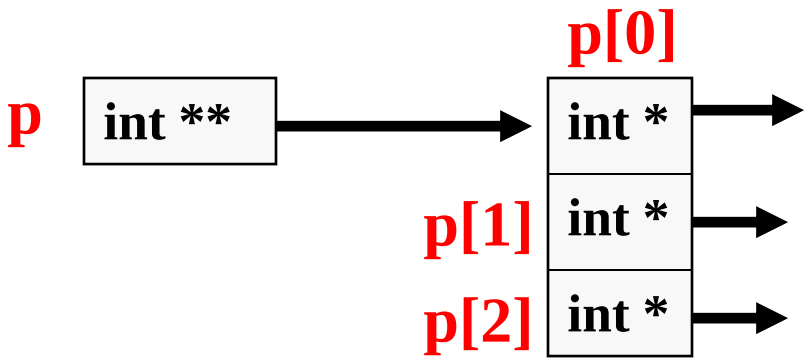
Pointers

Dynamic Memory Allocation of 2-D Arrays

Allocating Pointer to Pointer

```
int **p;
p = (int **) malloc(3 * sizeof(int *));
```



Dynamic Allocation of 2-d Arrays

```
int **allocate (int h, int w)
  {
    int **p;
    int i, j;

    p = (int **) malloc(h*sizeof (int *) );
    for (i=0;i<h;i++)
      p[i] = (int *) malloc(w * sizeof (int));
    return(p);
  }
```

Allocate array of pointers

Allocate array of integers for each row

```
void read_data (int **p, int h, int w)
  {
     int i, j;
     for (i=0;i<h;i++)
       for (j=0;j<w;j++)
         scanf ("%d", &p[i][j]);
  }
```

Elements accessed like 2-D array elements.

Contd.

```
void print_data (int **p, int h, int w)
 {
    int i, j;
     for (i=0;i<h;i++)
     {
     for (j=0;j<w;j++)
       printf ("%5d ", p[i][j]);
      printf ("\n");
     }
}
```

```
int main() {
      int **p;  int M, N;
      printf ("Give M and N \n");
      scanf ("%d%d", &M, &N);
       p = allocate (M, N);
      read_data (p, M, N);
      printf ("\nThe array read as \n");
       print_data (p, M, N);
      return 0;
}
```

Contd.

```
void print_data (int **p, int h, int w)
 {
   int i, j;
   for (i=0;i<h;i++)
    {
     for (j=0;j<w;j++)
        printf ("%5d ", p[i][j]);
     printf ("\n");
    }
}
```

```
Give M and N
3 3
1 2 3
4 5 6
7 8 9
 The array read as
   1    2    3
   4    5    6
   7    8    9
```

```
int main()
{
      int **p;
       int M,N;
      printf ("Give M and N \n");
      scanf ("%d%d", &M, &N);
       p = allocate (M, N);
       read_data (p, M, N);
      printf ("\nThe array read as \n");
      print_data (p, M, N);
      return 0;
}
```

Static array of pointers

```
#define  N   20
#define  M  10
int main()
{
    char word[N], *w[M];
    int i, n;
    scanf("%d",&n);
    for (i=0; i<n; ++i) {
        scanf("%s", word);
        w[i] = (char *) malloc ((strlen(word)+1)*sizeof(char));
        strcpy (w[i], word) ;
    }
    for (i=0; i<n; i++)
            printf("w[%d] = %s \n",i,w[i]);
    return 0;
}
```

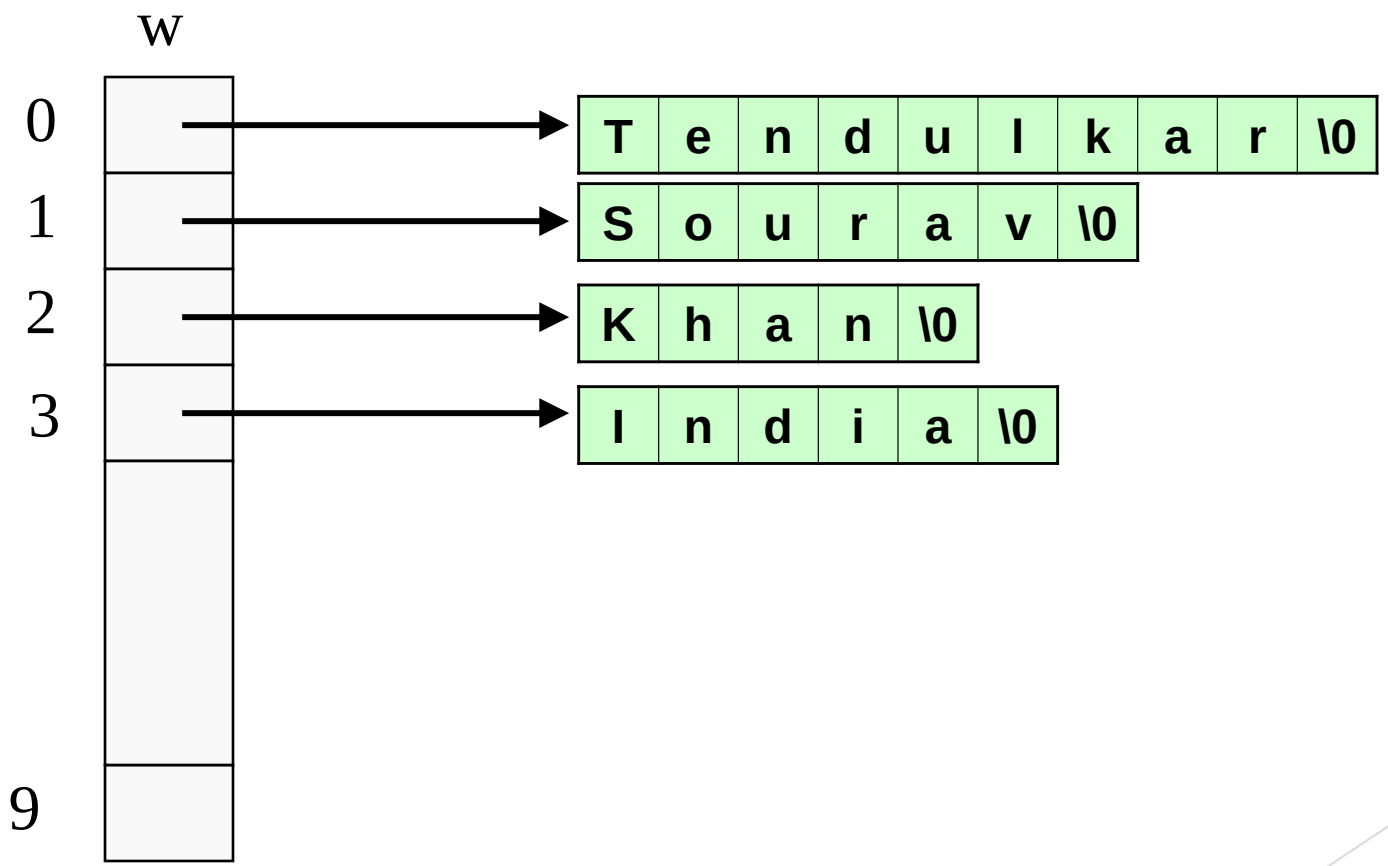
Static array of pointers

```
#define  N   20
#define  M  10
int main()
{
    char word[N], *w[M];
    int i, n;
    scanf("%d",&n);
    for (i=0; i<n; ++i) {
        scanf("%s", word);
        w[i] = (char *) malloc ((strlen(word)+1)*sizeof(char));
        strcpy (w[i], word) ;
    }
    for (i=0; i<n; i++) printf("w[%d] = %s \n",i,w[i]);
    return 0;
}
```

Output

```
4
Tendulkar
Sourav
Khan
India
w[0] = Tendulkar
w[1] = Sourav
w[2] = Khan
w[3] = India
```

How it will look like



Dynamic Arrays of pointers

```
int main()
{
  char word[20], **w;  /*  “**w” is a pointer to a pointer array */
  int i, n;
  scanf("%d",&n);
  w = (char **) malloc (n * sizeof(char *));
  for (i=0; i<n; ++i) {
     scanf("%s", word);
     w[i] = (char *) malloc ((strlen(word)+1)*sizeof(char));
     strcpy (w[i], word) ;
  }
  for (i=0; i<n; i++)
     printf("w[%d] = %s \n",i, w[i]);
  return 0;
}
```

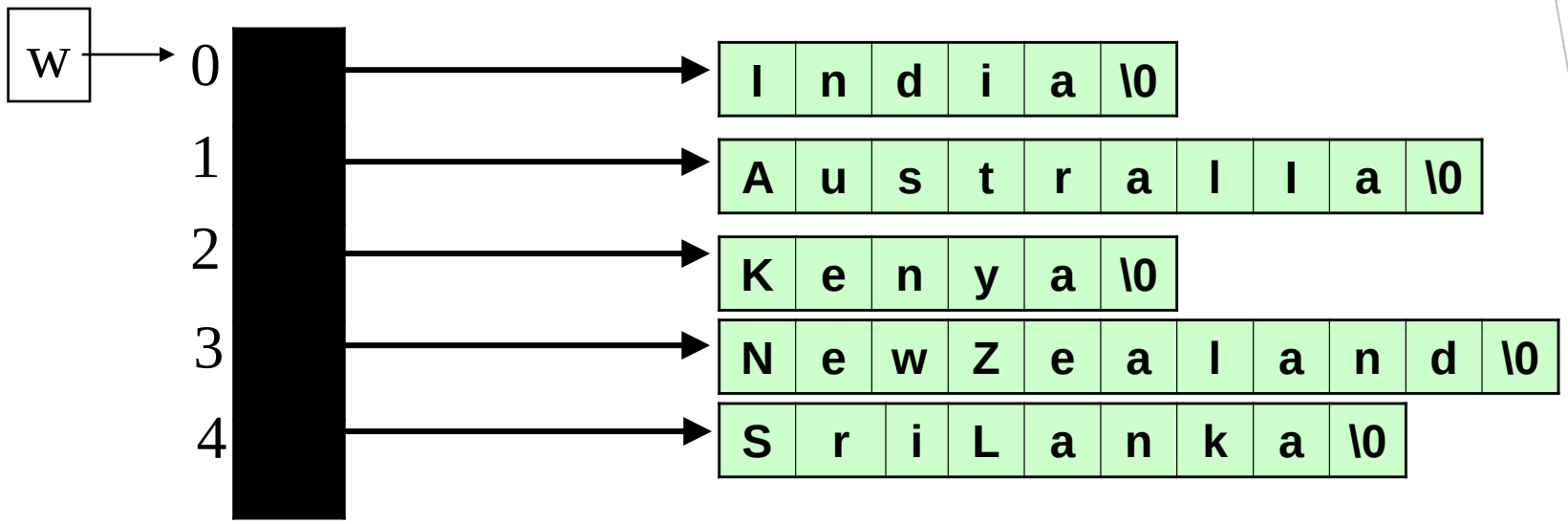
Dynamic Arrays of pointers

```
int main()
{
  char word[20], **w;  /*  "**w" is a pointer to a pointer array */
  int i, n;
  scanf("%d",&n);
  w = (char **) malloc (n * sizeof(char *));
  for (i=0; i<n; ++i) {
     scanf("%s", word);
     w[i] = (char *) malloc ((strlen(word)+1)*sizeof(char));
     strcpy (w[i], word) ;
  }
  for (i=0; i<n; i++)
        printf("w[%d] = %s \n",i, w[i]);
  return 0;
}
```

Output

```
5
India
Australia
Kenya
NewZealand
SriLanka
w[0] = India
w[1] = Australia
w[2] = Kenya
w[3] = NewZealand
w[4] = SriLanka
```

How this will look like



Homework

- Write a pointer version of a C program to compute the sum and product of two matrices. Use dynamic allocation for memory and use functions and pass the arrays to the read(), print(), sum(), product() functions.
- Sort an array of strings. Use dynamic array of pointers to allocate memory.