Unit - 1

Chapter

2 Fundamental Algorithms

Chapter Outline

- Fundamental Algorithms
- Exchanging The Values of Two Variables
- Counting
- Summation of a Set of Numbers
- Factorial Computation
- Generating of the Fibonacci Sequence
- Reversing the Digits of an Integer
- Character to Number Conversion
- Review Questions

2.1 Fundamental Algorithms

In any program or algorithm, there are very basic and small number of basic instructions whic
are used to perform computations. At the most fundamental level a computer has instructions fo
storing information. for performing arithmetic, and comparing information. There are also instruc
ions for controlling the way a computation is performed . Let us discuss few basic and fundamenta
algorithms in this section which are used to build complex algorithms and programs.

IMPORTANT NOTE

We assume that the students have not learnt programming in C so far and hence we will limit this chapter to only algorithms. The programs for these algorithms will be discussed in UNIT 2.

2.2 Exchanging The Values of Two Variables

Problem

Given two variables, a and b, exchange the values assigned to them.

Exchanging the values of two numbers means swapping the values of two variables with each other. For example, variable1 contains 20 and variable2 contains 40 after exchange the variable1 contains 40 and variable 2 contains 20. The problem of interchanging the values associated with two variables involves a very fundamental mechanism that occurs in many sorting and data manipulation algorithms.

Example **Exchanging the values of two variables**

Let us consider two variables a =10 and b =20

Before Swapping : a = 10 and b = 20.

After Swapping : a = 20 and b = 10.

We can use the assignment operator '=' to assign the value to a variable as shown below;

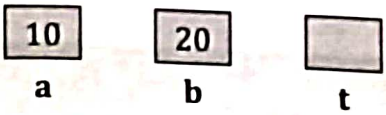
a = b

b = a

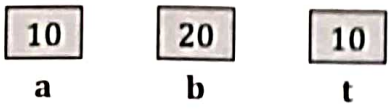
The assignment (a = b) has changed the value of a to 20 and the assignment (b = a) has changed the value of b to 20. Since the value of a is 20 after assignment (a = b) and its value is stored in b.

We can not solve the problem by assigning the value of 'a' to 'b' and 'b' to 'a'. It is because once the value of a is assigned to b, then original value of a is overwritten and will be lost.

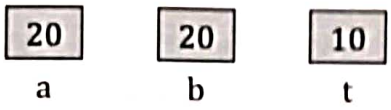
Let us use temporary variable 't' to solve this problem.



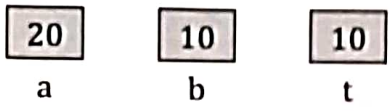
First Assign the value of 'a' to 't' (a = t)



Assign the value of 'b' to 'a' (a = b)



Assign the value of 't' to 'b' (b = t)



Finally, the value of a and b are swapped using temporary variable 't'

Algorithm **Exchanging the values of two numbers using temporary variable**

```
Step 1: Start
Step 2: Declare three variables a ,b and t
Step 3: Input First Number 'a'
Step 4: Input Second Number 'b'
Step 5: Set  t = a
Step 6: Set  a = b
Step 7: Set  b= t
Step 8: Print a and b
Step 9: End
```

We can swap two numbers without using temporary variable. By using + and - operator we can swap two number.

Example **Exchanging the values of two variables without using temporary variable.**

Suppose we are given 2 numbers a = 10 and b = 20 and we have to exchange these 2 numbers without using temporary variable. Then we have to perform below operations to swap the numbers.

Step 1 : Firstly we will add both the number and will assign to the first number.

a = a + b =10 + 20

a = 30

Step 2: Secondly we will subtract second number from the first number and will assign to the second number (updated value after step 1)

b = a - b = 30 - 20

b = 10

Step 3: Third step will be subtracting second number (updated value after step 2) from the first number (updated value from step 1) and will assign to the first number

a = a - b = 30 - 10

a = 20

After the above 3 steps we will get our required result i.e we have exchanged 2 numbers without using temporary variable.

Algorithm **Exchanging the values of two numbers without using temporary variable**

```
Step 1: Start
Step 2: Declare two variables a and b
Step 3: Input First Number 'a'
Step 4: Input Second Number 'b'
Step 5: Set  a = a + b
Step 6: Set  b = a - b
Step 7: Set  a = a - b
Step 8: Print a and b
Step 9: End
```

2.3 Counting

Problem

Given a set of 'n' students' examination marks (in the range 0 to l 00) make a count of the number of students that passed the examination. A pass is awarded for all marks of 50 and above.

Counting techniques are very frequently used in computer algorithms. Generally a count must be made on the number of items in a set which possess some particular property or which satisfy some particular condition or conditions. In the above problem, count must be made when marks of the student is 50 or above.

Example **Counting Example**

Let us consider the marks of 7 students.

First step is to read the marks of 7 students

Here, n=7 and marks = {75, 86, 45, 95, 37, 60, 50}

To get the count of the passes for this set we can start at the left , examine the first marks (i.e. 75), see if it is greater than or equal to 50, since 75>= 50 is true, count value is incremented by 1. The count value is 1.

The second marks (86) is then examined and since 86 >= 50 is true, count value is incremented by 1. The count value is 2.

The third marks (45) is then examined and no change in count value as 45 is less than 50. The count value is 2

The process continues in a similar manner until all marks have been tested.

We need another variable 'i' to iterate marks one by one. The value of 'i' is initialized to 1 and we continue reading as long as 'i' is less than or equal to 'n'.

Iteration	Marks	Condition	Count Value
i=1 (i<=7 is true)	75	75>=50 is true, increment count value	Count = 1
i=2 (i<=7 is true)	86	86>=50 is true, increment count value	Count = 2
i=3 (i<=7 is true)	45	45>=50 is false, no change in count value	Count = 2
i=4 (i<=7 is true)	95	95>=50 is true, increment count value	Count = 3

i=5 (i<=7 is true)	37	37>=50 is false, no change in count value	Count = 3
i=6 (i<=7 is true)	60	60>=50 is true, increment count value	Count = 4
i=7 (i<=7 is true)	50	50>=50 is true, increment count value	Count = 5
i=8 (i<=7 is false)	We stop reading the marks as all the marks are examined. The final count value is 5		

Total number of students who passed the examination = 5

Algorithm **Counting the number of students who scored 50 and above marks in examination**

```
Step 1: Start
Step 2: Declare variables n, count and i
Step 3: Read 'n' marks to be processed.
Step 4: Initialize counter to zero.
        Set count=0
Step 5: Initialize i to 1
        Set i=1
Step 6: while i is less than or equal to n, do the following steps
           (a) Read ith marks
           (b) if marks is greater than or equal to 50 then increment count by 1
           (c) increment i value by 1.
Step 7: Print count
Step 8: End
```

2.4 Summation of a Set of Numbers

Problem

Given a set of n numbers design an algorithm that adds these numbers and display the resultant sum . Assume 'n' is greater than or equal to zero.

Summation means adding the number. We have set of 'n' numbers and task is to add all these numbers. The sum should be returned from the algorithm.

sum = $a_1+a_2+a_3+.....+ a_n$

Computer can add two numbers at a time and return the sum of two numbers. So for this purpose we assign the variable sum as zero.

sum= sum+a_1 (Here a_1 is the first number) then we do

sum=sum+a_2 (Here the new value of sum contains a_1+a_2) then

sum=sum+a_3 (Here the new value of sum contains $a_1+a_2+a_3$) and so on

sum=sum+a_n (Here the new value of sum contains $a_1+a_2+a_3+.....+ a_n$)

Example

Let us consider numbers are {10,20,30,40,50}

Here, we have 5 numbers (n=5)

We will use a variable 'sum' to store the sum and it's initial value is 0. (sum=0)

We also need variable 'i' to iterate 'n' times. Initially the variable 'i' is assigned with 1. (i=1)

Expected output is : 10 + 20 + 30 + 40 + 50 = 150

We need to read each element until 'i' become 'n'. If 'i' is less than 'n' then first number is added to variable 'sum' and the value of 'i' is increased by 1. In each step, i^{th} number is added to 'sum' and 'i' is increased by 1. When the value of 'i' reaches 6, the condition (i <=5) becomes false. The sum value in the last step is the result of sum of 'n' numbers.

i = 1, sum = 0, n=5

Step	while (i < = n)	sum = sum+i^{th} number	increment i by 1
1	while (1< = 5) = True	sum = 0+10= 10	i=2
2	while (2 < = 5) = True	sum = 10+20 = 30	i=3
3	while (3 < = 5) = True	sum = 30+30 = 60	i=4
4	while (4< = 5) = True	sum = 60+40 = 100	i=5
5	while (5 < = 5) = True	sum = 100+50 = 150	i=6
6	while (6 < = 5) = False		

The sum of numbers = 150

Algorithm **Summation of 'n' numbers**

Step 1: Start

Step 2: Declare variables n, sum, i

Step 3: Read 'n' values that need to be summed up.

Step 4: Initialize sum to zero.

Set sum=0

Step 5: Initialize i to 1

Set i=1

Step 6: while i is less than or equal to n, do the following steps

(a) Read i^{th} number

(b) Add i^{th} number to the current value of sum (sum = sum + i^{th} number)

(c) increment i value by 1.

Step 7: Print the value of sum

Step 8: End

2.5 Factorial Computation

Problem

Given a number n , compute n factorial (written as n!) where n >= 0.

Factorial of a whole number 'n' is defined as the product of that number with every whole number till 1. For example, the factorial of 4 is 4×3×2×1, which is equal to 24. It is represented using the symbol '!' So, 24 is the value of 4! The study of factorials is at the root of several topics in mathematics, such as the number theory, algebra, geometry, probability, statistics, graph theory, and discrete mathematics, etc.

"n factorial" means: n! = 1 × 2 × 3 .....× n (or) n! = n × (n-1) × (n-2).....× 3 × 2 × 1

Note: 0! = 1 and 1! =1

Example

Observe the numbers and their factorial values given in the following table. To find the factorial of a number, multiply the number with the factorial value of the previous number. For example, to know the value of 5! multiply 24 (the factorial of 5) by 5, and we get 120. For 6! multiply 120 (the factorial value of 5) by 6, to get 720.

n	n!		
1	1	1	1
2	2 × 1	= 2 × 1!	= 2
3	3 × 2 × 1	= 3 × 2 !	= 6
4	4 × 3 × 2 × 1	= 4 × 3 !	= 24
5	5 × 4 × 3 × 2 × 1	= 5 × 4 !	= 120

We know that factorial of a number means product from 1 to that number. The variable **'fact'** is initialized to 1. For each iteration of **while** loop the **fact** is multiplied with **n** and 'n' value gets decremented by 1 at each iteration. When 'n' value becomes '0', then loop terminates and prints the factorial of given number.

Initially fact = 1 and Assume n = 4

Iteration	while condition (n > = 1)	fact = fact * n	n = n - 1
1	while (4 >= 1) = True	fact = 1 * 4 = 4	n=3
2	while (3 >= 1) = True	fact = 4 * 3 = 12	n=2
3	while (2 >= 1) = True	fact = 12 * 2 = 24	n=1
4	while (1 >= 1) = True	fact = 24 * 1 = 24	n=0
5	while (0 >= 1) = False	- -	- -

Factorial of 4 = 24

Algorithm Factorial of a Number

```
Step 1: Start
Step 2: Declare variables n, fact.
Step 3: Read a number n.
Step 4: Initialize fact to one.
        Set fact=1
Step 5: while n is greater than equal to 1, do the following steps
            (a) Calculate  fact = fact * n
            (b) decrement n value by 1.
Step 6: Print the value of fact
Step 7: End
```

2.6 Generating the Fibonacci Sequence

Problem

Generate and print the first n terms of the Fibonacci sequence where n>=1.

The first few terms are:

0, 1, 1, 2, 3, 5, 8, 13, ...

Each term beyond the first two is derived from the sum of its two nearest predecessors.

Fibonacci numbers are special kinds of numbers that form a special sequence. This sequence is one of the famous formulas in mathematics.

Fibonacci numbers are a sequence of whole numbers arranged as 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,...

Fibonacci's sequence is characterized by the fact that every number after the first two is the sum of the two preceding ones. For example, first 2 numbers (0, 1) are starting numbers of Fibonacci series and it has no preceding numbers. We can directly print the first two numbers. But after that, i.e the 3rd number (1) is the sum of 1st and 2nd number (0+1=1). Similarly to get 4th number, we add 2nd and 3rd number. (i.e., 1+1=2). We can use this pattern to find Fibonacci series up to any number.

Mathematical expression to find Fibonacci number is : $F_n = F_{n-1} + F_{n-2}$ with base values $F_0 = 0$ and $F_1 = 1$

n	Term	$F_{(n-1)}$	$F_{(n-2)}$	$F_n = F_{(n-1)} + F_{(n-2)}$ (for n >1)
0	First	-	-	$F_0 = 0$
1	Second	$F_0 = 0$	-	$F_1 = 1$
2	Third	$F_1 = 1$	$F_0 = 0$	$F_2 = 0 + 1 = 1$
3	Fourth	$F_2 = 1$	$F_1 = 1$	$F_3 = 1 + 1 = 2$
4	Fifth	$F_3 = 2$	$F_2 = 1$	$F_4 = 2 + 1 = 3$
5	Sixth	$F_4 = 3$	$F_3 = 2$	$F_5 = 3 + 2 = 5$
6	Seventh	$F_5 = 5$	$F_4 = 3$	$F_6 = 5 + 3 = 8$

Algorithm **Generating first 'n' terms of Fibonacci Sequence**

```
Step 1: Start
Step 2: Declare variables a, b, c, n, i
Step 3: Initialize variable a = 0, b = 1 and  i = 3
Step 4: Read a number n.
Step 5: Print a and b
Step 6: while i is less than or equal to n, do the following steps
          (a)  c = a + b
          (b)  print c
          (c)  a = b
          (d)  b = c
          (e)  increment i by 1 (i = i + 1)
Step 7: End
```

Let us consider n value is 7 (n=7). We need to print the Fibonacci sequence up to 7 terms as shown below.

0 1 1 2 3 5 8

We know that first number and second number in Fibonacci sequence is 0 and 1. Hence variables a and b are initialized to 0 and 1 respectively. In Step 5, We directly print a and b, So the Output till Step 5 is-

0 1

First two numbers are already printed and hence we need to print 3rd number onwards. The variable 'i' is initialized to 3 in step 3.

In Step 6, value of "i" is 3 as the first two numbers (0, 1) are already printed. Now for every iteration of "i", we add a and b in Step 6(a) , so variable "c" now becomes 1 (i.e. 0 + 1). Then the value of "c" is printed in Step 6(b). So Output now becomes -

0 1 1

In Step 6(c) and 6(d), we move 2nd last value (which is stored in variable "b") to variable "a" and last value (which is stored in variable "c") to b. So Now,

a = 1, b = 1

In Step 6(e) , Value of "i" is incremented by 1 (i.e. i = i + 1) and iteration again continues until value of "i" is less than user entered number "n".

So, For i=4, Again following steps 6(a) to 6(e), Output will be-

0 1 1 2

So, For i=5, Again following Steps 6(a) to 6(e), Output will be-

0 1 1 2 3

So, For i=6, Again following Steps 6(a) to 6(e), Output will be-

0 1 1 2 3 5

So, For i=7, Again following Steps 6(a) to 6(e), Output will be-

0 1 1 2 3 5 8

For i=8, condition i less than n becomes false as (8<=7) is false. Hence the Final Output will be -

0 1 1 2 3 5 8

2.7 Reversing the Digits of an Integer

Problem

Design an algorithm that accepts a positive integer and reverses the order of its digits.

Example: Input : 12345

Output : 54321

Reversing the digits of an integer means printing the given number back to the front. For example, the given number is 789, then the reverse of this number is 987. Let us see how we can build the logic.

Let us declare three variables number, lastDigit and reverse.

- Number is a variable that originally contains our integer,
- LastDigit is a variable that contains the last digit of number,
- Reverse is a variable representing the reverse of number.

Step 1 : Isolate the last digit in number

lastDigit = number % 10

The modulo operator (%) returns the remainder of a divison. In this case, we divide number by 10 and return the remainder. Consider the integer 1234. When we divide 1234 by 10, the remainder of the division will be 4. The remainder of the division will be 4 representing the ones column which could not be divided by 10. We now have last digit of a number We save this in the variable lastDigit.

Step 2 : Append lastDigit to reverse

reverse = (reverse * 10) + lastDigit

We can start building the reversed number now by appending lastDigit onto the end of reverse. Remember that both of these variables are integers so we can not simply concatenate. We multiply reverse by 10 so that the ones column becomes the tens column, the tens column becomes the hundreds column, and so on. This also leaves us with a new ones column where we can add our lastDigit which we determined was 4 in our example. The initial value of reverse is zero.

reverse = (0 * 10) + 4

reverse = 4

We have successfully copied the last digit of number and appended it to reverse.

Step 3 : Remove last digit from number

number = number / 10

Notice that number is still 1234, even though we have already used that 4 on the end. We have isolated the last digit and added it to reverse. Now, we need to isolate '3' from 1234. To do that, first divide the number by 10 to remove the last digit 4. To remove the last digit from number we simply divide it by 10. This works because we are performing integer division which rounds results down to the nearest integer (example: 1234/ 10 = 123, 5879/10=587).

Repeat this process until number is reduced to zero and reverse is completed.

Example **Reversal of a Number**

Let us consider number = 1234 and reverse = 0.

The statement lastDigit = number % 10 gives the remainder and number = number/10 gives the quotient.

In the first iteration, lastDigit=4 and reverse = 0*10+4 = 4 and n becomes 123

In the second iteration, lastDigit=3 and reverse = 4*10+3 = 43 and n becomes 12

In the third iteration, lastDigit=2 and reverse = 43*10+2 = 432 and n becomes 1

In the fourth iteration, lastDigit=1 and reverse = 432*10+1 = 4321 and n becomes 0

Since n becomes 0, we stop the iteration. Finally we can print the value of reverse ie. 4321. We can see that by multiplying the number by 10, we are shifting the place by one unit. The iterations performed are shown in the following table.

Step	while (num > 0)	lastDigit = number %10	number = number/10	reverse = reverse * 10+lastDigit
1	while (1234>0) = True	lastDigit =1234 %10 lastDigit = 4	number = 1234/10 number = 123	reverse = 0 * 10+4 reverse = 4
2.	while (123>0) = True	lastDigit = 123 %10 lastDigit = 3	number = 123/10 number = 12	reverse = 4 * 10+3 reverse = 40+3 = 43
3	while (12>0) = True	lastDigit = 12% 10 lastDigit = 2	number = 12/10 number = 1	reverse = 43 * 10+2 reverse = 430+2 = 432
4	while (1> 0) = True	lastDigit = 1 %10 lastDigit = 1	number = 1/10 number = 0	reverse = 432 * 10+1 reverse = 4320+1 = 4321
5	while (0 >0) = False	—	—	—

The last value of reverse is the reverse of a given number 1234.

Algorithm Reversing the Digits of an Integer

```
Step 1: Start
Step 2: Declare variables number, reverse, lastDigit
Step 3: Initialize reverse to 0. (Set reverse = 0)
Step 4: Read a number n.
Step 5: while n not equal to 0, do the following steps
        (a)  lastDigit = number % 10
        (b)  reverse = (reverse * 10) + lastDigit
        (c)  number = number / 10
Step 6: Print reverse
Step 7: End
```

2.8 Character to Number Conversion

Problem

Given the character representation of an integer convert it to its conventional decimal format.

The character representation of an integer is nothing but integers represented as strings. In most the programming languages, these strings are enclosed in double quotes (" ").

Example : "987" is character representation of an integer. But not in decimal format

987 is an integer representation in decimal format.

A common task in computer programs is to convert strings (e.g. "1234") to a numeric value 123 The string version comes from a text file, or perhaps is typed in at the keyboard. We cannot perfor arithmetic operations like addition, subtraction, multiplication, or division on numbers if it represented in characters or strings. So, we need to convert character representation of an integer numeric values in decimal format in order to perform arithmetic operations.

Most languages provide convenient library routines to perform these operations. Occasionally might need to do this in a program. Let us understand few important concepts before we solve th problem.

How Decimals are Stored in Memory?

We are familiar with the decimal number system which is used in our day-to-day work. Ten digits are used form decimal numbers. To represent these decimal digits, ten separate symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9 a used. We know that computer understands only 0's and 1's. So, each decimal digit, letters, symbols etc. writte by the programmer (an user) are converted to binary codes in the form of 0's and 1's within the computer.

Decimal number system have ten digits represented by 0,1,2,3,4,5,6,7,8 and 9. So, the base or radix of suc system is 10.

Example: Number = 65

Binary equivalent of $(65)_{10}$ is $(1000001)_2$.

Typically, integers will be stored in computer using 4 bytes (32 bit) of memory.

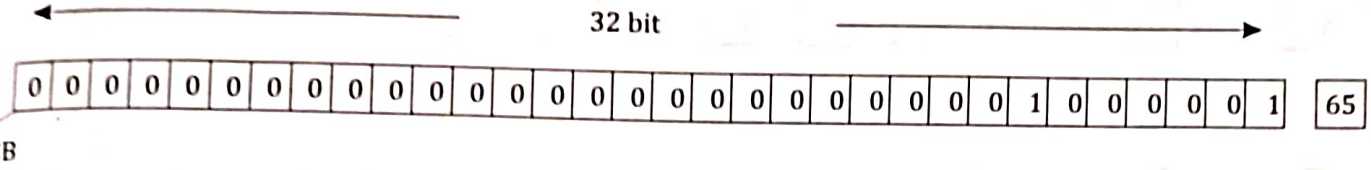
The MSB (most significant bit) bit is used to indicate whether the number is positive or negative.

For positive numbers MSB will be 0.

For negative numbers MSB will be 1.

In our case, 65 is positive so MSB will be 0.

This binary equivalent of 65 will be stored in 32-bit memory like below,



How Characters are Stored in Memory?

Characters are the things that humans recognize as having some meaning. Since a computer only deals with 0/1 (binary numbers), there is a mapping between characters and character-codes (number). The mapping commonly used in computers is ASCII.

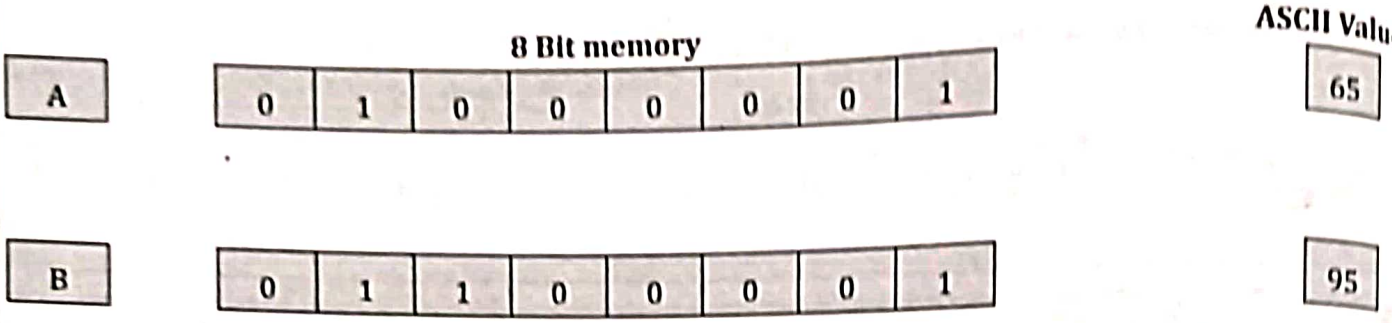
Computers operate using numbers. They therefore there need a way for a computer to convert letters (and other "characters") to and from numbers so that they can be stored inside the computer and manipulated by the computer. A set of codes, known as "ASCII" (American Standard Code for Information Interchange) are used.

Each letter is assigned a value according to its position within the ASCII table. Every letter, number, punctuation mark, etc. (known as a character) is given a unique code. Note there is a difference between the 8-bit binary representation of the number zero (00000000) and the corresponding ASCII '0' (00110000) character.

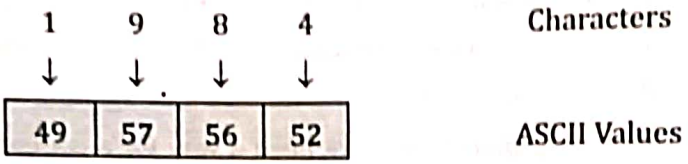
ASCII Character Codes and their Decimal Equivalents

Character	8 bit Code	Decimal Value
0	00110000	48
1	00110001	49
2	00110010	50
3	00110011	51
:	:	:
A	01000001	65
B	01000010	66
C	01000011	67
:	:	:
a	01100001	97
b	01100010	98
:	:	:

Example : If we want to store char 'A' in computer, the corresponding ASCII value will be stored in compu ASCII value for capital A is 65. To store character value, computer will allocate 1 byte (8 bit) memory. 65 w converted into binary form which is $(1000001)_2$. Because computer knows only binary number system. T binary number 1000001 will be stored in 8-bit memory in a computer. The character 'A' and 'a' is represent in memory as shown below.



Example: The character sequence of "1984" and its equivalent ASCII values are shown below.



Logic of Converting Character to Number

Now, we know that each character is associated with ASCII code. Let us consider the ASCII code of digits fro 0 to 9 as shown below.

Character	0	1	2	3	4	5	6	7	8	9
ASCII Code	48	49	50	51	52	53	54	55	56	57

We can use ASCII value of the given character for conversion. The respective decimal is calculated from th ASCII value by subtracting it from the ASCII value of 0. In other words, it converts the character to integer b finding the difference between the ASCII value of this character and the ASCII value of 0. The conversion fro character-code to digit is:

digit = ASCII value of (character) -ASCII value of ('0')

Example: Character '5' to digit 5 is shown below.

digit = ASCII Value of ('5') - ASCII value of ('0')
= 53 - 48
= 5

How to get ASCII value for any character?

In some programming languages like C, and C++ , there is no separate function to get the ASCII value of character. For example, when we use %d format specifier for any character, we can get its equivalent ASC value. You will understand more about format specifiers in subsequent chapters. In few languages like pytho and pascal, there exists a predefined function ord() to get the ASCII value of a character.

Example **Steps to Convert Character Sequence "725" to Number**

Converting string to numbers can be done one character at a time. Process the characters from left to right and build up the final value. The steps are:

Step 1: Let us consider number n ='725'

Step 2: The integer value will be stored in a variable 'value' and its initial value is 0. (value=0)

Step 3: Read first character '7'

Step 4: Find the ASCII value of '7' - which is 55

Step 5: Compute value = value * 10 + (ASCII value of ('7')- ASCII value of ('0'))

value = 0 * 10 + (55-48)

value = 0 + 7

Now, we need to repeat Step 3 to Step 5 for each character. Let us read next character '2'

Step 3: Read second character '2'

Step 4: Find the ASCII value of '2' - which is 50

Step 5: Compute value = value * 10 + (ASCII value of ('2')- ASCII value of ('0'))

value = 7 * 10 + (50-48)

value = 70 + 2

value = 72

Let us read next character '5'

Step 3: Read third character '5'

Step 4: Find the ASCII value of '5' - which is 53

Step 5: Compute value = value * 10 + (ASCII value of ('5')- ASCII value of ('0'))

value = 72 * 10 + (53-48)

value = 720 + 5

value = 725

We have iterated all the characters of the input. So, the variable value contains the final number.

Step 6: Print value - It prints 725

Algorithm **Converting Character Representation of an Integer it to its equivalent decimal format.**

```
Step 1: Start
Step 2: Declare number n, i and value
Step 3: Initialize value to 0. (value=0)
Step 4: Read character representation of an integer number n.
Step 5: Loop over the characters in n from left to right, do the following step
          for i = 1 to n
                value = value * 10 + (ASCII Value of ith Character - ASCII Value of '0')
Step 6: Print value
Step 7: End
```