Féidearthachtaí as Cuimse
Infinite Possibilities

Week 6 - Tutorial

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

Programming - Week 6

Overview

- Arrays
- Working With Arrays
- Program Example 1
- Program Example 2

Arrays

- An array is a container that can hold multiple values.
- Most programming languages support arrays.
- In C an **array** is a collection of elements of the **same data type** stored in contiguous memory locations.
- An array is a way to substitute having to declare multiple variables **of the same data type**.
- Arrays allow us to group multiple variables of the same type together

Array Declaration

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

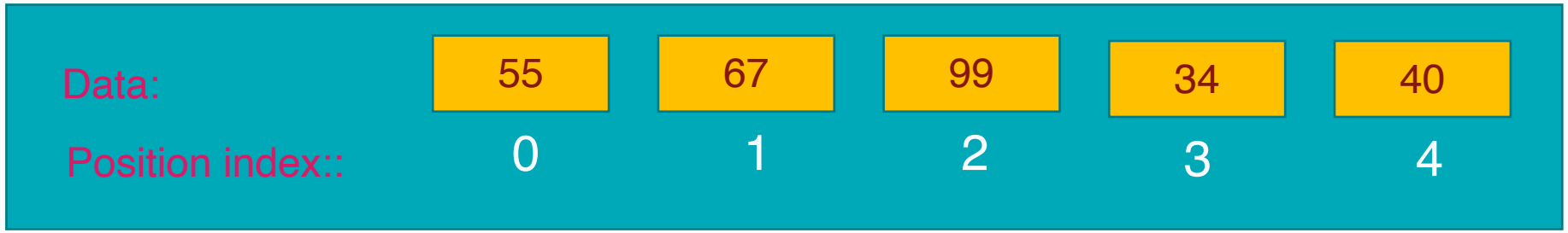
```
data_type array_name [array_size];
```

- Where:
  - data_type: The type of elements in the array (e.g., int, float, char).
  - array_name: The name of the array.
  - array_size: The number of elements the array can hold.

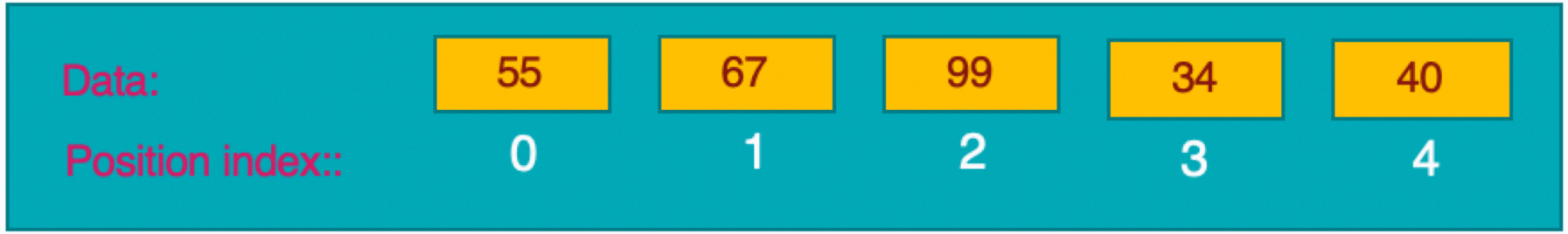
Working with an Array

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

int project_grades [5]



Working with an Array



This is an array of integer values.

We cannot mix data types, an int array can only store integer values.

The data is accessed via the position index.

The array starts with position 0

Example_1.c > ...

```c
#include <stdio.h>

int main() {
    int project_grades [5];

    project_grades[0] = 55;
    project_grades[1] = 67;
    project_grades[2] = 99;
    project_grades[3] = 34;
    project_grades[4] = 40;

    return 0;
}
```

Print an Array

Example_1.c > ...

```c
#include <stdio.h>

int main() {
    int project_grades [5];

    project_grades[0] = 55;
    project_grades[1] = 67;
    project_grades[2] = 99;
    project_grades[3] = 34;
    project_grades[4] = 40;

    for (int i=0; i<5; i++) {
        printf("Position %d stores a value of %d\n", i, project_grades[i]);
    }

    return 0;
}
```

- A for loop is used to loop through the contents of the integer array.
- The condition for the loop is i < 5

Store user input in an Array

Example_2.c > ...

```c
#include <stdio.h>

int main() {
    int project_grades [5];

    for (int i = 0; i < 5; i++) {
        printf("\nFor position %d please enter a grade: ", i);
        scanf("%d", &project_grades[i]);
    }

    for (int i = 0; i < 5; i++) {
        printf("Position %d stores a value of %d\n", i, project_grades[i]);
    }

    return 0;
}
```

- A for loop is used to loop 5 times.
- For each iteration we ask to user to input a grade for a given position in the array.
- The second for loop is used to print out the data stored in the array.

Program Example 1

- Create a program to fulfil the following requirements.
- Ask the user to input 10 grades (0 to 100).
  - Calculate the average grade.
  - Identify the highest grade
  - Identify the lowest grade

Example 1 - Solution

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

problem_1.c > main()

```c
#include <stdio.h>

int main() {
    int project_grades [10];
    int lowest_grade;
    int highest_grade;
    int overall_total;
    float average_grade;

    // Loop to get the user input
    for (int i = 0; i < 10; i++) {
        printf("\nFor position %d please enter a grade: ", i);
        scanf("%d", &project_grades[i]);
    }

    lowest_grade = project_grades[0];
    highest_grade = project_grades[0];

    // loop to get the sum total and identify the lowest / highest grade
    for (int i = 0; i < 10; i++) {
        overall_total = overall_total + project_grades[i];

        // check is the value in project_grades[i]
        // is greater than our highest_grade
        if (highest_grade < project_grades[i]) {
            highest_grade = project_grades[i];
        }

        // check is the value in project_grades[i]
        // is less than than our lowest_grade
        if (lowest_grade > project_grades[i]) {
            lowest_grade = project_grades[i];
        }
    } // end for loop

    average_grade = overall_total / 10;

    printf("\nAverage: %f", average_grade);
    printf("\nHighest grade: %d", highest_grade);
    printf("\nLowest grade: %d\n", lowest_grade);

    return 0;
}
```

Symbolic Names

- 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’

Program Example 2

- Create a program to fulfil the following requirements.
  - Generate a random number between 1 and 100
  - Tell the user they have 10 guesses.
  - Prompt the user to enter their guess
  - All guesses must be stored and displayed to the user at the end of the game
  - If they guess correct tell them they won
  - If the guess is incorrect they get another guess if they have used up their 10 guesses.

Example 2 - Solution

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

problem_2.c > main()

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 10


int main () {
    // generate a random number
    srand ( time(NULL) ); // Seeting the random generator
    int random_number = (rand() % 100) + 1;

    printf("The ramdom number is %d\n", random_number);
    int guesses[SIZE];
    int counter;

    printf("Number guessing game (1- 100)");

    for (counter = 0; counter < SIZE; counter++) {
        printf("\nEnter guess number %d: ", counter+1);
        scanf("%d", &guesses[counter]);

        if (guesses[counter] == random_number) {
            printf("CCorrect Guess (%d)!!\n", random_number);
            break; // exit the for loop
        } else {
            printf("Incorrect Guess (%d)", guesses[counter]);
        }
    } // end for loop

    if (counter == SIZE-1) { // counter starts at 0
        printf("\nSorry, you are out of guesses");
    }

    printf("\n\nYour guesses:\n");
    for (int j = 0; j < counter; j++) {
        printf("%d - ", guesses[j]);
    }
}
```

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