innovate achieve lead

BITS Pilani

# Lecture 11

## Operations on array variables, record and union data type, Design Issues, storage allocation

# Array op

```c
#include <stdio.h>
int main()
{
    int *p, *q, x, y;
    int A[7]={1,2, 3,4,5,6,7}, B[
    x = 4;
    y = 89;
    p = &x;
    q = &y;
    p = p + q;      //Line 1
    p = A ;         //Line 2
    p = p + 12;     //Line 3
    p = p + A;      //Line 4
    B = q;          //Line 5
    p = p + q;      //Line 6
    q = B;          //Line 7
    return 0;
}
```

Variables in the program: p, q, x, y, A, B

Line 1: Adding unsigned integers-address added to address

Line 2: Assigning start address of array to the location of p

Line 3: Adding a number to an address

Line 4: adding one address to another address

Line 5: Assigning an address to the location where address of B[0] is placed.

Line 6: adding addresses

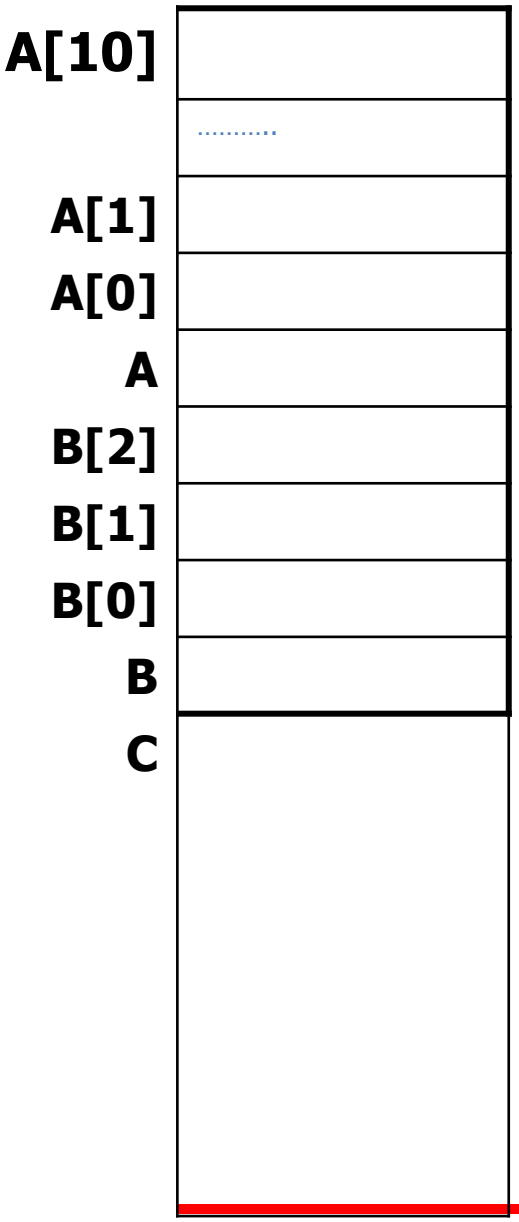
Line 7: assigning start address of the array to the variable q

# Array operations

- The array operation refers to the operations on an array (and not its elements)
- Operations:
  - Concatenation (ADA, Python, Ruby)
  - Assignment (Perl, ADA, Python, FORTRAN 95)
  - Comparison (Python, Ruby- the operator == )
  - Array slicing (Python, Matlab, Perl)
  - Addition (elemental in FORTRAN95)

# Array concatenation

- arrays A[11], B[3], C[20] of integers
- C= A+B
- Possible Internal operations-
  - copying of all elements of A first and then B.
  - Copying of start address of A and mechanism of reaching B[0] after accessing A[11]
  - Copying of sum of both arrays



# Array assignment

```
int x, *p, a[10], b[10];
x=34;
p=&x;
p=a; //valid or invalid in C language?
a=p; //valid or invalid?
```

- Python-array assignment is only reference change (a=b)

# Array comparison

- Elemental comparison
- Fortran 95+ : assignment, arithmetic, relational and logical operators are overloaded for arrays of any size and shape.

# Other types of arrays

- Jagged arrays- Rows can be of varying length
- Associative arrays- can be accessed using strings as hash keys A[“string”]

# Array element selection

Static and dynamic: depending upon whether the index can be computed at compile time or at run time.

- A[k] – run time
- A[10]- compile time
- A[“abc”]- compile time (associative arrays)

# Design the grammar for array element selection

- <element>→ ID SQOP <index> SQCL
- <index>→ ID //run time computable
- <index>→ NUM //compile time computable
- Example: Draw parse trees for A[12], A[k]
- Update expression grammar to include array elements in it

  <expr>→ <expr> <op1> <term>

  <expr>→<term>

  <term>→ <term> <op2> <factor>

  <term>→<factor>

  <factor>→ ID | NUM | OP_PAR <expr> CL_PAR

  <factor>→ <element>