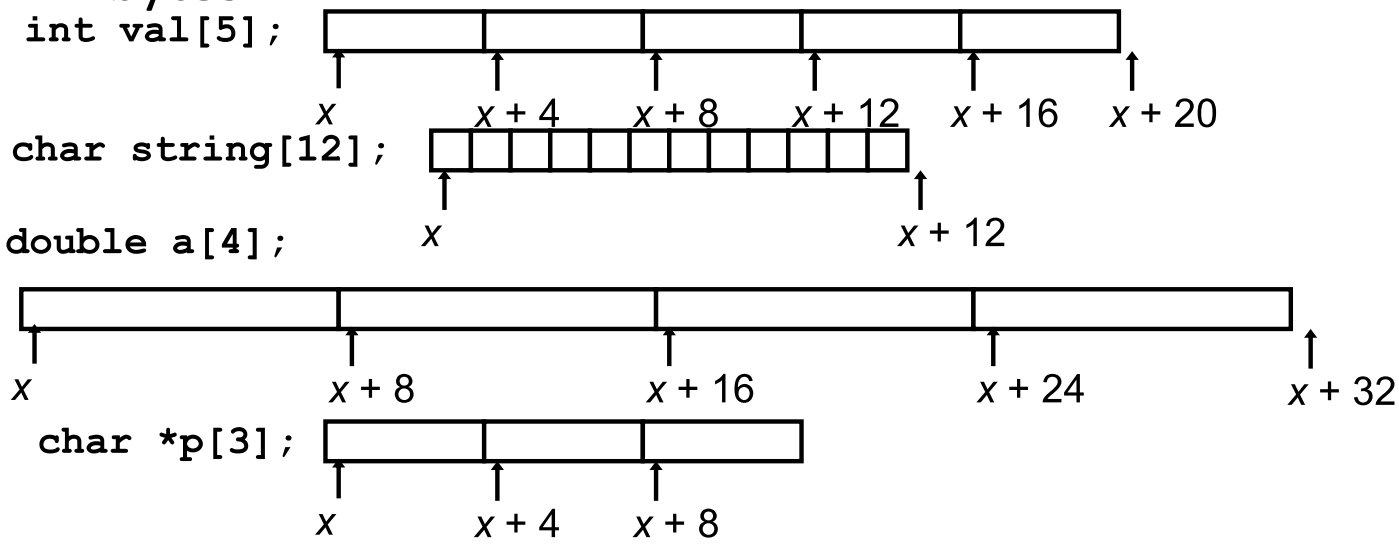
Array Allocation

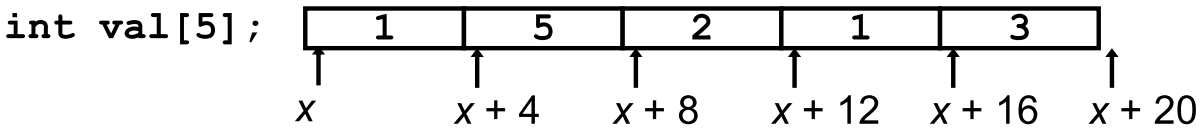
- Basic Principle

  T `A[`*L*`];`

  – Array of data type *T* and length *L*

  – Contiguously allocated region of *L* * `sizeof(`*T*`)` bytes





- Reference Type Value

`val[4]`	`int`	`3`	
`val`	`int *`	*x*	
`val+1`	`int *`	*x* + 4	
`&val[2]`	`int *`	*x* + 8	
`val[5]`	`int`	??	
`*(val+1)`		`int`	5
`val + i`	`int *`	*x* + 4 *i*	

Example

```
int get_digit(int z[], int dig)
{
  return z[dig];
}

%edx = z
%eax = dig
movl (%edx,%eax,4),%eax  # z[dig]
```

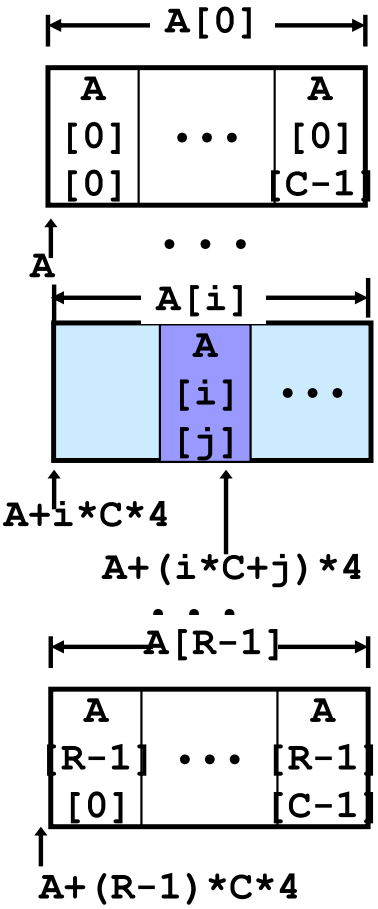
Array Loop

```
int loopint(int z[])
{
  int zi = 0;
  int *zend = z + 4;
  do {
    zi = 10 * zi + *z;
    z++;
  } while(z <= zend);
  return zi;
}
```

```
# %ecx = z   %eax = zi %ebx = zend
    xorl %eax,%eax      # zi = 0
    leal 16(%ecx),%ebx   #zend= z+4
.L:
    leal (%eax,%eax,4),%edx #5*zi
    movl (%ecx),%eax          #*z
    addl $4,%ecx              #z++
    leal (%eax,%edx,2),%eax
#zi = *z + 2*(5*zi)
    cmpl %ebx,%ecx          #z:zend
    jle .L        # if <= goto loop
```

10*zi + *z implemented as *z + 2*(zi+4*zi)

Nested Array (multidimentional)



- Array Elements
  - `A[i][j]` is element of type *T*
  - Address `A` + (*i* * *C* + *j*) * *K*

Declaration

T **`A[R][C];`**

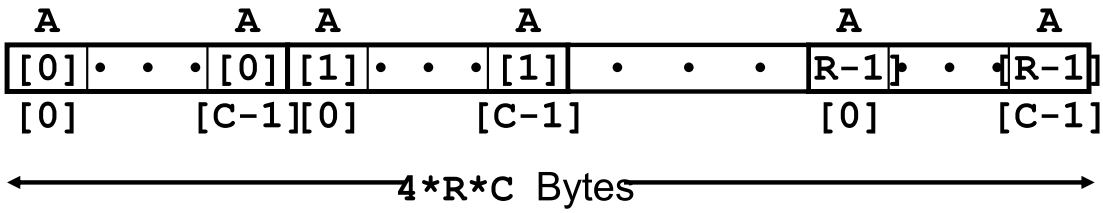
- **Array of data type *T***
- ***R* rows, *C* columns**
- **Type *T* element requires *K* bytes**

Array Size

- ***R* * *C* * *K* bytes**

Arrangement

- **Row-Major Ordering**



Nested Array element access

`int pgh[3][5]`

Array Elements

`pgh[index][dig]` is `int`

Address:

`pgh + 20*index + 4*dig`

```
int get_pgh_digit(int index, int dig)
{
  return pgh[index][dig];
}
```

```
  # %ecx = dig %eax = index
  leal 0(,%ecx,4),%edx          # 4*dig
  leal (%eax,%eax,4),%eax       # 5*index
  movl pgh(%edx,%eax,4),%eax    # *(pgh + 4*dig + 20*index)
```