Lecture 9

Arrays: Design Issues, subscript binding, type signature, Compile time layout

Array data type

- Array data type allows the element indices to be computed at run time.
- Example: int A[30]; // in C language

```
for(int i=0;i<10;i++)
        A[(i+1)*2] = i*4;
```

  is a valid C statement as the index is computed at run time.
- An error occurs if (i+1)*2 =30, i.e. if the loop is executed for i<15 (semantic error)
- Language design for an array type focusses on the non-spurious efficient access of data values

Subranges of an array : in C language

- Only size of the array is required to define the subrange
- The subrange is simply [0,..,size-1]
- i.e. the array elements are referenced with only these indices e.g. A[0], A[1], ..A[size-1]
- Address of A[i]= base +(i × w)

  where w is the width of the element, and *base* is the address of the array element A[0]
- w is known at compile time, while i (array subscript) is computed at run time.

Subranges of an array : in Pascal language

- Var A : array [low..high] of typeT;
- Subrange is (low,..,high)
- The array elements are referenced as A[low], A[low+1], .., A[high]
- Address of A[i]= base + (i-low)x w

  where w is the width of the element, and *base* is the address of the array element A[low]
- The width w of the type T is known at compile time, while i (array subscript) is computed at run time.

Array type signature (type expression)

- The type expression for an array variable can be defined by subrange or size and its basic element type.
- example type expression for Pascal array declared as

  a: array [2..4] of integer;

is given as (array, <2,4>, integer)

Pascal array element type

```
Program arraydemo(output);

var
  a: array [2..4] of integer;
  b: array [2..4] of integer;
  c: array [6..8] of integer;
begin
      a[2]:=23;
      b[3]:= 12;
      b[4]:=11;
      c[7]:=20;
      a[3]:=a[2]+b[4];
      a[2]:= b[3] + c[7];
      writeln('a[', 3, '] = ', a[3] );
      writeln('a[', 2, '] = ', a[2] );
end.
```

For array elements it does not include subrange comparison for type checking.

```
$main
a[3] = 34
a[2] = 32
```

Bound checking

- Let the variable A be declared in a C-like language as below

  int A[12]; //static source code

- A[10]- Whether index 10 <12 or not is computed at compile time
- A[k]- Whether index k <12 or not is computed at run time
- A[i*k]-expression i*k (say t1) is computed at run time and whether index t1 <12 or not is computed at run time

Dynamic arrays

- Dynamic arrays are the array variables whose index ranges may be expanded at any time after creation, without changing the values of its current elements.
- Some languages allow dynamic arrays , e.g. perl
- C does not support dynamic arrays as the size is defined while instructing to resize (malloc and realloc in C)

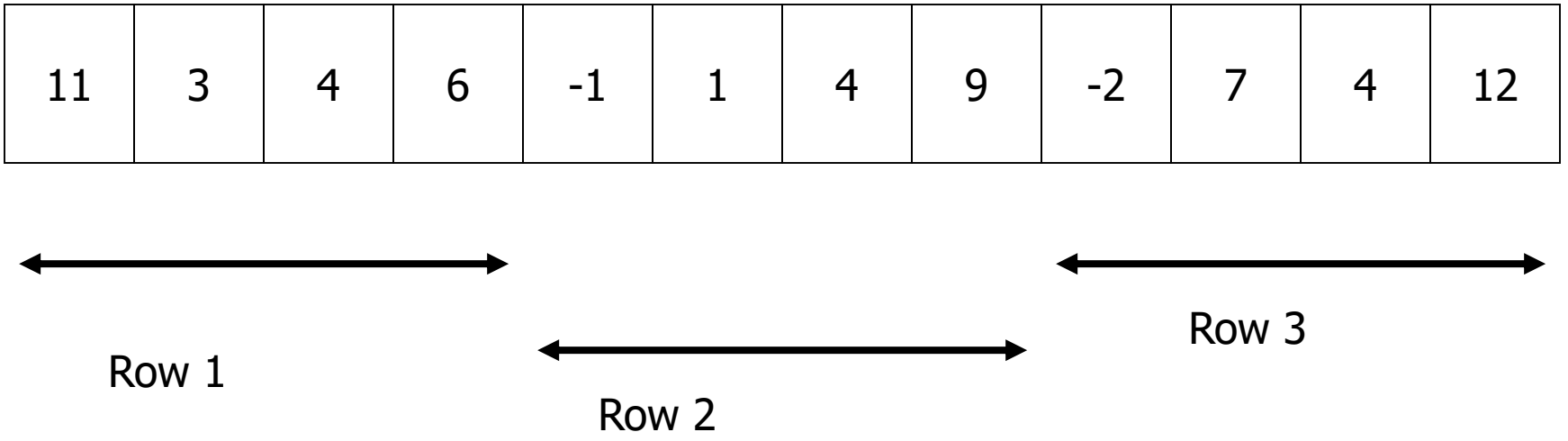
Layout of Multidimensional arrays

- Address computation
  - row-major layout

    rows appear side by side
  - column-major layout

    columns appear side by side

Row major array layout (in C)



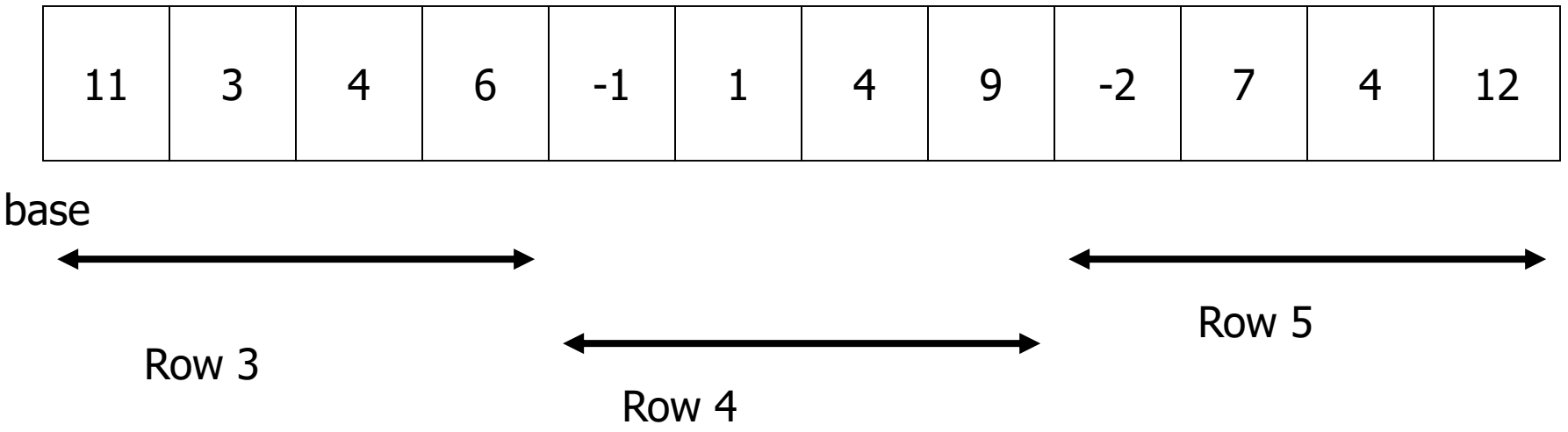
Address computation of an element M[i][j]

let the size of the matrix be m x n

the type of all elements is same (Let w be the width of the type)

address of M[i][j] = (i x n + j) x w

Row major array layout (in Pascal)
var M: array[3..5] of [2..5] of integer



Address computation of an element M[i][j]

subranges [low1..high1] and [low2..high2]

the type of all elements is same (say w be the width)

address of M[i][j] = ??