COMP0204: Introduction to Programming for Robotics and AI

Arrays and Strings

Course lead: Dr Sophia Bano

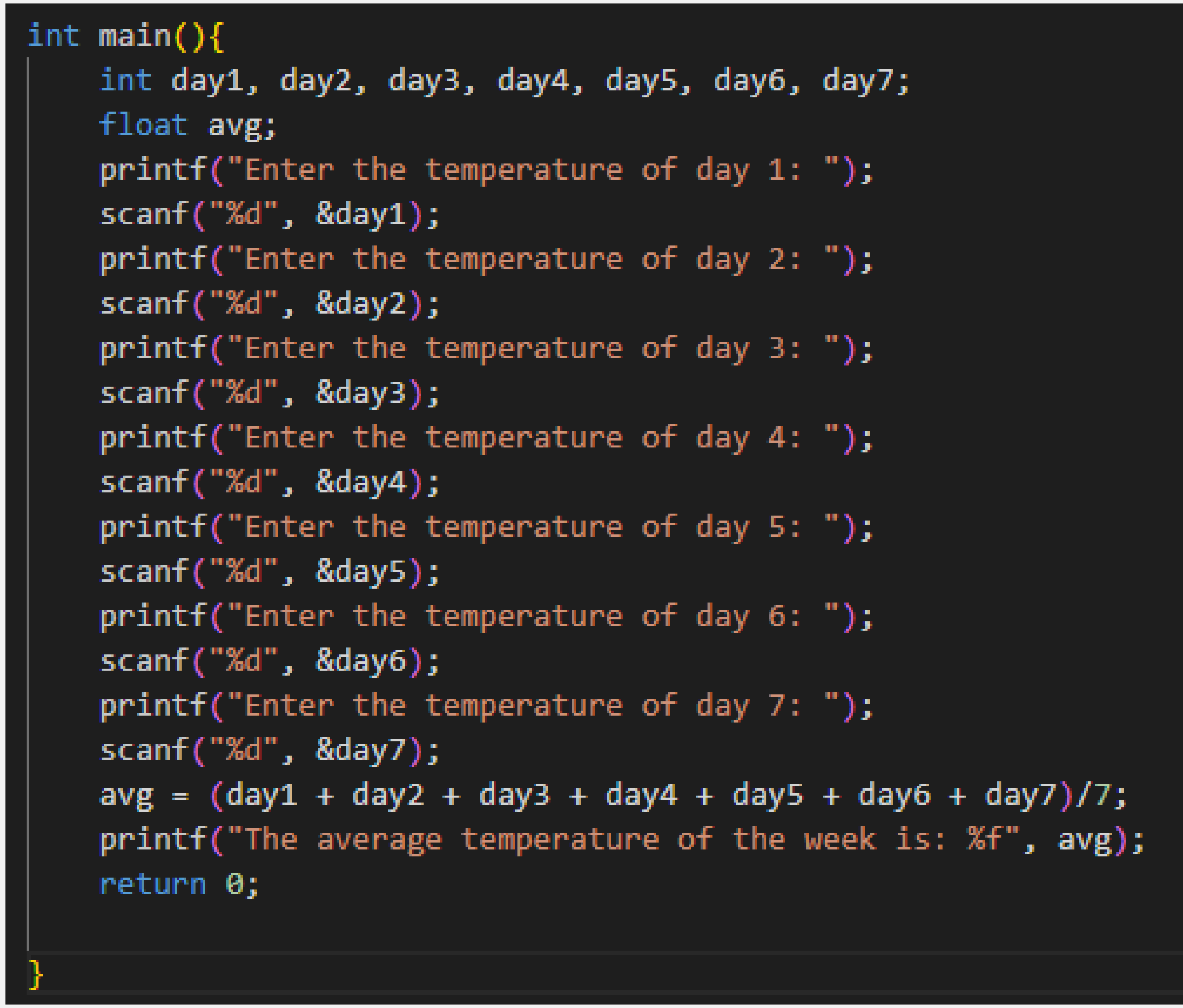
MEng Robotics and AI
UCL Computer Science

Problem with normal variables

Suppose if you wanted to find the **average temperature for a particular week**. If each day's temperature had a **unique value**.

Solution:

- Give a unique variable name to each day, you would end up by declaring 7 different variables.



```
int main(){
    int day1, day2, day3, day4, day5, day6, day7;
    float avg;
    printf("Enter the temperature of day 1: ");
    scanf("%d", &day1);
    printf("Enter the temperature of day 2: ");
    scanf("%d", &day2);
    printf("Enter the temperature of day 3: ");
    scanf("%d", &day3);
    printf("Enter the temperature of day 4: ");
    scanf("%d", &day4);
    printf("Enter the temperature of day 5: ");
    scanf("%d", &day5);
    printf("Enter the temperature of day 6: ");
    scanf("%d", &day6);
    printf("Enter the temperature of day 7: ");
    scanf("%d", &day7);
    avg = (day1 + day2 + day3 + day4 + day5 + day6 + day7)/7;
    printf("The average temperature of the week is: %f", avg);
    return 0;

}
```

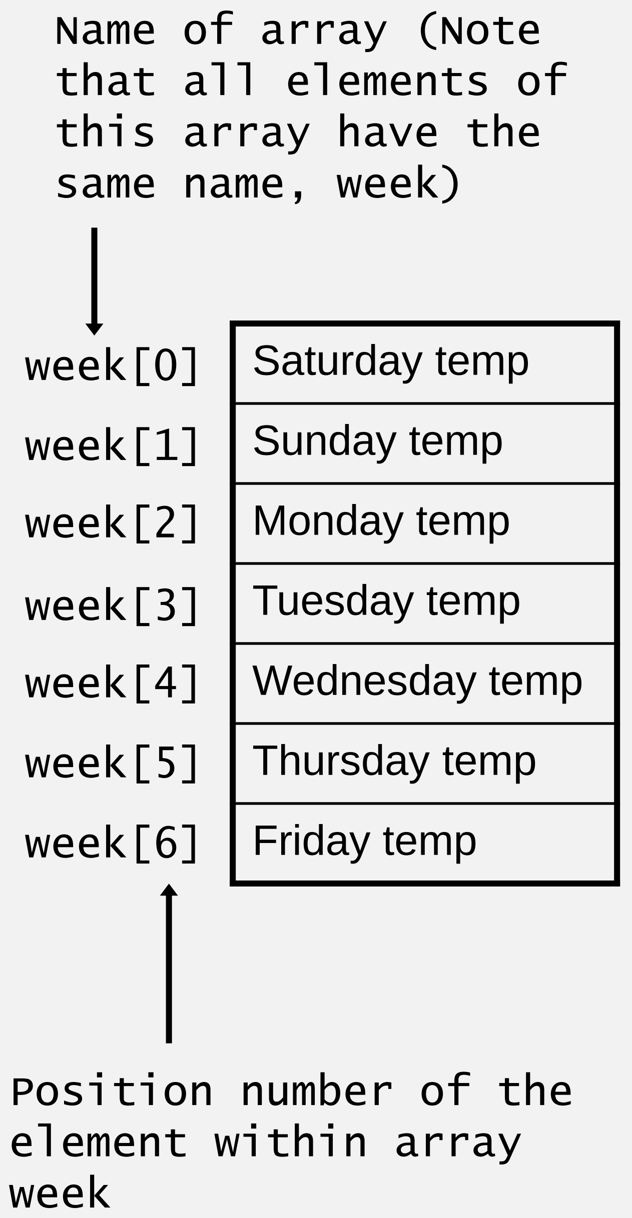
Arrays

- Collection of similar data types stored at contiguous memory locations.
- Provides a convenient way to store multiple values of the same data type under a single identifier.

Declaration:

```
datatype arrayName[arraySize];
int numbers[5]
```

- First element at position 0
- n element array named week: week[0], week[1]...week[n – 1]



Arrays

- Array elements are like normal variables

```
    week[0] = 3;
    printf("%d", week[0]);
```

Defining Arrays

- **When defining arrays, specify**
  - Name
  - Type of array
  - Number of elements

```
arrayType arrayName[numberOfElements];
```

  - **Examples:**

```
int week[10];
float myArray[3284];
```

- **Defining multiple arrays of same type**
  - Format similar to regular variables
  - Example:

```
int b[ 100 ], x[ 27 ];
```

Arrays declaration and initialization

- **Declaration**

  Arrays can be declared with or without initial values.

  ```
  int numbers[5];
  ```

- **Initialization**

  Arrays can be initialized at the time of declaration.

  ```
  int numbers[5] = {2, 4, 6, 8, 10};
  ```

Initializers

- **Initializers**

```
int numbers[5] = {2, 4, 6, 8, 10};
```

  - If not enough initializers, rightmost elements become 0

```
int n[ 5 ] = { 0 };
```

  - If too many a syntax error is produced

- **If size omitted, initializers determine it**

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

  - 5 initializers, therefore 5 element array

Referring to individual elements of a Array

- Refer the individual elements of an array using **subscripts** (the number in brackets following the array name).
- The number specifies the element’s position in the array.
- All the array elements are numbered, starting at 0 to size.
  - **Example:**

```
printf(“Day %d temperature is %d\n", 2, week[2]);
```

week[0]	Saturday temp
week[1]	Sunday temp
week[2]	Monday temp
week[3]	Tuesday temp
week[4]	Wednesday temp
week[5]	Thursday temp
week[6]	Friday temp

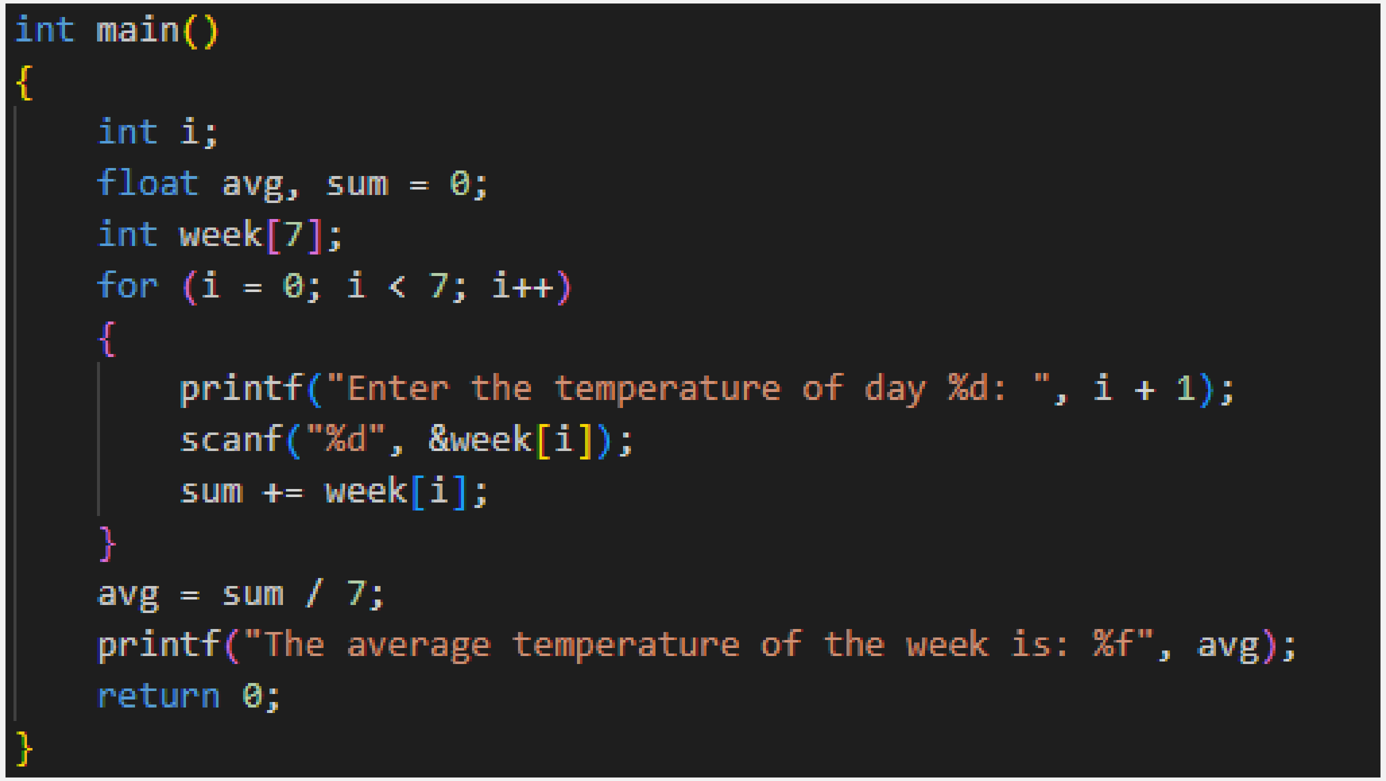
Entering Data into the Array

- Example:

```
for (i = 0; i < 7; i++)
    {
        printf("Enter the temperature of day %d: ", i + 1);
        scanf("%d", &week[i]);
    }
```

Reading Data from the Array

- To calculate the average of the week’s temperature using array (here is the code).



```c
int main()
{
    int i;
    float avg, sum = 0;
    int week[7];
    for (i = 0; i < 7; i++)
    {
        printf("Enter the temperature of day %d: ", i + 1);
        scanf("%d", &week[i]);
        sum += week[i];
    }
    avg = sum / 7;
    printf("The average temperature of the week is: %f", avg);
    return 0;
}
```

Multi-dimensional Arrays

- Multi-dimensional arrays are arrays that contain arrays as their elements. They are useful for representing data in multiple dimensions, such as matrices or tables.
- Declaration:

```
datatype arrayName[size1][size2]; // 2D array
```

Practice exercise: Array

Exercise

Write a C program that takes 5 integers as input from the user and stores them in an array. Find and print the second largest element in the array.

Strings

- One dimensional arrays is used to create character strings.
- A string is defined as a character array that is terminated by a null.
- A null is specified using ‘\0’ and is zero.
- A null is appeared at the end of every string.
- Because of the null terminator, it is necessary to declare a string to be one character longer.

Strings

- Arrays of characters.
- All strings end with **`null`** (`'\0'`).
- Examples
  - **`char string1[] = "hello";`**
  - **`Null`** character implicitly added
  - **`string1`** has 6 elements
  - **`char string1[] = { 'h', 'e', 'l', 'l',      'o', '\0′ };`**
- Subscripting is the same
  - **`string1[ 0 ]`** is **`'h‵`**
  - **`string1[ 2 ]`** is **`'l'`**

Reading strings from keyboard

- The easiest way to read a string entered from the keyboard is to make a character array

```
int main(){
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    printf("The string you entered is: %s\n", str);
    return 0;
}
```

String

- Input from keyboard

```
char string2[10];
scanf("%s", string2);
```

  - Puts user input in string
    - Stops at first whitespace character
    - Adds **`null`** character
  - If too much text entered, data written beyond array
  - Printing strings
  - **`printf ("%s", string2);`**
  - Characters printed until **`null`** found

Reading strings from keyboard

There is a problem with previous program, if the string has whitespace characters

```
// Using gets() to read a string from the keyboard.
int main(){
    char str[100];
    printf("Enter a string: ");
    gets(str);
    printf("The string you entered is:”);
    puts(str);
    return 0;
}
```

Reading strings from keyboard

- Remember scanf, gets() do not perform **any bounds checking** on the array.
- Use fgets() when input needs to be bounded.

```
fgets(str, 5, stdin);
```

Accessing Individual Elements of an string

- We can access the individual elements of a string using subscript:

```
char str1[]= "Hello world";
   for (int i = 0; i < 11; i++){
       printf("%c,", str1[i]);
   }
```

Output:

```
   H,e,l,l,o, ,w,o,r,l,d,
```

Practice exercise: Array and String

Exercise

Write a C program that takes an integer input between 1 and 10 from the user and converts it into its corresponding word form. For example, if the user inputs 5, the program should print "Five". Ensure that the program displays an appropriate message if the user enters a number outside the specified range.