**Document**



**Session 2024 - 2028**

**Submitted by:**

Rustgaar Ahmad 2024-CS-234

**Supervised by:**

Waqas Ali

**Course:**

Discrete Mathematics

Department of Computer Science

**University of Engineering and Technology Lahore Pakistan**

**Number System Conversion and Encryption Program**

This program is a C++ application that allows users to convert numbers between various number systems (**binary**, **octal**, **decimal**, and **hexadecimal**) and also provides functionality for **encrypting** and **decrypting messages**. Users can select the base for conversion or encryption, and the program handles all the necessary operations.

**Features**

**1. Number Conversion:**

* The program converts a number from one base to another:
  + **Binary**
  + **Decimal**
  + **Octal**
  + **Hexadecimal**
* The program ensures the input is valid for the specified base.

**2. Text Encryption:**

* The program encrypts plain text into numeric formats in the specified base provided by the user (**Binary**, **Octal**, **Decimal**, or **Hexadecimal**).
* For encryption, the program performs the following steps:
  + Converts each character to its **ASCII** equivalent.
  + Converts the ASCII value to the desired base.

**3. Text Decryption:**

* The program decrypts encoded text from a specified base (**Binary**, **Octal**, **Decimal**, or **Hexadecimal**) back to plain text.
* The decryption process involves:
  + Converting the numeric value from the specified base back to its ASCII equivalent.
  + Converting the ASCII value to the original characters.

**Program Flow:**

1. **Start the Program:**
   * Display a menu with the following options:
     + Number Conversion
     + Text Encryption
     + Text Decryption
     + Exit
2. **User Input:**
   * The user selects an option from the menu.
3. **Number Conversion:**
   * User enters the base (Binary, Octal, Decimal, Hexadecimal) and the number.
   * Convert the number into all other bases and display the results.
4. **Text Encryption:**
   * User enters the message and the base for encryption (Binary, Octal, Decimal, Hexadecimal).
   * Convert each character of the message to the chosen base and display the encrypted message.
5. **Text Decryption:**
   * User enters the encrypted message and its base.
   * Convert the encrypted numbers back to characters and display the original message.
6. **Repeat or Exit:**
   * User can return to the menu for another task or exit the program.

**Dry Run for Number Conversion:**

**Function:** Perform Conversion.

**Input:**

1. **Number:** 1010
2. **Base:** 2

**Expected Output:**

1. **Decimal:** 10
2. **Octal:** 12
3. **Hexadecimal:** A

**Dry Run Table:**

|  |  |  |
| --- | --- | --- |
| **Step** | **Operation/Variable** |  |
|  | Input Number | **1010** |
|  | Input Base | **2** |
|  | Validate Input | **Valid** |
| **4.** | Convert 1010 to decimal | **10** |
| **5.** | Convert 10 to octal | **12** |
| **6.** | Convert 10 to hexadecimal | **A** |
| **7.** | Output Decimal Representation | **10** |
| **8.** | Output Octal Representation | **12** |
| **9.** | Output Hexadecimal Representation | **A** |

**Dry Run for converting 1010 into its equivalent numbers in other formats**

* **Decimal to Binary:**

**Initial Case:**

Number = 10;

K = 0;

Array num[100];

**First Iteration:**

(number = 10, k = 0)

Num[k] = number % 2;

Num[0] = 10 % 2 = 0

Number = number / 2;

Number = 10 / 2 = 5

K++;

K = 0 + 1 = 1

**Second Iteration:**

(number = 5, k = 1)

Num[k] = number % 2;

Num[1] = 5 % 2 = 1

Number = number / 2;

Number = 5 / 2 = 2

K++;

K = 1 + 1 = 2

**Third Iteration:**

(number = 2, k = 2)

Num[k] = number % 2;

Num[2] = 2 % 2 = 0

Number = number / 2;

Number = 2 / 2 = 1

K++;

K = 2 + 1 = 3

**Fourth Iteration:**

(number = 1, k = 3)

Num[k] = number % 2;

Num[3] = 1 % 2 = 1

Number = number / 2;

Number = 1 / 2 = 0

K++;

K = 3 + 1 = 4

**Termination of the loop:**

(number = 0, k = 4)

As the number is 0 now so the loop will stop.

**Printing the binary result:**

The values in the array are:

Num[0] = 0

Num[1] = 1

Num[2] = 0

Num[3] = 1

So now as the loop prints the values in the reverse order

So the answer will be

**1010**

Which is the binary equivalent of denary system 10.

* **Decimal to Hexadecimal:**

**Initial Case:**

Number = 10;

K = 0;

Array num[100];

**First Iteration:**

(number = 10, k = 0)

Num[k] = number % 16;

Num[0] = 10 % 16 = 10

Since in hexa decimal system 10 = A

So num[0] = A

(In the program)

As the remainder is greater than equal to 10

So it will give a letter

Hence it will be equal to

Num[0] – 10 + ‘A’

Where **A** is the ascii value of A which is 65

So num[0] will be equal to A

Number = number / 16;

Number = 10 / 16 = 0

K++;

K = 0 + 1 = 1

**Termination of the loop:**

(number = 0, k = 1)

As the number is 0 now so the loop will stop just after 1st iteration.

**Printing the binary result:**

The values in the array are:

Num[0] = A

So now as the loop prints the values in the reverse order

But here it will not effect as there is only one value

So the answer will be

**A**

Which is the Hexa Decimal equivalent of denary system 10.

* **Decimal to Octal:**

**Initial Case:**

Number = 10;

K = 0;

Array num[100];

**First Iteration:**

(number = 10, k = 0)

Num[k] = number % 8;

Num[0] = 10 % 8 = 2

Store 2 in num[0]

Number=number/8:

10/8=1.

Number=1

K=k+1 = 1

**Second Iteration:**

(number = 1, k =1)

Num[k] = number % 8;

Num[1] = 1%8=1

Number = number/8:

1/8 = 0.

Number = 0

K = k + 1 =2

**Termination of the loop:**

(number = 0, k = 2)

As the number is 0 now so the loop will stop just after 2nd iteration.

**Printing the binary result:**

The values in the array are:

Num[0] = 2

Num[1] = 1

So, the octal representation of 10(decimal) is: 12.

**Dry run for converting other bases into base 10.**

* **Octal to Decimal**
* **Initial Case:**
* **Number = 12.**
* **K = 0.**
* **Decimal = 0.**

**First Iteration:**

Use the formula for converting from octal to decimal:

Extract rightmost digit

**Decimal = decimal+(digit \* 8^k)**

**Decimal = 0+(2\*8^0)**

**Decimal = 0+2**

**Decimal = 2**

**Number updated**

**Number = 12/10 = 1**

**K = 1**

**Second Iteration:**

**Number=1**

**K = 1**

**Extract leftmost digit**

**Formula**

**Decimal=2+(1\*8^1)**

**Decimal = 2+8**

**Decimal=10**

**Updated,**

**number = 1/10=0**

**k=2**

**Termination**:

The number is now 0, the loop stops.

### **Final Decimal Value:**

The decimal result is **10**.

* **Binary to Decimal**
* **Initial Case:**

Number = 1010

k = 0

**First Iteration:**

Use the formula for converting from binary to decimal:

Extract rightmost digit

Decimal = decimal+(digit\*2^k)

Decimal=0+(0\*2^0)

Decimal = 0

Updated,

number = 1010/10 = 101

k = 1

**Second Iteration:**

Number = 101

K = 1

Extract right most digit

Formula

Decimal=0+(1\*2^1)

Decimal=0+2

Decimal=2

Updated,

number = 101/10 = 10

k= 2

**Third Iteration:**

Number = 10

K = 2

Extract right most digit

Formula

Decimal = 2+(0\*2^2)

Decimal = 2+ 0

Decimal = 2

Updated,

Number = 10/10 = 1

K = 3

**Fourth Iteration:**

Number = 1

K = 3

Extract right most digit

Formula

Decimal = 2+(1\*2^3)

Decimal = 2+8

Decimal = 10

Updated,

Number = 1/10 = 0

K = 4

**Termination**:

The number is 0, so the loop terminates.

### **Final Decimal Value:**

The decimal result is **10**.

* **Hexadecimal to Decimal**
* **Initial Case:**

Number = A(10)

K = 0

Array num[100];

**First Iteration:**

Use the formula for converting from hexadecimal to decimal:

Extract rightmost digit

Decimal = decimal+(digit\*16^k)

Decimal = 0+(10\*16^0)

Decimal = 0+(10\*1) = 10

Updated number,

Number = number/16

Number = 10/16 = 0

K = k+1 = 1

**Termination**:

The number is 0, so the loop terminates.

**Final Output:**

The decimal value of “A” in base 16 is 10.

**Text Encryption**

**Binary conversion**

**Initial case**

Input case: Hi

**First Iteration**

Character: “H”

ASCII value of ‘H’: 72

**72 into Binary format**

72/2 = 36 remainder = 0

36/2 = 9 remainder = 0

18 ÷ 2 = 9 remainder 0

9 ÷ 2 = 4 remainder 1

4 ÷ 2 = 2 remainder 0

2 ÷ 2 = 1 remainder 0

1 ÷ 2 = 0 remainder 1

Binary of 72: 1001000

**Second Iteration**

Character: i

ASCII value of ‘i’: 105

**105 into binary format**

105 ÷ 2 = 52 remainder 1

52 ÷ 2 = 26 remainder 0

26 ÷ 2 = 13 remainder 0

13 ÷ 2 = 6 remainder 1

6 ÷ 2 = 3 remainder 0

3 ÷ 2 = 1 remainder 1

1 ÷ 2 = 0 remainder 1

Binary of 105: 1101001

**Final Output:**

Hi has ascii value: 72 105

Binary number:1001000 1101001.

**Octal conversion**

**Initial case**

Input case: Hi

**First Iteration**

Character: “H”

ASCII value of ‘H’: 72

**72 into Octal format**

72/8 = 9 remainder 0

9/8 = 1 remainder 1

1/8 = 0 remainder 1

Octal of 72 is: 110

**Second Iteration**

Character: i

ASCII value of ‘i’: 105

**105 into Octal Format**

105/8 = 13 remainder 1

13/8 = 1 remainder 5

1/8 = 0 remainder 1

Octal of 105 is: 151

**Final Output:**

Hi has ascii value: 72 105

Octal number: 110 151.

**Text Decryption**

1001000 1101001 = Hi = 72 105

### **Convert the binary** 1001000 **to decimal:**

#### 1st Binary Value: 1001000

* **Binary value:** 1001000

**1001000 = (1\*2^6) + (0\*2^5) + (0\*2^4) + (1\*2^3) + (0\*2^2) + (0\*2^1) + (0\*2^0)**

**1001000 = (1\*64) + (0\*32) +(0\*16) + (1\*8) + (0\*4) + (0\*2) + (0\*1)**

**1001000 = 64 + 8 = 72 = H**

The ASCII value **72** corresponds to the character **'H'**.

### **Convert the binary** 1101001 **to decimal:**

#### 2nd Binary Value: 1101001

* **Binary value:** 1101001

**1101001 = (1\*2^6) + (1\*2^5) + (0\*2^4) + (1\*2^3) + (0\*2^2) + (0\*2^1) + (1\*2^0)**

**1101001 = (1\*64) + (1\*32) + (0\*16) + (1\*8) +(0\*4) + (0\*2) + (1\*1)**

**1101001 = 64 + 32 + 8 + 1 = 105 = i**

The ASCII value **105** corresponds to the character **'i'**.

**Final Output**

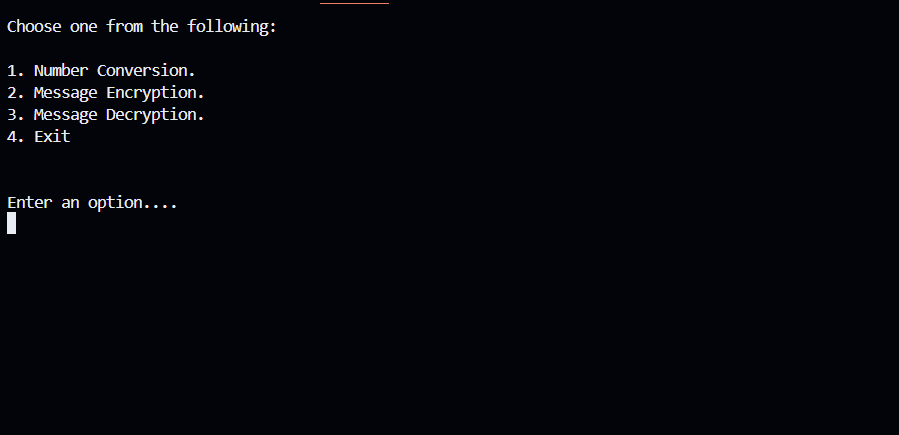
* The binary 1001000 1101001 decrypts to the ASCII values:

1001000 = 72 = Character: H

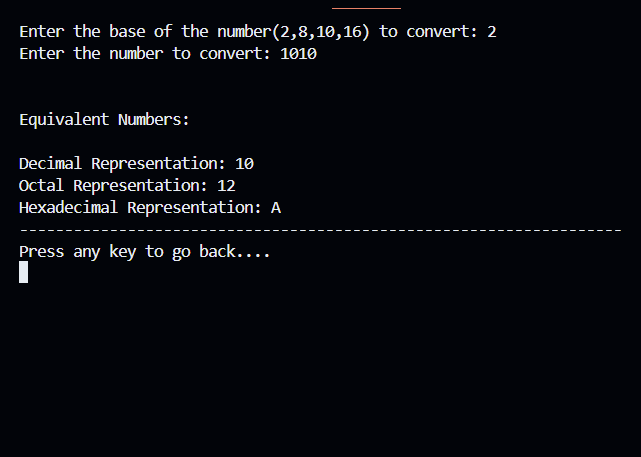
1101001 = 105 = Character: i

Thus, decrypted message is “Hi”

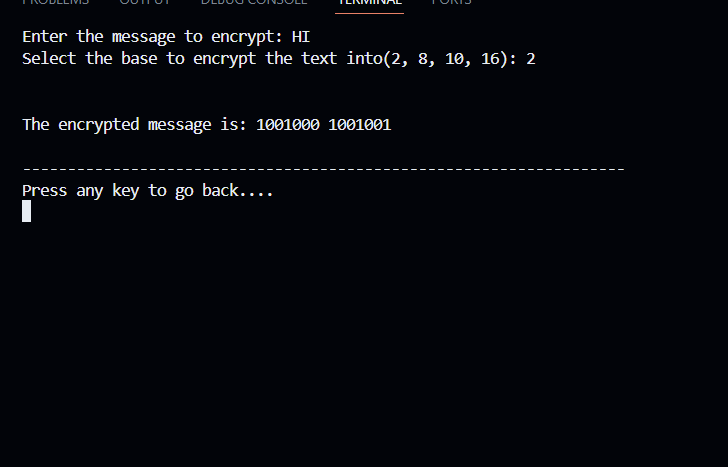
**Output snippets**

****

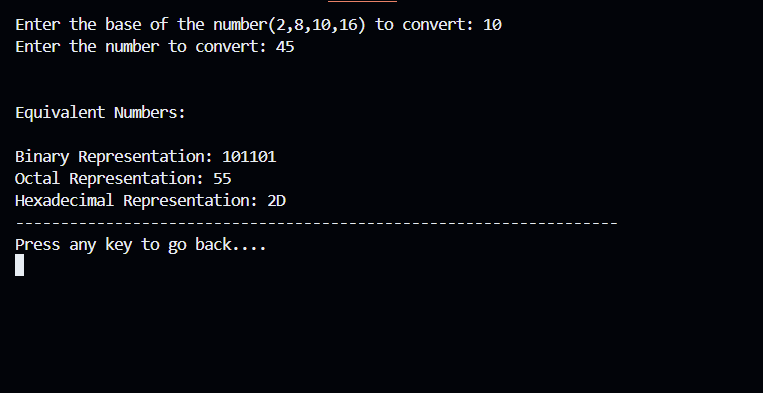
Figure



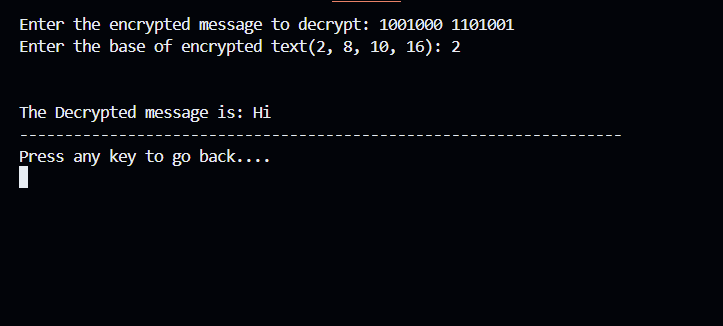
Figure



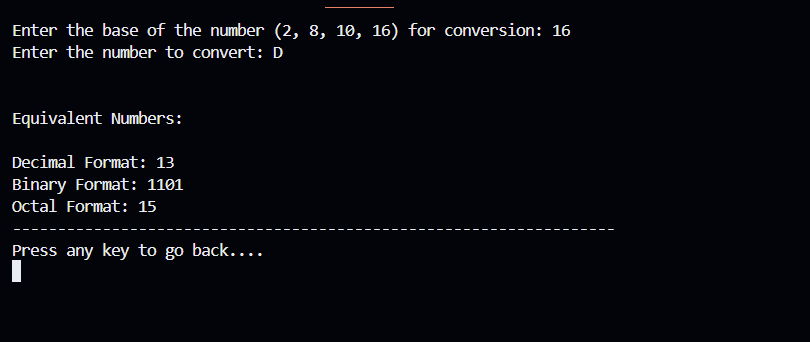
Figure



Figure



Figure



Figure