Féidearthachtaí as Cuimse
Infinite Possibilities

Week 8 - Tutorial

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

Programming - Week 8

Overview

- Revision – Arrays
- 2 Dimensional Arrays
- Coding and Problem Solving

Symbolic Names (revision)

- In C, a symbolic name is used to **avoid hard-coding values** in a program.
- Normally use all **uppercase characters** for the names of symbolic names in order to **visually differentiate** them from regular variables in the program.
- A symbolic name looks like this:

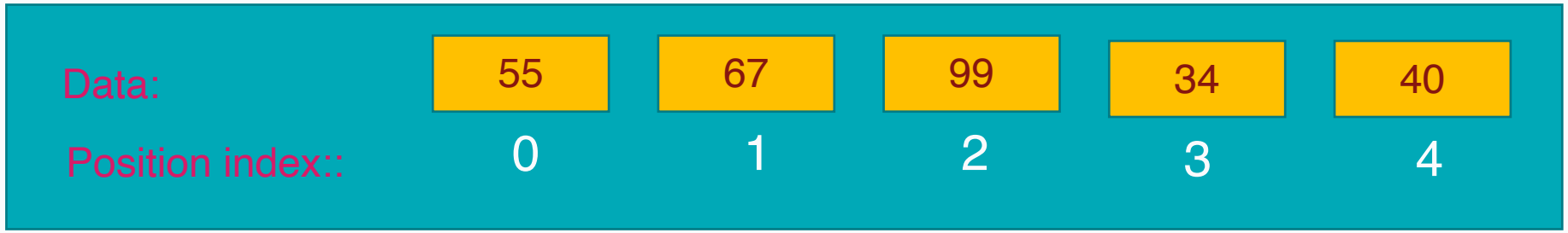
  #define symbolic_name data_value

- Eg.
  - #define SIZE 5
  - #define LETTER ‘A’

Working with an Array (revision)

- An array is a linear data structure.
- Data stored in the array are sequential (one after the other).
- The data at the start of the array is in position 0. This is the index of the array. We can use the index to access the data stored in the array.

int project_grades [5]



Initialising an array

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Initialize with Specific Values

- We can initialize an array by listing the values in curly braces {}
- int arr[5] = {1, 2, 3, 4, 5};

Partially Initialize

- We can also initialize only part of the array. Unspecified elements will default to 0.
- int arr[5] = {1, 2};

Automatic Size Deduction

- We can omit the array size if you initialize it with a set number of values.
- int arr[] = {1, 2, 3, 4, 5};

Program Example 1

- Create a program to fulfil the following requirements.
  - Ask a user to input the daily temperature recorded for the past 7 days (Monday to Sunday).
  - Once entered display the 7 temperatures on screen.
  - Ask the user to enter a number corresponding to the day they wish to view the temperature for. The number entered should only be between 1 and 7. This needs to be validated.
  - Display the temperature for the day the user requested.

Example 1 - Solution

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

program_1.c > main()

```c
#include <stdio.h>

#define SIZE 7

int main() {
    float temperatures[SIZE];
    int user_number;

    //Data entry from user
    for (int i = 0 ; i < SIZE ; i++) {
        printf("Enter Temperature %d >>", i+1);
        scanf(" %f", &temperatures[i]);
    }

    printf("\n\n");

    //Display data entered
    for ( int i = 0 ; i < SIZE ; i++) {
        printf("Temperature %d >> %f\n", i+1, temperatures[i]);
    }

    printf("\n\n");

    // Ask user for a number between 1 and 7
    do {
        printf("Enter a temperature day (1-7 >>)");
        scanf(" %d", &user_number);

        if (user_number < 1 || user_number > 7) {
            printf("Error... indalid day (1-7)\n\n");
        }
    } while( !(user_number >= 1 && user_number <= 7) );

    printf("For %d day the temp is %f\n", user_number, temperatures[user_number-1]);

    return 0;
}
```

Multidimensional Arrays

- A **multidimensional array** is essentially an array of arrays
- The most common type of multidimensional array is the **two-dimensional array**
- A two-dimensional array is useful for representing data in a grid or table.
- **You create a 2-D as follows:**

array_data_type array_name [no_of_rows] [no_of_columns];

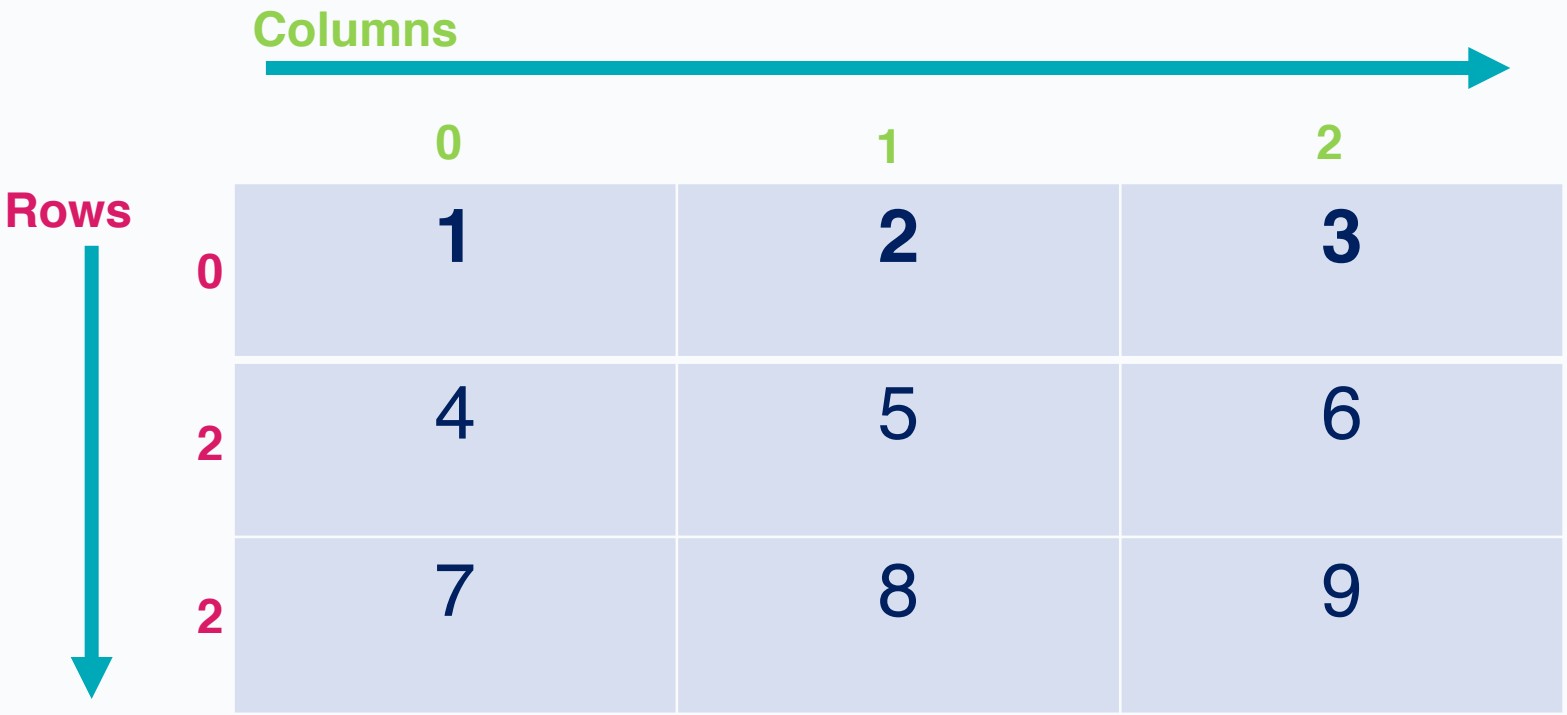
```
int matrix[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Two-Dimensional Arrays

- matrix is a 3x3 array (3 rows and 3 columns).
- Each row is an array
- Our matrix is an array of arrays.
- Data is accessed using the [row][column] notation
  - matrix[0][0]

```
int matrix[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Working with a 2D Array




```
int matrix[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

matrix[Rows][Columns]

Working with a 2D Array

What are the values for the following??:

- matrix[0][2]
- matrix[1][0]
- matrix[2][2]
- matrix[0][3]
- matrix[2][3]
- matrix[1][1]

```
int matrix[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Iterating through a 2D Array

```c
// Iterate through each element in the 2D array
for (int row = 0; row < 3; row++) {
    for (int col = 0; col < 3; col++) {
        // Access and print each element
        printf("Element at matrix[%d][%d] = %d\n", row, col, matrix[row][col]);
    }
}
```

```
Element at matrix[0][0] = 1
Element at matrix[0][1] = 2
Element at matrix[0][2] = 3
Element at matrix[1][0] = 4
Element at matrix[1][1] = 5
Element at matrix[1][2] = 6
Element at matrix[2][0] = 7
Element at matrix[2][1] = 8
Element at matrix[2][2] = 9
```

```c
int matrix[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Program Example 2

- Create a program to fulfil the following requirements for a supermarket. At the end of each week the part-time employees must enter the hours worked for Monday to Sunday. Eg. Monday (6hrs), Tuesday (4hrs) etc…
  - The company has 13 part-time employees
  - Use the sample employee data (next slide) to answer the following:
  - Print all the hours worked (Mon to Sun) for each employee
  - Print the total hours worked for each employee
  - What was the longest shift worked

Employee Data

- Copy the data and paste into your C program

```
int hours_worked[13][7] = {
{8, 4, 5, 3, 7, 0, 0}, // Worker 1
{6, 7, 8, 2, 4, 3, 0}, // Worker 2
{5, 5, 6, 4, 5, 0, 4}, // Worker 3
{7, 8, 6, 5, 4, 2, 0}, // Worker 4
{6, 3, 7, 8, 6, 0, 0}, // Worker 5
{4, 5, 4, 5, 6, 0, 6}, // Worker 6
{5, 7, 6, 10, 5, 1, 0}, // Worker 7
{6, 6, 8, 4, 7, 0, 0}, // Worker 8
{8, 7, 5, 3, 6, 2, 2}, // Worker 9
{7, 6, 4, 5, 8, 0, 0}, // Worker 10
{5, 5, 6, 5, 7, 0, 3}, // Worker 11
{4, 4, 7, 8, 6, 0, 0}, // Worker 12
{6, 8, 5, 4, 7, 1, 0} // Worker 13
};
```

Problem Solving

- Divide the large problem into smaller problems
- Slowly build the codebase and run to see each small change made
- Let’s code up problem 2 together following the above approach.

Hours worked

```
23      // Display the hours worked by each worker from Monday to Sunday
24      printf("Hours worked by each worker from Monday to Sunday:\n");
25      printf("          Mon  Tue  Wed  Thu  Fri  Sat  Sun\n");
26
27      for (int worker = 0; worker < 13; worker++) {
28          printf("Worker %2d: ", worker + 1);
29          for (int day = 0; day < 7; day++) {
30              printf("%4d", hours_worked[worker][day]);
31          }
32          printf("\n");
33      }
```

Total Hours Worked

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

```
35      // Total hours worked for each employee
36      printf("\nTotal hours worked\n");
37      for (int worker = 0; worker < 13; worker++) {
38          int total_hours = 0;
39          printf("Worker %2d: ", worker + 1);
40          for (int day = 0; day < 7; day++) {
41              total_hours = total_hours + hours_worked[worker][day];
42          }
43          printf("%d\n", total_hours);
44      }
```

Find the longest shift

```
44      int longest_shift = 0;
45      // Find the longest shift
46      printf("\nFind the longest shift\n");
47      for (int worker = 0; worker < 13; worker++) {
48          for (int day = 0; day < 7; day++) {
49              if( longest_shift < hours_worked[worker][day]) {
50                  longest_shift = hours_worked[worker][day];
51              }
52          }
53      }
54      printf("The longest shift was: %d\n", longest_shift);
55
```

Programming Pitfall

1. Array Index Out of Bounds
   - Accessing an array element outside its defined bounds
2. Misunderstanding Array Size
3. Off-By-One Errors
   - Miscalculate the array bounds by one, usually due to confusion between zero-based indexing and the length of the array

Questions