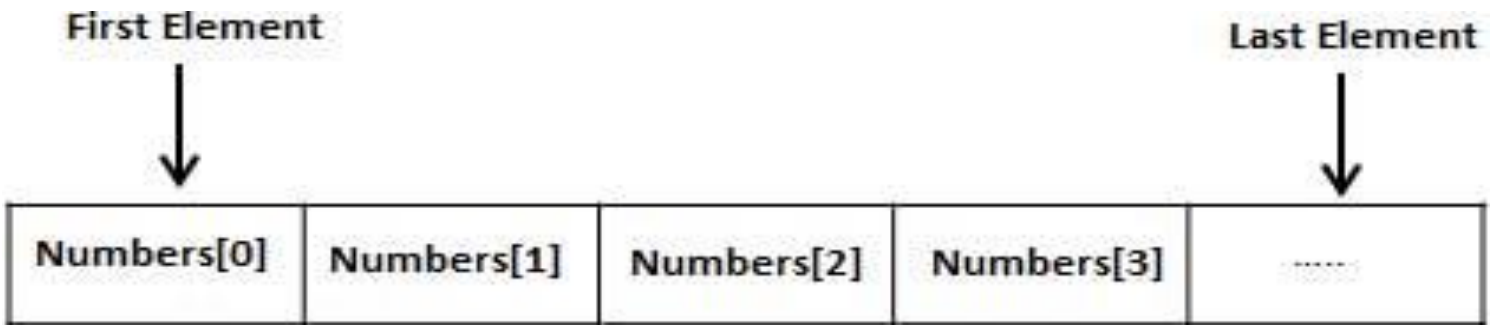
Solution: */ * Number of digits */*

```
#include <stdio.h>
int main()
{
  int n,count=0;
  printf("Enter an integer: ");
  scanf("%d", &n);
  while(n!=0)
  {
      n/=10;              /* n=n/10 */
      ++count;
  }
  printf("Number of digits: %d",count);
}
```

Arrays

Arrays - Introduction

- Definition: Array is a derived data type. Array is a collections of variables of same type.
- Fixed size, size defines the number of elements in the array
- Stored in contiguous memory locations
- Array element is accessed by its index value



Example of 1-D Array

```
int iarr[3] = {2, 3, 4};

char carr[20] = “ACTS" ;

char other_carr[20] = {‘A’, ’C’, ’T’, ’S’};

 float farr[3] = {12.5,13.5,14.5} ;
```

Declaring & Initializing Array

Declaration: type Array_Name [array size]

Example: 5 element array of integer type

```
#define N 5
int main(){
        int arr [5];
        int third_arr[ ] = {1,2,3,4,5};
        int second_arr[N]={1,2};
        ….
}
```

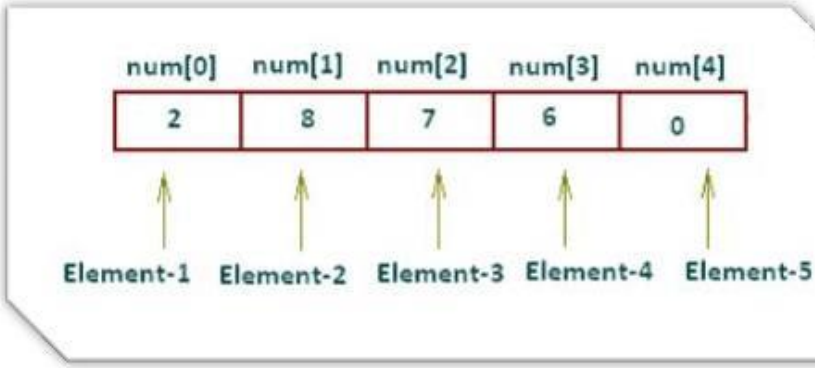
Initialization:

```
for(j=0;j<5;,j++)
        arr [ j ]=j+1;
```

Declaring & Initializing Array

Array Size specified Directly:

int num[5] = {2,8,7,6,0};



Array Size specified In-Directly

int num[] = {2,8,7,6,0}; // total 5 elements in array

int arr2[5] = {1}; // initialized as {1,0,0,0,0}

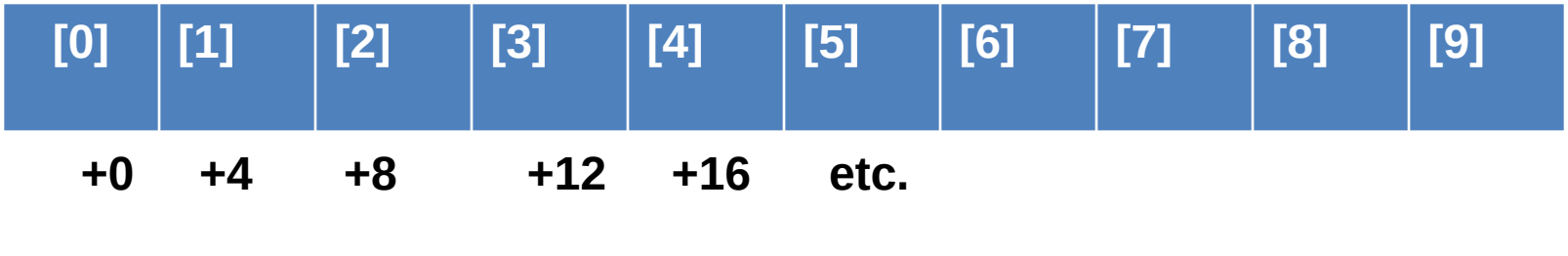
int arr3[5] = {0,1,0,2,0};

int arr4[5] = { [1] = 1, [3]=2}; // designated initializer(C99)

Visualizing 1-D Arrays

Representing a single dimension integer Array

```
int A[10];      // assuming integer size as 4 bytes
```



program on 1-D Array

```
#include <stdio.h>
int main()
{
    int num[] = {21,18,57,45,50};
    int i;
    for(i=0;i<5;i++)
    {
        printf("\n Integer Array Element num[%d] : %d",i+1,num[i]);
    }
    return 0;
 }
```

Multidimensional Arrays

Multidimensional Arrays

- Arrays with more than one dimension
- 2D Arrays : grid of rows and columns
- GCC allows arrays of up to 29 dimensions (source:linuxtopia.com)
- Using an array of more than three dimensions is very rare

Representing a double dimension Array

int a[3] [4] ;

a

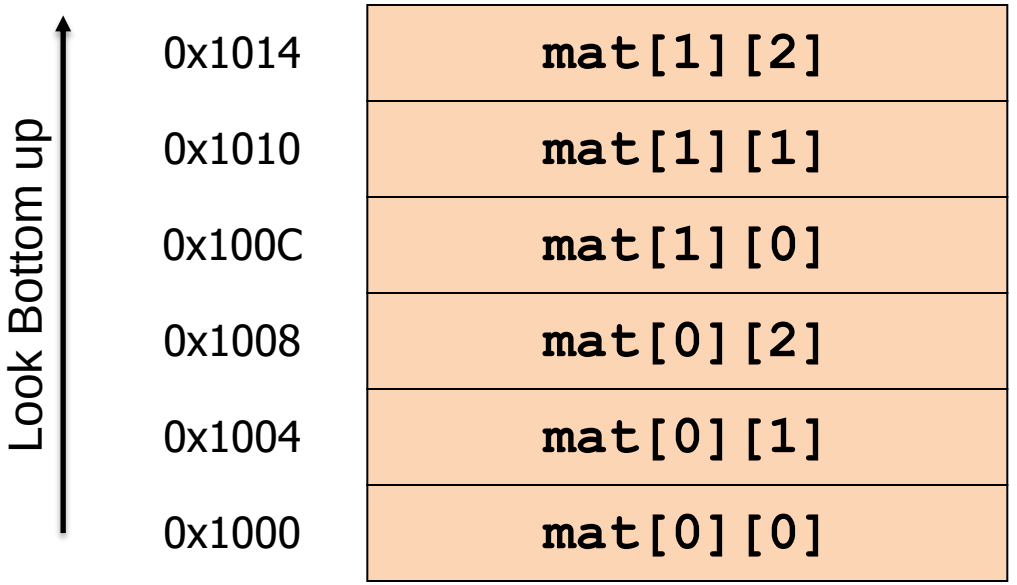
	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Memory Representation – 2D Arrays

```
int  mat[2][3];

mat[1][0] = 17;
```

Valid Initialization



What happens when you initialize

```
mat[0][3] = 20;
```

No bounds checking…causes no syntax error, may cause run time error

2D Arrays

A[3][3]={1,2,3,4,5,6,7,8,9};
A[3][3]={ {1,2,3},{4,5,6},{7,8,9} }; // array of 1D array
a[0][1] = a[0][0] + Size of Data Type

	Col 0	Col 1	Col 2
Row 0	1	2	3
Row 1	4	5	6
Row 2	7	8	9

	Col 0	Col 1	Col 2
Row 0	1 4000	2 4004	3 4008
Row 1	4 4012	5 4016	6 4020
Row 2	7 4024	8 4028	9 4032

[0][0]	[0][1]	[0][2]	[1][0]	[1][1]	[1][2]	[2][0]	[2][1]	[2][2]
Base Address	**+0**	**+4**	**+8**	**+12**	**+16**	**etc.**		

Initializing 2D Arrays

```
#include<stdio.h>
int main()
 {
        int i, j;
        int a[3][2] = {
                        { 1, 4 },
                        { 5, 2 },
                         { 6, 5 }
                };
         for (i = 0; i < 3; i++)
         {
                for (j = 0; j < 2; j++)
                {
                        printf("%d ", a[i][j]);
                }
                printf("\n");
        }
        return 0;

 }
```

```
1 4
5 2
6 5
```

Initializing 2D Arrays

```
#include<stdio.h>
int main()
 {
        int i, j;
        int a[3][2] = { 1, 4, 5, 2, 6, 5 };

         for (i = 0; i < 3; i++)
        {
                for (j = 0; j < 2; j++)
                {
                        printf("%d ", a[i][j]);
                }
                printf("\n");
         }
        return 0;

 }
```

```
1 4
5 2
6 5
```

Initializing 2D Arrays

```
#include<stdio.h>
int main()
 {
        int i, j;
        int a[3][2] = {
                        { 1 },
                        { 5 , 2 },
                         { 6 }
                 };

         for (i = 0; i < 3; i++)
        {
                for (j = 0; j < 2; j++)
                {
                printf("%d ", a[i][j]);
                }
                printf("\n");
         }
        return 0;

}
```

```
1 0
5 2
6 0
```

3D array – array of 2D Arrays

```
#include<stdio.h>
int main()
 {
        int i, j,k;
        int a[2][3][4] = {
                    { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },
                    { {13,14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24} }
                 };
         for (i = 0; i < 2; i++)
        {
                for (j = 0; j < 3; j++)
                {
                        for (k = 0; k < 4; k++)
                        {
                                printf("%d ", a[i][j][k]);
                           }
                printf("\n");
                  }
                printf("\n");
         }
        return 0;
 }
```

1-D Array

1-D Array Program

Write a C Program to display one dimensional array elements with addresses

Write a C Program to Calculate Addition of All Elements in one dimensional array

Write a C Program to Find Greatest Element in Array in one dimensional array

Write a C Program to Reversing the 1-D Array Elements

Write a C Program to Search an element in 1-D Array

Write a C Program to sort elements in 1-D Array

THANK YOU