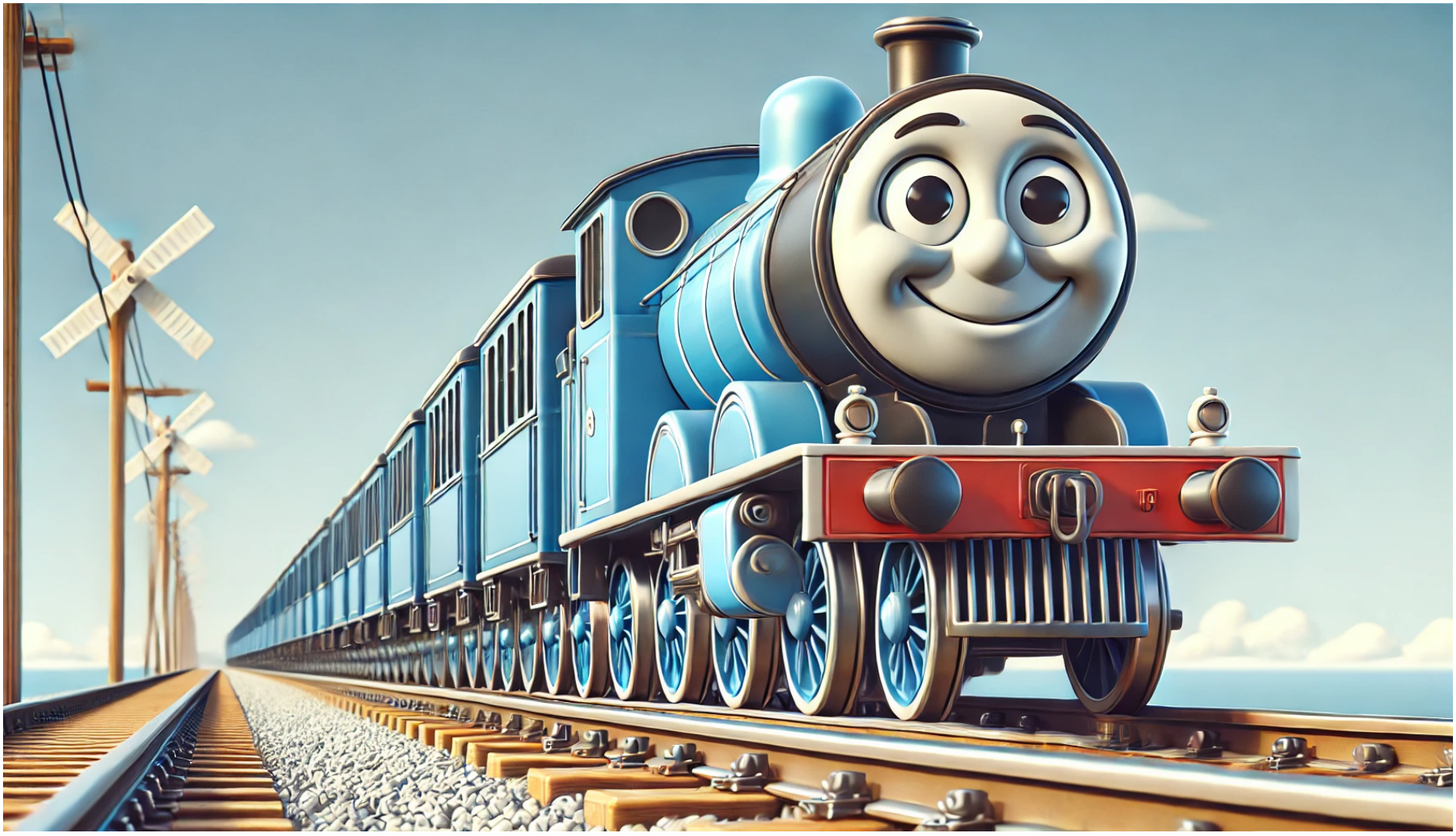
Programming and Data Structures - I

Lecture 10

Kripabandhu Ghosh

CDS, IISER Kolkata

ARRAYS



Arrays

Features

- A group of related (same data type) items sharing a common name (e.g. *marks*[10]: marks of 10 students)
- Contiguous memory allocation
- Each member accessed by an index (starts from 0)

One-dimensional Arrays

Format: declaration

type variable_name[size]

- **type**: data type of each element of the array (e.g. int, float etc.)
- **size**: number of elements in the array

Example

int marks[10];

One-dimensional Arrays

Format: access

marks[i]: the i^{th} element of the array *marks* ($0 \leq i <$ size)

marks[0], *marks*[1], ..., *marks*[i], ..., *marks*[size - 1]

One-dimensional Arrays: Initialization

```
int marks[5] = {1, 2, 3, 4, 5};
```

R.H.S number less than size

```
int marks[5] = {1, 2, 3}; //Remaining values will be set to zero
```

implicit size

```
int marks[5] = {1, 2, 3, 4, 5};
```

character array (string)

```
char name[] = {'R', 'a', 'm', '\0'};
char name[] = "Ram";
```

One-dimensional Arrays: Example

Program

```
#include<stdio.h>

void main()
{
   int num[] = {1, 5, 3, 45, 4}, size, i, max;
   size = sizeof(num)/sizeof(int);
   max = num[0];
   for(i = 1 ; i < size ; i++)
   {
      if(num[i] > max)
      {
         max = num[i];
      }
   }
   printf("Max = %d\n", max);
}
```

Output

```
Max = 45
```

Two-dimensional Arrays

Format: declaration

type variable_name[no_of_rows][no_of_cols]

- **type**: data type of each element of the array (e.g. int, float etc.)
- **no_of_rows**: number of row elements in the array
- **no_of_cols**: number of columns elements in the array

Example

int marks[10][5];

Two-dimensional Arrays

Format: access

marks[i][j]: the element at i^{th} row and j^{th} column of the array *marks* ($0 \leq i <$ no_of_rows, $0 < i <$ no_of_cols)

marks[0][0] *marks*[0][1] ... *marks*[0][j] ... *marks*[0][no_of_cols - 1]
marks[1][0] *marks*[1][1] ... *marks*[1][j] ... *marks*[1][no_of_cols - 1]
...
marks[i][0] *marks*[i][1] ... *marks*[i][j] ... *marks*[i][no_of_cols - 1]
...
marks[no_of_rows - 1][0] *marks*[no_of_rows - 1][1] ... *marks*[no_of_rows - 1][j] ... *marks*[no_of_rows - 1][no_of_cols - 1]

Two-dimensional Arrays: Initialization

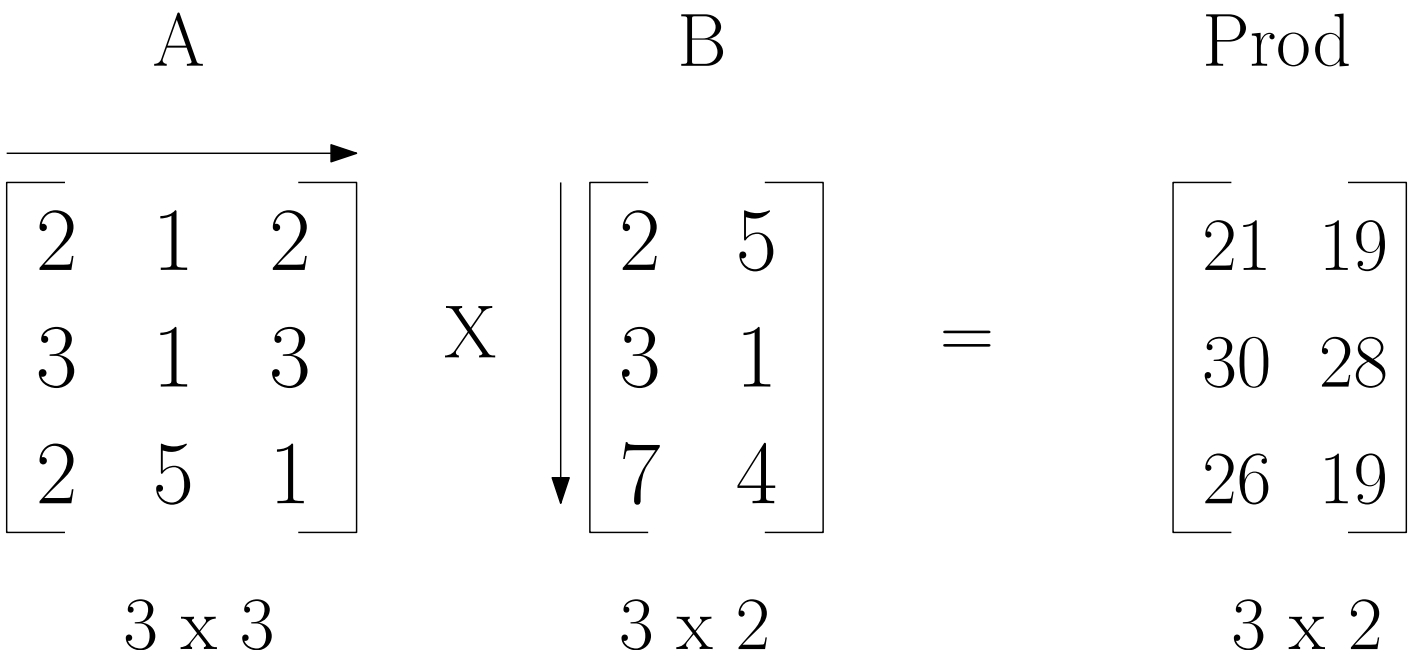
```
A[3][2] = {{2, 5}, {3, 1}, {7, 4}};
```

```
A[3][2] = {2, 5, 3, 1, 7, 4};
```

```
A[3][2] = {{2, 5},
           {3, 1},
           {7, 4}};
```

```
A[3][2] = {};
```

Matrix Multiplication



Matrix Multiplication: program

Program

```
#include<stdio.h>

void main()
{
  int A[3][3] = {{2, 1, 2}, {3, 1, 3}, {2, 5, 1}}, B[3][2] = {{2, 5}, {3, 1}, {7, 4}}, Prod[10][10] = {}, nr1 = 3, nc1 = 3, nr2 = 3, nc2 = 2, nr3,
nc3, i, j, k;
  if(nc1 != nr2)
  {
    printf("Invalid multiplication!\n");
  }
  else
  {
    for(i = 0 ; i < nr1 ; i++)
    {
      for(j = 0 ; j < nc2 ; j++)
      {
        for(k = 0 ; k < nc1 ; k++)
        {
          Prod[i][j] += A[i][k]*B[k][j];
        }
      }
    }
    printf("The product:\n");
    nr3 = nr1;
    nc3 = nc2;
    for(i = 0 ; i < nr1 ; i++)
    {
      for(j = 0 ; j < nc2 ; j++)
      {
        printf("%d ", Prod[i][j]);
      }
      printf("\n");
    }
  }
}
```

Matrix Multiplication: program (contd.)

Output

```
The product:
21 19
30 28
26 19
```

Multi-dimensional Arrays

Format

type variable_name[s1][s2][s3]...[sn]

- **type**: data type of each element of the array (e.g. int, float etc.)
- **si**: size of the i^{th} dimension

Example

int marks[10][5][20];
float chart[20][2][30[25];

STRINGS

Character Test Functions

Function	Operation
isalpha(c)	Is c an alphabetic character
isdigit(c)	Is c a digit?
isalnum(c)	Is c an alphanumeric character?
isspace(c)	Is c a space character?
islower(c)	Is c a lower case character?
isupper(c)	Is c an upper case character?
isxdigit(c)	Is c a hexadecimal character?
iscntrl(c)	Is c a control character?
isprint(c)	Is c a printable character?
ispunct(c)	Is c a punctuation?
tolower(c)	Converts c to lower case.
toupper(c)	Converts c to to upper case.

Example

Program

```
#include<stdio.h>
#include<ctype.h>

void main()
{
  char ch;
  printf ( "\nInput a character (Press enter at the end)\n" ) ;
  ch = getchar( );
  if(isalpha(ch))
  {
    printf("%c is an alphabet!\n", ch);
    if(islower(ch))
    {
      printf("%c is lower case!\n The upper case:\n", ch);
      putchar(toupper(ch));
    }
    else
    {
      printf("%c is upper case!\n The lower case:\n", ch);
      putchar(tolower(ch));
    }
  }
  else
  {
    printf("%c is not an alphabet!\n", ch);
  }
  printf("\n");
}
```

Example (contd.)

Output

```
Input a character (Press enter at the end)
K
K is an alphabet!
K is upper case!
The lower case:
k
```

String Functions

Function	Functionality
strcat(s1, s2)	concatenates s1 and s2 and stores in s1
strcmp(s1, s2)	compares s1 and s2 (0 if same)
strcpy(s1, s2)	copies s2 to s1
strlen(s1)	returns the number of characters in s1

String Functions: Examples

Program

```
#include<stdio.h>
#include<string.h>

void main()
{
   char s1[] = "kripa", s2[] = "bandhu", s3[] = "ghosh";
   //strcat
   strcat(s1, s2);
   printf("s1 = %s s2 = %s\n", s1, s2);
   //strcpy
   strcpy(s1, s2);
   printf("s1 = %s s2 = %s\n", s1, s2);
   //strcmp
   printf("%d\n", strcmp(s1, s3));
   //strlen
   printf("The length of %s is %zu\n", s3, strlen(s3));
}
```

Output

```
s1 = kripabandhu s2 = bandhu
s1 = bandhu s2 = bandhu
-5
The length of ghosh is 5
```


TRAIN
THANK YOU