Week 5

Hari Sethuraman

By the end of today

You should have a solid understanding of

- Loops in C
  - While
- Arrays in C
  - What is an array?
  - How are arrays represented in memory?
- What an Algorithm is (a brief introduction)
- What ‘Runtime’ is (a brief introduction)

Recap

- There are three types of operators
  - Arithmetic operators (operations related to numbers: +, -, *, /, %)
  - Relational operators (==, >= ,<=, != …)
  - Logical operators (&&, ||)
- If conditions bring Logic into a program
  - If (a condition is true), then execute {the code between the curly braces}
- Else is used after an if or else if condition:
  - If __ is true, then execute __. Else, execute __
- Else if is used if you want to stack multiple ifs and only want 1 to execute
  - Always comes after an if
  - All ifs, else ifs and else’s must have Mutually exclusive conditions and code

Loops

- Loops are essentially an if condition that repeats itself:
  - while (a condition is true), repeat {the code in the curly braces}
  - One cycle of a loop is called an ‘Iteration’

```
while (    )
{

}
```

Enter condition here

Enter code here

Program

- Task: Print all integers from 1 through 20 each in a new line

Method 1

Manually print all numbers

```
printf("1\n");
printf("2\n");
…
printf("19\n");
printf("20\n");
```

Bad code

- Always avoid repeating lines in programming as it makes the code less aesthetic, and it makes debugging MUCH harder

Method 2

Use a loop to print all numbers

```
int i = 1;
while (i <= 20)
{
    printf("%d\n", i);
    i = i + 1;
}
```

The iterator variable: a variable that changes its value after each iteration

Incrementing *i* after each iteration

Incrementing shortcuts

- i = i + 1;
- i += 1;
- i++;

All mean the same thing
(increase i's value by 1)

Arrays

- Arrays are a data structure that can store multiple variables of the same data type
  - An array of characters, integers, etc…
- They are an ordered list of elements
- To access the elements of an array, we need to provide the element's 'Index'
  - The Index starts counting from 0
    - To access the first element, you would need to pass in *0* as the index
    - To access the *n*th element, you would need to pass in (n-1) as the index
  - Use the name of the array followed by square brackets
    - *a[0]*

Arrays (continuation)

- Declaring an array is like declaring a regular variable
  - `char a[] = {'a', 'b', 'c', ... , 'x', 'y', 'z'};`

    Datatype name

    The elements in the array
  - If you don't know the elements, but know the length:
    - `char a[26];`
- When declaring an array, you need to know either the length or the elements in it.
- Suppose we had an array *a* which stores all the alphabets (in lower case)
  - `char a[] = {'a', 'b', 'c', ... , 'x', 'y', 'z'};`
  - `Printf("%c\n", a[5]);` what would this print?

Program

- Print all elements in *a*
- Use a while loop

```
int i = 0;
while (i < 26)
{
    printf("%c\n", a[i]);
    i++;
}
```

Getting elements into an array

- To get elements into an array created by the user, you need to know the length *l* of the array.
  - Ask this first through a scanf
- Then, create an array length *l*
- Using a loop, ask the individual elements through a scanf

Program

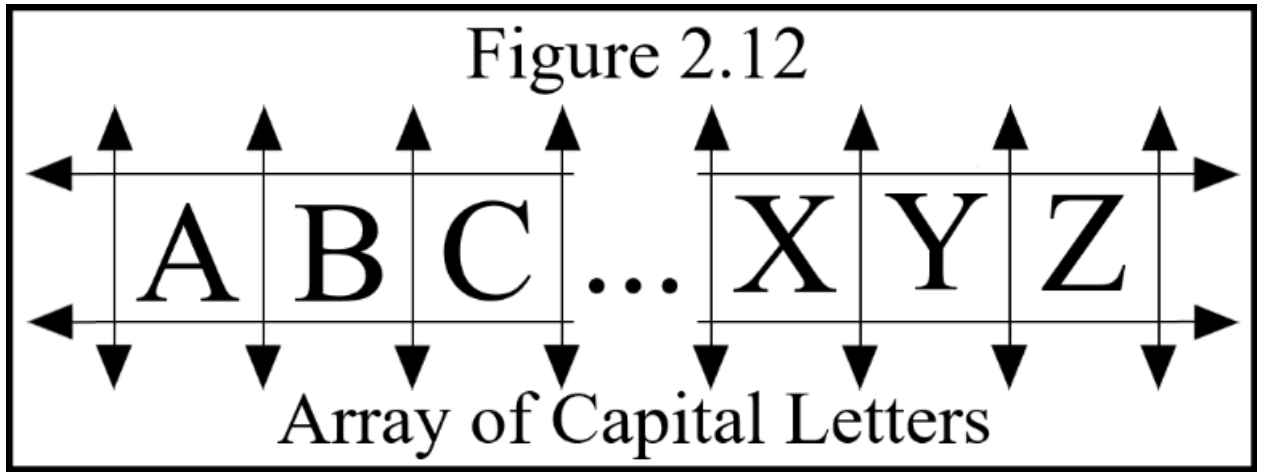
```
int l = 0;
scanf("%d", &l);
int nums [l];

int i = 0;
while (i < l)
{
    scanf("%d", &nums[i]);
    i++;
}

i = 0;
while (i < l)
{
    printf("%d, ", nums[i]);
    i++;
}
```

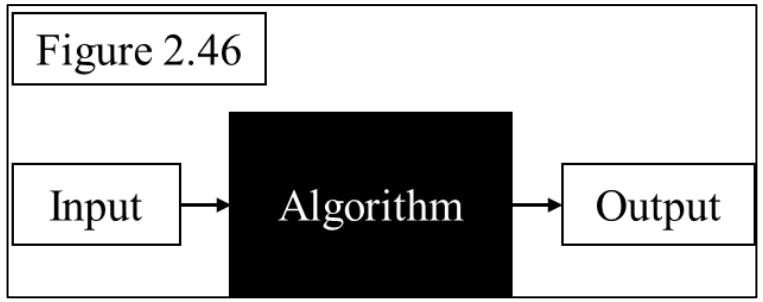
How are arrays represented in memory

- If you remember, the memory is like a large grid of bytes
- An array stores its elements consecutively in memory
- When you declare an array, the program allocates (or reserves) a certain amount of memory for the array
  - How much? The number of elements * the size of an element



Algorithms

- An Algorithm is a series of sequential steps that solve a certain problem or type of problem.
- It is like a box, which takes in an input, and gives out an output



- For any problem, there are multiple algorithms that will work
  - Depending on the design of the algorithms, we classify it into a specific family

Families of Algorithms

- Brute Force
- Greedy
- Dynamic Programming
- Divide and Conquer
- Randomized
- …

Runtime

- For any given problem, there are multiple algorithms that will work
- Some are better than others
- To describe how fast a program will run, we use Runtime
- The ‘Runtime’ of an algorithm is a function describing how efficient an algorithm is in relation to the input size *n*.
- We represent Runtime using a notation called ‘Asymptotic Notation’
- There are three main ways to show a runtime
  - Worst case: Big – O
  - Best Case: Big – Omega
  - Best Case + Worst Case: Big – Theta

Homework

- Watch this video on For loops: https://youtu.be/Qn8dNgvqPoo
- Create a simple program:
  - The user passes in a number *n:* the number of elements in the array
  - The user then passes the *n* elements (1 per line)
  - Print out the sum of all elements in the array
  - Print out the largest number in the next line