

# PROJETO INTEGRADOR – CONVERSOR DE BASES EM C

**Ana Paula Antunes Araujo**

Curso Tecnólogo em Análise e desenvolvimento de Sistema – Universidade do Vale do  
Itajaí (UNIVALI) – Campus de Itajaí  
88302-901– Itajaí – SC– Brasil

`Anapaula.antunesaraujo@outlook.com`

**Abstract.** *This meta-article presents the concepts and describes how these concepts were put into practice when solving the proposed problem. The problem situation that must be solved is to program a console application that performs the conversion between numeric bases, in C language.*

**Resumo.** *Este meta-artigo traz os conceitos e descreve como estes conceitos foram colocados em prática ao resolver o problema proposto. A situação problema que deve ser resolvida é programar uma aplicação console que realize a conversão entre bases numéricas, em linguagem C.*

## 1. Introdução

Este trabalho foi elaborado com o intuito de praticarmos o que vimos em aula.

Utilizamos alguns dos conceitos que nos foram apresentados como lógica de programação, algoritmo, sistemas de numeração, conversão entre as bases numéricas e sintaxe da linguagem C.

Este trabalho tem como intuito instigar o(a) acadêmico(a) a buscar soluções ao problema proposto, que foi: Conseguir desenvolver utilizando a linguagem C uma aplicação console intuitiva, que o usuário consiga compreender o que deve fazer e conseguir interpretar o que as ações dele podem gerar dentro da aplicação. Essa aplicação deve conseguir realizar conversões entre bases numéricas de uma forma que seja simples e fácil para o usuário.

## 2. Lógica de programação e algoritmos

A lógica de programação nada mais é do que a capacidade que todo programador precisa ter para conseguir resolver os problemas que aparecem no dia a dia. A capacidade de dividir o problema em partes menores é uma etapa essencial da lógica de programação e precisa ser levada em consideração quando nos deparamos com qualquer problema que precisa ser resolvido. É nesse ponto que entra o conceito de algoritmo, que geralmente é descrito como uma sequência lógica de ações capaz de resolver um problema (GASPAROTTO, 2017).

Ressaltando, no entanto, que o conceito de algoritmo vai muito além da programação. Qualquer resolução de problema pode ser organizada em partes como uma receita de bolo, esse é um exemplo simples de algoritmo (GASPAROTTO, 2017).

Com isso entendemos que algoritmos é muito utilizado na programação, porém pode ser aplicado em outras tarefas do dia a dia. É na verdade uma forma de pensar e organizar o que precisa ser feito.

Na programação utilizamos o pensamento lógico ao desenhar e prever cada ação e reação que a aplicação que está sendo desenvolvida precisa realizar, o que precisa ser executado primeiro, o que precisa ser executado em seguida e assim sucessivamente.

### **3. Sistemas de numeração**

Sistemas de numeração são conjuntos de símbolos e regras para representar o número de elementos de conjuntos quaisquer (GARCIA, 2011).

O número de símbolos utilizados na representação dos números é chamado de base do sistema de numeração (GARCIA, 2011).

As bases que serão vistas neste projeto são:

- Decimal
- Binária
- Octal
- Hexadecimal

#### **3.1. Sistema decimal**

Sistema posicional de base 10 em que os seus algarismos indo-arábicos (0,1,2,3,4,5,6,7,8,9) servem para contar unidades, dezenas, centenas, entre outros, da direita para a esquerda (CÓRDOVA JUNIOR, 2018).

A principal regra para escrever quantidades utilizando o sistema decimal é que todo algarismo escrito à esquerda de uma posição possui um peso 10 vezes maior do que se fosse escrito nessa posição. Por isso, o sistema numérico é conhecido como um sistema posicional (GARCIA, 2011).

#### **3.2. Sistema binário**

O sistema binário é de base 2 e utiliza os símbolos '0' e '1', sendo que cada um chamado de 'bit'. Uma sequência composta por 8 bits forma um 'byte', esse é o sistema numérico que forma a base da ciência da computação (GARCIA, 2011).

#### **3.3. Sistema octal**

O sistema octal, também chamado de sistema de base 8, possui oito algarismos (0,1,2,3,4,5,6,7), possui relação direta com o sistema binário. Foi muito utilizado na computação, na programação em linguagem de máquina, como uma alternativa mais compacta do sistema de base 2 (CÓRDOVA JUNIOR, 2018).

#### **3.4. Sistema hexadecimal**

Hexadecimal é um sistema de numeração muito utilizado na programação de microprocessadores, especialmente em equipamentos de estudos e sistemas de desenvolvimento. Utiliza os símbolos do sistema decimal mais as letras A, B, C, D, E e

F, sendo que a letra A equivale a 10, a letra B a 11, a letra C a 12, a letra D a 13, a letra E a 14 e a letra F a 15 (CÓRDOVA JUNIOR, 2018).

#### 4. Conversão de bases

A conversão de bases nada mais é do que transformar uma representação numérica de um sistema de numeração para outro. Exemplo: O número decimal 50 em binário é 00110010.

Neste capítulo vamos ver de forma breve como realizar estas conversões.

##### 4.1. Conversão de decimal para binário

A conversão de números decimais para números binários é realizada através de divisões consecutivas. Dividimos o número da base decimal por 2 até que não seja mais divisível, ao final, resultado da última divisão ajuntado dos restos das demais divisões "de baixo para cima" formam o número binário (ALVES, 2018).

Exemplo:

Figura 1. Exemplo de conversão de decimal para binário

The diagram illustrates the conversion of the decimal number 29 to its binary equivalent. It consists of a series of division steps arranged in a descending staircase pattern. Each step shows a division by 2, with the quotient and remainder. An arrow points from the final quotient to the final binary result.

29		2
1		14
		0
		7
		1
		3
		1
		1

$29_{10} = 11101_2$

Fonte: CÓRDOVA JUNIOR, 2018

#### 4.2. Conversão de decimal para octal

A conversão numérica de Decimal para Octal é muito similar a conversão de decimal para binário, a diferença é que para converter para octal é necessário dividir por 8 (ALVES, 2018).

Exemplo:

Figura 2. Exemplo de conversão de decimal para octal

$$\begin{array}{r|l} 140 & 8 \\ \hline 4 & 17 \\ & \hline & 1 \\ & \hline & 2 \end{array}$$

$$140_{10} = 214_8$$

Fonte: CORDOVA JUNIOR, 2018

#### 4.3. Conversão de decimal para hexadecimal

A conversão de números decimais para a base hexadecimal também é muito similar a conversão de decimal para binário ou octal, porém dividimos por 16. (ALVES, 2018).

Também não podemos esquecer os algarismos do sistema hexadecimal para os números 10,11,12,13,14,15, pois no lugar deles devem ser utilizadas as letras A, B, C, D, E e F (ALVES, 2018).

Exemplo:

Figura 3. Exemplo de conversão de decimal para hexadecimal

$$\begin{array}{r|l} 2834 & 16 \\ \hline 2 & 177 \\ & \hline & 11 \end{array}$$

Resultado:  $B12_{16}$

Fonte: ALVES, 2018

#### 4.4. Conversão de octal para decimal

A conversão de números da base octal para a base decimal é muito semelhante a conversão de binário para decimal, porém utilizamos 8 no lugar do número 2. (ALVES, 2018).

Exemplo:

**Figura 4. Exemplo de conversão de octal para decimal**

$$\begin{array}{cccc} 1 & 0 & 5 & 6_8 \\ (8^3 \times 1) + (8^2 \times 0) + (8^1 \times 5) + (8^0 \times 6) = 558_{10} \end{array}$$

Fonte: CÓRDOVA JUNIOR, 2018

#### 4.5. Conversão de hexadecimal para decimal

O processo para converter um número de hexadecimal para decimal é similar aos processos de conversão de binário para decimal e de octal para decimal, porém ao invés de 2 ou 8 utilizamos o 16 e é necessário substituir as letras A, B, C, D, E e F por 10, 11, 12, 13, 14 e 15 (ALVES, 2018).

Exemplo:

**Figura 5. Exemplo de conversão de hexadecimal para decimal**

$$\begin{array}{cccc} 1 & 0 & A & 6_{16} \\ (16^3 \times 1) + (16^2 \times 0) + (16^1 \times 10) + (16^0 \times 6) = 4262_{10} \end{array}$$

Fonte: CÓRDOVA JUNIOR, 2018

#### 4.6. Conversão de binário para decimal

A conversão de números binários para números decimais é realizada através de uma somatória dos algarismos binários da direita pra a esquerda onde cada termo da somatória é multiplicado por 2 elevado a um número sequencial iniciado em 0 e vai até 7 (ALVES, 2018).

Exemplo:

**Figura 6. Exemplo de conversão de binário para decimal**

$$\begin{array}{ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 1_2 \\ (2^6 \times 1) + (2^5 \times 0) + (2^4 \times 0) + (2^3 \times 1) + (2^2 \times 0) + (2^1 \times 1) + (2^0 \times 1) = 75_{10} \end{array}$$

Fonte: CÓRDOVA JUNIOR, 2018

#### 4.7. Conversão de binário para octal

A conversão de números da base binária para a base octal, é bem parecida com a conversão binário-decimal, mas antes de iniciar a conversão é preciso separar os dígitos binários de 3 em 3 da direita para a esquerda e converter separadamente cada grupo de 3 dígitos, depois concatenar os resultados (ALVES, 2018).

Exemplo:

Figura 7. Exemplo de conversão de binário para octal

$11010111_2 = 327_8$					
11	010	111	<div>Procedimento ↓</div>		
011	010	111			
421	421	421			
$0+2+1$	$0+2+0$	$4+2+1$			
3	2	7			

Fonte: CÓRDOVA JUNIOR, 2018

#### 4.8. Conversão de binário para hexadecimal

A conversão de números da base binária para a base hexadecimal é quase idêntica à conversão de binário para octal, só que agora separamos os dígitos binários de 4 em 4 da direita para a esquerda e antes de unir os dígitos ao final, trocamos os números 10, 11, 12, 13, 14 e 15 por A, B, C, D, E e F (ALVES, 2018).

Exemplo

Figura 8. Exemplo de conversão de binário para hexadecimal

$1011010111_2 = 2D7_{16}$					
10	1101	0111	<div>Procedimento ↓</div>		
0010	1101	0111			
8421	8421	8421			
$0+0+2+0$	$8+4+0+1$	$0+4+2+1$			
2	D	7			

Fonte: CÓRDOVA JUNIOR, 2018

## 5. Linguagem C

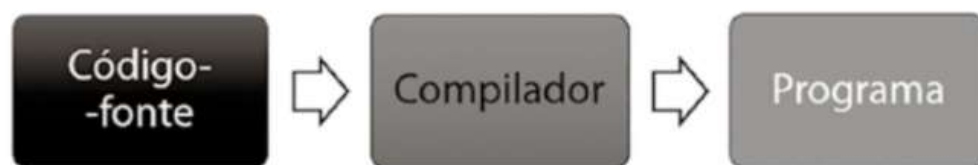
A linguagem C é uma LP bastante popular, criada por volta de 1970, por Dennis Ritchie (SANTOS, 2018).

Algumas das características da linguagem C é a portabilidade, poder e variedade dos operadores, sintaxe elegante, estruturada e flexível, acesso facilitado à memória e a todo o hardware, uso de procedimentos e funções para desenvolver sistemas desacoplados (SANTOS, 2018).

Usar a linguagem C é muito simples, é possível escrever um código fonte em C até mesmo em um bloco de notas. Para isso podemos escrever um algoritmo, depois traduzimos o mesmo para a sintaxe em C e salvamos o arquivo com um nome + .c, exemplo programa.c (SANTOS, 2018).

Com o código escrito precisamos de um compilador para gerar o programa e por fim precisamos apenas executar o sistema. (SANTOS, 2018).

**Figura 9. Como um programa compilado é criado?**



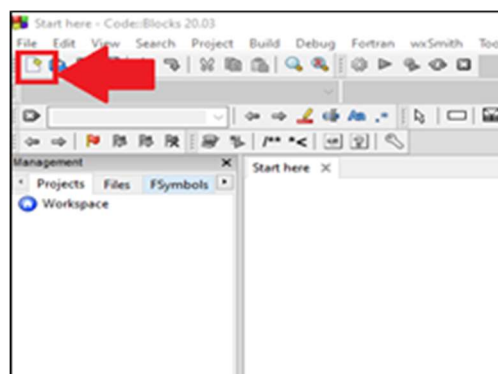
Fonte: SANTOS, 2018

## 6. Projeto prático

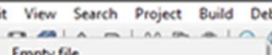
Para a aplicação dos conteúdos vistos em sala foi utilizado o sistema Code::Blocks e o compilador no Code::Blocks, o GNU GCC Compiler.

Foi criado um projeto de nome “handsonworkiii” como mostra as imagens a seguir:

**Figura 10. Criar novo projeto**

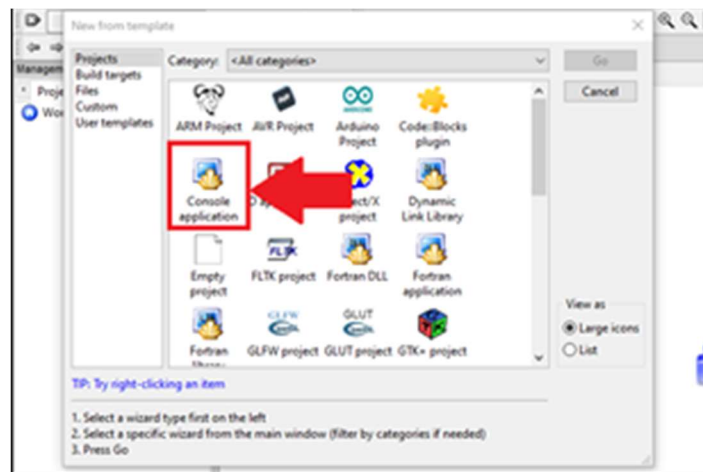


Fonte: Print feito durante o projeto

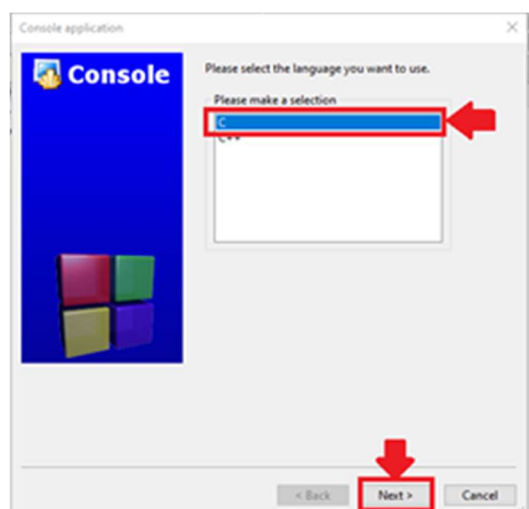


A screenshot of the Code::Blocks 20.03 application window. The 'File' menu is open, showing options: 'Empty file', 'Class...', 'Project...' (highlighted with a red rectangle), 'Build target...', 'File...', 'Custom...', 'From template...', and 'Nassi Shneiderman diagram'. A red arrow points from the right towards the 'Project...' option.

**Figura 12. Seleccionando projeto tipo console application**



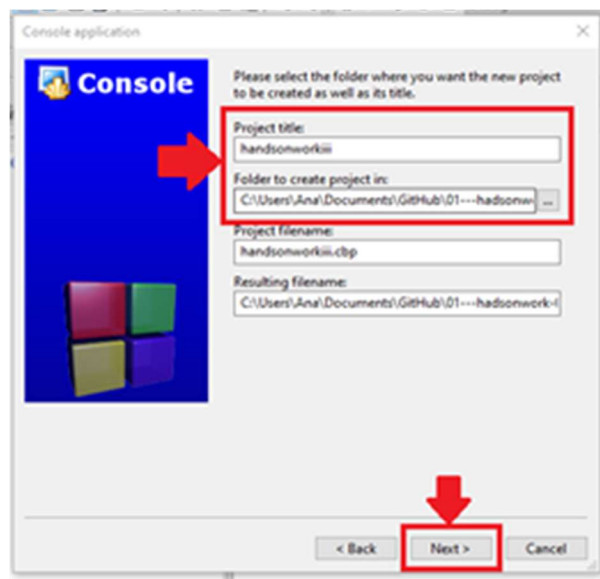
**Figura 13. Selecionando linguagem C**



**Fonte: Print feito durante o projeto**

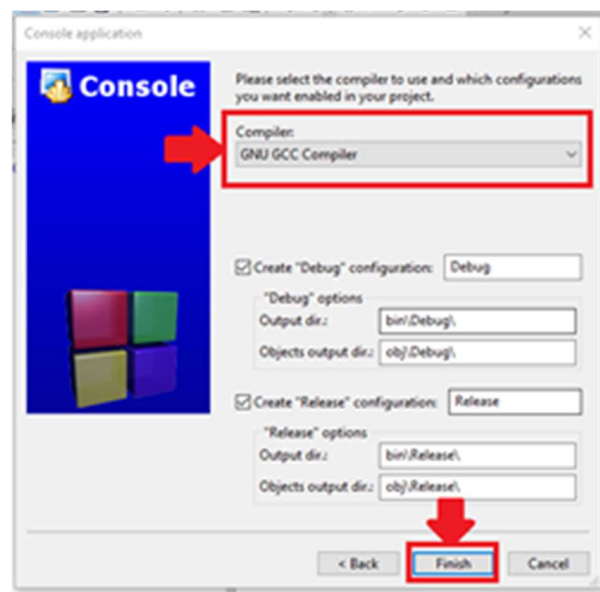


**Figura 14. Definindo o nome do projeto**



Fonte: Print feito durante o projeto

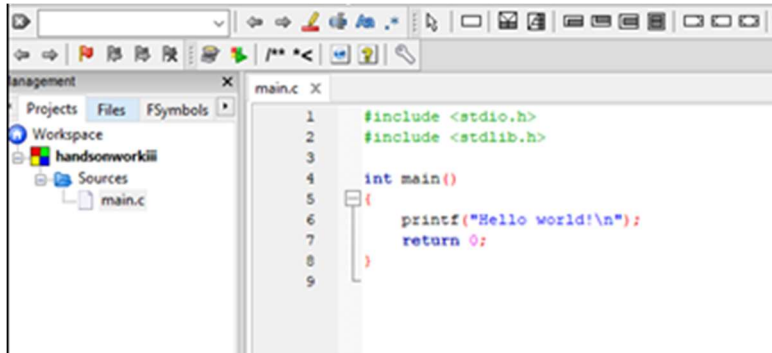
**Figura 15. Selecionando o compilador**



Fonte: Print feito durante o projeto

Por padrão o Code::Blocks cria um projeto que imprime na tela a frase “Hello world!”.

**Figura 16. Arquivo inicial do projeto**

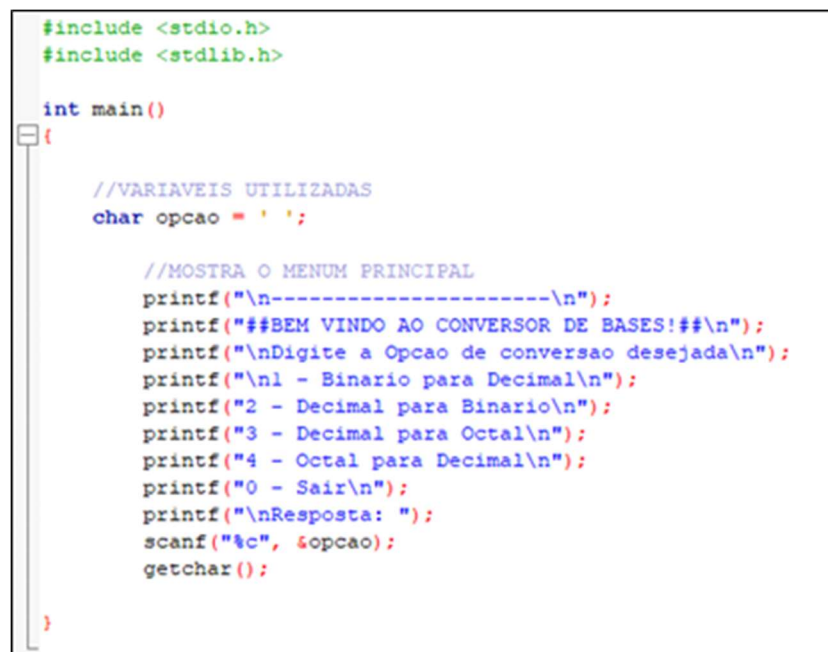


**Fonte: Print feito durante o projeto**

Apaguei este código e iniciei a digitação do código para o conversor de bases.

A princípio foi declarada a variável opcao que irá capturar a opção de conversão que o usuário selecionará. Também foi montado o menu da aplicação utilizando o comando ‘printf’ e foi configurado para pegar o que for digitado e salvar na variável opcao utilizando o comando ‘scanf’.

**Figura 17. Variáveis e menu**



**Fonte: Print feito durante o projeto**

Feito isso foi defini que utilizaria funções e que cada função realizaria um calculo de conversão.

Para criar essas funções foi feita a chamada das mesmas no início do código, fora da função main(). Nesse momento também defini mais algumas variáveis que foram inclusas juntamente com a variável opcao.

**Figura 18. Chamada de funções e variáveis**

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <string.h>
4
5  //DECLARAÇÃO OU PROTÓTIPO DAS FUNÇÕES UTILIZADAS
6  //A DECLARACAO É FEITA DA SEGUINTE FORMA:
7  //TIPO + NOME + (LISTA DE PARÂMETROS QUEA FUNÇÃO PRECISA)
8  int converterBinDec(long long valor_binario);
9  long long converterDecBin(int valor_decimal);
10 int converterDecOct(int valor_decimal);
11 long long converterOctDec(int valor_octal);
12
13 int main()
14 {
15
16     //VARIÁVEIS UTILIZADAS
17     char opcao = ' ';
18     int valor_octal;
19     long long valor_binario;
20     int valor_decimal;
21     int sequencial;
```

Fonte: Print feito durante o projeto

Depois de fazer a chamada das funções elas foram criadas dentro da função main() como mostra as imagens a seguir:

**Figura 19. Função de conversão de binário para decimal**

```
///FUNCOES PARA AS CONVERSÕES
//CONVERSÃO DE BINARIO X DECIMAL
int converterBinDec(long long valor_binario)
{
    //VARIÁVEIS UTILIZADAS NA FUNCAO
    int valor_decimal = 0, sequencial = 0, resto;

    //LAÇO DE REPETIÇÃO
    //ENQUANTO EXISTIR VALOR NO BINÁRIO ZERO
    while (valor_binario != 0)
    {
        //PEGA O RESTO DA DIVISÃO DO VALOR POR 10
        resto = valor_binario % 10;

        //DIVIDE O VALOR BINARIO POR 10
        valor_binario /= 10;

        //INCREMENTA O VALOR DECIMAL COM O RESTO DA DIVISÃO MULTIPLICANDO POR 2 ELEVADO AO SEQUENCIAL
        //VALOR DECIMAL É IGUAL A SOMA DE (OS RESTOS MULTIPLICADOS PELO NUMERO SEQUENCIAL ELEVADO A 2)
        valor_decimal += resto * pow(2, sequencial);

        //INCREMENTA A SEQUENCIAL
        ++sequencial;
    }

    return valor_decimal;
}
```

**Fonte: Print feito durante o projeto**

A conversão de binário para decimal será feita pela aplicação utilizando a seguinte lógica:

Vamos converter o número binário 0010100 que em decimal é 20.

O primeiro cálculo feito é  $10100/10$  e o resultado deste cálculo é 1010 e dessa divisão não temos resto.

Com esses números calculamos o valor decimal atual multiplicando o resto por 2 elevado ao sequencial atual e somamos o resultado com o valor decimal anterior.

Nesse caso finalizamos o primeiro ciclo do laço de repetição com as seguintes informações:

Valor\_binario = 1010

Sequencial = 0

Resto = 0

Valor\_decimal =  $\rightarrow (0 * (2)^0) + 0 = 0$

Com estes valores, dividimos o atual valor binário por 10 novamente, assim ficamos com o valor binário 101.

A variável sequencial é incrementada e recebe +1.

O valor decimal passa a ser o resultado do cálculo, o resto vezes 2 elevado ao sequencial + o valor decimal anterior. E ao fim desse ciclo ficamos com os seguintes valores em cada variável.

Valor\_binario ->  $1010/10 = 101$

Sequencial ->  $0+1 = 1$

Resto = 0

Valor\_decimal ->  $(0 * (2)^1) + 0 = 0$

Agora repetimos os mesmos cálculos até que o valor binário seja igual a 0. Nesse caso  $101/10$  nos dá como número binário o valor 10, o sequencial é incrementado mais uma vez e passa a ser 2, o resto da divisão é 1 e com isso podemos calcular o valor decimal atual. Sendo 2 elevado a 2 igual a 4, multiplicamos 4 pelo número resto que resulta ainda em 4 e depois somamos com o valor decimal anterior que era 0, então nosso número decimal passa a ser 4.

Valor\_binario ->  $101/10 = 10$

Sequencial ->  $1+1 = 2$

Resto = 1

Valor\_decimal =  $(1 * (2)^2) + 0 = 4$

Como o valor binário ainda é maior que 0 então é preciso realizar todos os cálculos mais uma vez.

Dividimos 10 por 10, o resultado é 1 e o resto é 0. O sequencial é incrementado e passa a ser 3, e o valor decimal continua sendo 4.

Valor\_binario = 1

Sequencial = 3

Resto = 0

Valor\_decimal =  $(0 * (2)^3) + 4 = 4$

Mais uma vez dividimos o valor binário, porém dessa vez o resultado retorna 0, sendo assim esse será o último ciclo do laço de repetição. O resto da divisão é 0, o sequencial ao incrementar mais 1 passa a ser 4.

O valor decimal receberá o resultado da multiplicação do resto por 2 elevado ao sequencial e ao fim a soma desse resultado com o último valor decimal.

Valor\_binario ->  $1/2 = 0$

Sequencial ->  $3+1 = 4$

Resto = 1

Valor\_decimal =  $(1 * (2)^4) + 4 = 20$

Com esse cálculo chegamos ao valor decimal 20.

**Figura 20. Função de conversão de decimal para binário**

```
//CONVERSAO DE DECIMAL X BINARIO
long long converterDecBin(int valor_decimal)
{
    //VARIÁVEIS UTILIZADAS NA FUNCAO
    long long valor_binario = 0;
    int resto, sequencial = 1;

    //LAÇO DE REPETIÇÃO
    //ENQUANTO O VALOR DECIMAL FOR DIFERENTE DE ZERO
    while (valor_decimal != 0)
    {
        //PEGA O RESTO DA DIVISAO
        resto = valor_decimal % 2;

        //DIVIDE O VALOR DECIMAL POR 2
        valor_decimal /= 2;

        //INCREMENTA O VALOR BINÁRIO, MULTIPLICANDO O RESTO DA DIVISAO PELO SEQUENCIAL
        valor_binario += resto * sequencial;

        //MULTIPLICANDO O SEQUENCIAL POR 10
        sequencial *= 10;
    }

    return valor_binario;
}
```

**Fonte: Print feito durante o projeto**

A conversão de decimal para binário será realizada da seguinte forma:

Como no exemplo anterior vamos utilizar o número decimal 20 como exemplo.

Primeiro precisamos dividir o decimal por 2, sendo que,  $20/2$  resulta em 10 e o resto dessa divisão é 0.

Olhando para as variáveis identificamos que as variáveis resto e sequencial iniciam com o valor 1 e a variável de valor binário inicia em 0.

A seguir um resumo dos valores que temos até o momento em cada variável.

Valor\_binario = 0

Sequencial = 1

Resto = 0

Valor\_decimal ->  $20/2 = 10$

Agora precisamos realizar os seguintes cálculos.

O sequencial será multiplicado por 10, sendo assim passará a ser 10, o valor decimal será dividido novamente por 2 e passará a ser 5, o resto permanece como 0 e o valor binário será calculado multiplicando o resto pelo sequencial e somando o resultado dessa multiplicação com o valor binário atual.

$$\text{Valor\_binario} \rightarrow (0 \cdot 10) + 0 = 0 = 0$$

$$\text{Sequencial} = 10$$

$$\text{Resto} = 0$$

$$\text{Valor\_decimal} \rightarrow 10/2 = 5$$

Multiplicamos novamente o sequencial por 10, resultando em 100, dividimos o valor decimal atual por 2, o resultado desta divisão é 2 e o resto da mesma é 1. Novamente calculamos o valor binário multiplicando o resto pelo sequencial e por fim somamos o resultado da multiplicação com o último valor binário.

$$\text{Valor\_binario} \rightarrow (1 \cdot 100) + 0 = 100$$

$$\text{Sequencial} \rightarrow 10 \cdot 10 = 100$$

$$\text{Resto} = 1$$

$$\text{Valor\_decimal} \rightarrow 5/2 = 2$$

Como o valor decimal ainda é maior que 0 então o laço de repetição ainda não é encerrado e a aplicação realizaria os mesmos cálculos mais uma vez, esses cálculos resultam nos seguintes valores

$$\text{Valor\_binario} \rightarrow (100 \cdot 0) + 100 = 100$$

$$\text{Sequencial} \rightarrow 100 \cdot 10 = 1000$$

$$\text{Resto} = 0$$

$$\text{Valor\_decimal} \rightarrow 2/2 = 1$$

Agora o valor decimal é 1, sendo assim o resultado de sua divisão por 2 será igual a 0, então o ciclo será repetido apenas mais esta vez e então teremos o resultado.

Realizando todos os cálculos novamente conseguimos os seguintes valores.

$$\text{Valor\_binario} \rightarrow (10000 \cdot 1) + 100 = 10100$$

$$\text{Sequencial} \rightarrow 1000 \cdot 10 = 10000$$

Resto = 1

Valor\_decimal ->  $1/2 = 0$

Sendo assim 20 em binário é 10100

**Figura 21. Função de conversão de decimal para octal**

```
//CONVERSAO DE DECIMAL X OCTAL
int converterDecOct(int valor_decimal)
{
    //VARIÁVEIS UTILIZADAS NA FUNCAO
    int valor_octal = 0, sequencia = 1;

    //LAÇO DE REPETIÇÃO
    //ENQUANTO O VALOR DECIMAL FOR DIFERENTE DE ZERO
    while (valor_decimal != 0)
    {
        //INCREMENTA O VALOR OCTAL COM O RESTO DA DIVISAO DO DECIMAL POR 8 MULTIPLICANDO PELO SEQUENCIAL
        valor_octal += (valor_decimal % 8) * sequencia;

        //O VALOR DECIMAL SERÁ DIVIDIDO POR 8
        valor_decimal /= 8;

        //O SEQUENCIAL SERÁ MULTIPLICADO POR 10
        sequencia *= 10;
    }

    return valor_octal;
}
```

**Fonte: Print feito durante o projeto**

A conversão de decimal para octal é muito similar ao cálculo de conversão de decimal para binário, porém dividimos os valores por 8 ao invés de dividir por 2.

Vamos ver como ficaria o cálculo em um teste de mesa convertendo assim como nos exemplos anteriores o valor decimal 20.

Iniciamos o cálculo com os seguintes valores:

Valor\_octal = 0

Sequencial = 1

Valor\_decimal = 20

Com estes valores realizamos o seguinte cálculo.

Pegamos o resto da divisão de 20 por 8, que é 4 e multiplicamos pelo sequência atual que é 1, o resultado é 4, depois somamos este valor com o valor octal atual que é 0, sendo assim o novo valor octal passa a ser 4.

O sequencial depois desse cálculo é multiplicado por 10 e passa a valer 10 e o valor decimal é dividido por 8 e passa a ser 2.



Valor\_octal = 4

Sequencial = 10

Valor\_decimal = 2

Realizamos o mesmo cálculo mais uma vez. Conseguimos o resto da divisão de 2 por 8, que é 2, multiplicamos esse valor pelo sequencial atual que é 10, o que resulta em 20, e esse valor somamos ao valor octal atual que é 4, nosso resultado é 24.

Valor\_octal = 24

Sequencial = 100

Valor\_decimal = 0

Como o valor decimal chegou a 0 o laço de repetição se encerra e temos como resultado da conversão do valor decimal 20 para octal o valor 24.

**Figura 22. Função de conversão de octal para decimal**

```
//CONVERSAO DE OCTAL X DECIMAL
long long converterOctDec(int valor_octal)
{
    //VARIÁVEIS UTILIZADAS NA FUNCAO
    int valor_decimal = 0, sequencia = 0;

    //LAÇO DE REPETIÇÃO
    //ENQUANTO O VALOR OCTAL FOR DIFERENTE DE ZERO
    while(valor_octal != 0)
    {
        //INCREMENTA O VALOR DECIMAL COM O RESTO DA DIVISÃO DO VALOR OCTAL POR 10 MULTIPLICADO POR 8 ELEVADO PELO SEQUENCIAL
        valor_decimal += (valor_octal % 10) * pow(8, sequencia);

        //INCREMENTA O SEQUENCIAL
        ++sequencia;

        //DIVIDE O VALOR OCTAL POR 10
        valor_octal /= 10;
    }

    return valor_decimal;
}
```

**Fonte: Print feito durante o projeto**

Para converter um valor de octal para decimal, o processo é bem similar a conversão de binário para decimal, porém o número 8 será elevado ao valor de sequencial.

Vamos converter como exemplo o valor octal 24, que em decimal é 20.

Começamos com o valor decimal e sequencial igual a 0 e o primeiro cálculo realizado é multiplicar o resto da divisão 24/10 por 8 elevado pelo sequencial.

Ficamos com os seguintes valores

Valor\_decimal ->  $0 + (24 \% 10) * (8)^0 \rightarrow 0 + 4 * 1 = 4$

Valor\_octal = 24

Sequencial = 0

Depois destes cálculos o valor sequencial é incrementado e passa a ser 1, o valor octal é dividido por 10 resultando no valor 2,4. O resto de 2,4 por 10 é 2, o valor 8 elevado ao sequencial 1 é igual a 8, 2 vezes 8 é igual a 16, somamos este valor com o valor decimal anterior e chegamos ao resultado 20.

Valor\_decimal ->  $4 + (2,4 \% 10) * (8)^1 \rightarrow 4 + 2 * 8 = 20$

Valor\_octal = 2,4

Sequencial = 1

Após este cálculo o sequencial é incrementado e passa a ser 2 e o valor octal é dividido por 10, nessa divisão o valor octal passa a ser igual a 0, com isso o laço de repetição é encerrado e nosso resultado é o valor decimal 20.

Com as funções prontas comecei a montar as condicionais para cada opção que o usuário pode selecionar.

**Figura 23. Comando IF para as condicionais**

```
//OPÇÃO 1 - BINÁRIO -> DECIMAL
else if (opcao == '1') {
    printf("Digite número binário: ");
    scanf("%lld", &valor_binario);
    printf("%lld em binário = %d em decimal\n", valor_binario, converterBinDec(valor_binario));
}

//OPÇÃO 2 - DECIMAL -> BINÁRIO
else if (opcao == '2') {
    printf("Digite o número decimal: ");
    scanf("%d", &valor_decimal);
    printf("%d em decimal = %lld em binário\n", valor_decimal, converterDecBin(valor_decimal));
}

//OPÇÃO 3 - DECIMAL -> OCTAL
else if (opcao == '3') {
    printf("Digite o número decimal: ");
    scanf("%d", &valor_decimal);
    printf("%d em decimal = %d em octal\n", valor_decimal, converterDecOct(valor_decimal));
}

//OPÇÃO 4 - OCTAL -> DECIMAL
else if (opcao == '4') {
    printf("Digite o número octal: ");
    scanf("%d", &valor_octal);
    printf("%d em octal = %lld em decimal\n", valor_octal, converterOctDec(valor_octal));
}

//OPÇÃO DESCONHECIDA
else {
    printf("Opcao desconhecida(%c)\n", opcao);
}
```

**Fonte: Print feito durante o projeto**

Com as condicionais criadas a aplicação já começa a funcionar e converter as bases, porém preferi fazer um laço de repetição para poder voltar ao menu quando uma conversão for executada e para fechar quando clicar em “0”.

Para isso adicionei mais uma variável:

Figura 24. Variável para laço de repetição

```
int main()
{

    //VARIÁVEIS UTILIZADAS
    char opcao = ' ';
    int valor_octal;
    int opcao2 = 1;
    long long valor_binario;
    int valor_decimal;
    int sequencial;
```

Fonte: Print feito durante o projeto

Também adicionei uma função While antes do menu do Sistema e nas condicionais adicionei a opção de digitar 0 que preencher a variável opcao2 com o valor 2 e fecha a aplicação:

Figura 25. Variável para laço de repetição

```
//ENQUANTO FOR A OPCAO 1 EXECUTA NOVAMENTE, SE FOR DIFERENTE DE 1 SISTEMA É FINALIZADO
while (opcao2 == 1) {

    printf("\n-----\n");
    printf("##BEM VINDO AO CONVERSOR DE BASES:##\n");
    printf("\nDigite a Opcao de conversao desejada\n");
    printf("\n1 - Binario para Decimal\n");
    printf("\n2 - Decimal para Binario\n");
    printf("\n3 - Decimal para Octal\n");
    printf("\n4 - Octal para Decimal\n");
    printf("\n0 - Sair\n");
    printf("\nResposta: ");
    scanf("%c", &opcao);
    getchar();

    //OPÇÃO 0 - FECHA O SISTEMA
    if (opcao == '0') {
        printf("Clique em ENTER e o sistema sera finalizado\n");
        opcao2 = 2;
        exit(0);
    }

    //OPÇÃO 1 - BINARIO -> DECIMAL
    else if (opcao == '1') {
        printf("Digite numero binario: ");
        scanf("%lld", &valor_binario);
        printf("[%lld] em binario = [%d] em decimal\n", valor_binario, converterBinDec(valor_binario));
    }
}
```

Fonte: Print feito durante o projeto

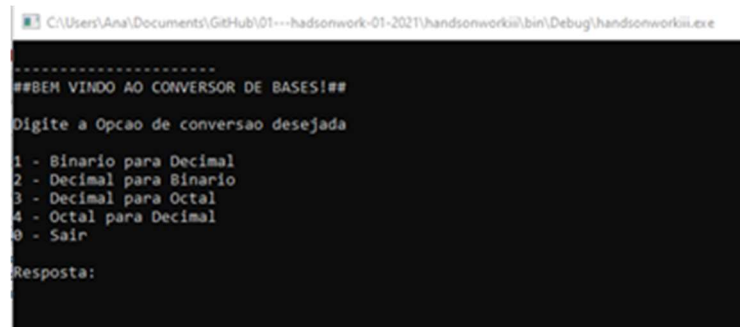
Link para Código: <https://github.com/anapaulaantunesaraujo/ConversorDeBases>

Link para video de apresentação: [https://youtu.be/n\\_7E\\_qwXByw](https://youtu.be/n_7E_qwXByw)

## 6.1. Como ficou?

Ao final a aplicação ficou com a aparência conforme imagem a seguir:

**Figura 26. Imagem de testes feitos com a aplicação**



```
C:\Users\Ana\Documents\GitHub\01---hadsonwork-01-2021\handsonworkiii\bin\Debug\handsonworkiii.exe

-----
##BEM VINDO AO CONVERSOR DE BASES!##
Digite a Opcao de conversao desejada

1 - Binario para Decimal
2 - Decimal para Binario
3 - Decimal para Octal
4 - Octal para Decimal
0 - Sair

Resposta:
```

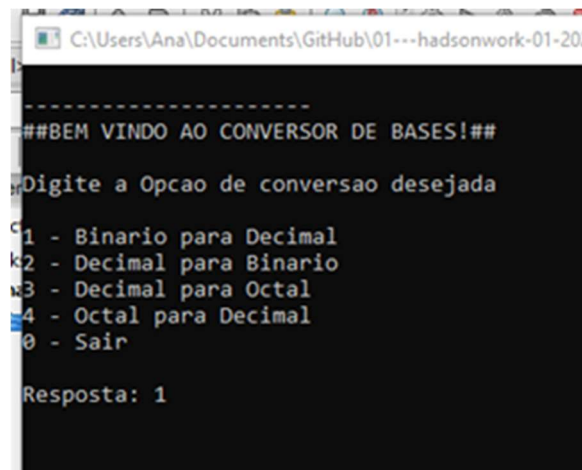
**Fonte: Print feito durante o projeto**

Vejamos um exemplo, testamos a conversão do valor decimal 80 entre as bases disponíveis na aplicação.

Primeiro de binário para decimal, no caso 80 em binário é 01010000.

Selecionar nesse caso a opção 1.

**Figura 27. Imagem do teste de conversão de binário para decimal**



```
C:\Users\Ana\Documents\GitHub\01---hadsonwork-01-202

-----
##BEM VINDO AO CONVERSOR DE BASES!##
Digite a Opcao de conversao desejada

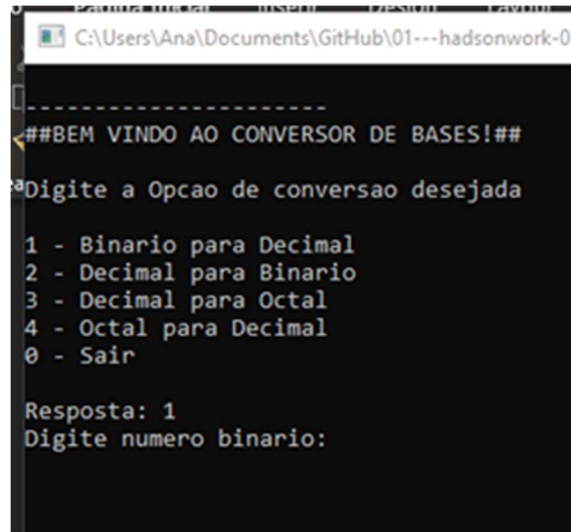
1 - Binario para Decimal
2 - Decimal para Binario
3 - Decimal para Octal
4 - Octal para Decimal
0 - Sair

Resposta: 1
```

**Fonte: Print feito durante o projeto**

Após digitar 1, clicar em enter e a aplicação irá perguntar qual número binário deseja converter.

Figura 28. Imagem do teste de conversão de binário para decimal

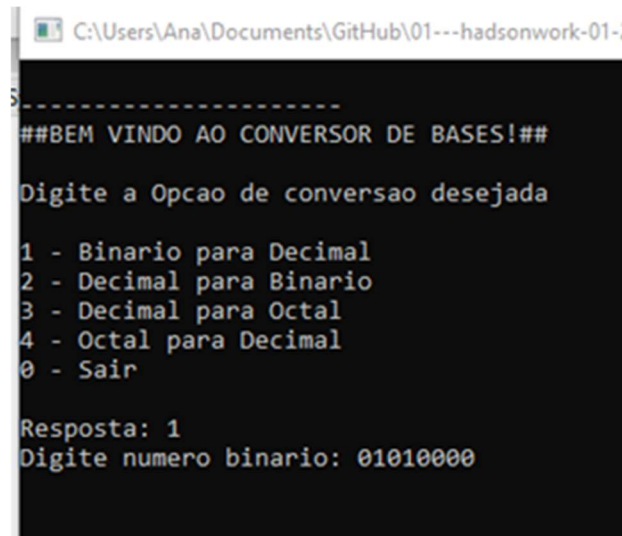


```
C:\Users\Ana\Documents\GitHub\01---hadsonwork-01-  
-----  
##BEM VINDO AO CONVERSOR DE BASES!##  
Digite a Opcao de conversao desejada  
1 - Binario para Decimal  
2 - Decimal para Binario  
3 - Decimal para Octal  
4 - Octal para Decimal  
0 - Sair  
  
Resposta: 1  
Digite numero binario:
```

Fonte: Print feito durante o projeto

Digitar o valor binário 01010000, depois de digitar o número binário clicar em enter.

Figura 29. Imagem do teste de conversão de binário para decimal



```
C:\Users\Ana\Documents\GitHub\01---hadsonwork-01-2  
-----  
##BEM VINDO AO CONVERSOR DE BASES!##  
Digite a Opcao de conversao desejada  
1 - Binario para Decimal  
2 - Decimal para Binario  
3 - Decimal para Octal  
4 - Octal para Decimal  
0 - Sair  
  
Resposta: 1  
Digite numero binario: 01010000
```

Fonte: Print feito durante o projeto

A aplicação irá retornar com o resultado da conversão.

Figura 30. Imagem do teste de conversão de binário para decimal

```
C:\Users\Ana\Documents\GitHub\01---hadsonwork-01-21-2021

#####
##BEM VINDO AO CONVERSOR DE BASES!##
#####
Digite a Opcao de conversao desejada

1 - Binario para Decimal
2 - Decimal para Binario
3 - Decimal para Octal
4 - Octal para Decimal
0 - Sair

Resposta: 1
Digite numero binario: 01010000
[1010000] em binario = [80] em decimal
#####
##BEM VINDO AO CONVERSOR DE BASES!##
#####
Digite a Opcao de conversao desejada

1 - Binario para Decimal
2 - Decimal para Binario
3 - Decimal para Octal
4 - Octal para Decimal
0 - Sair

Resposta:
```

Fonte: Print feito durante o projeto

Agora de decimal para binário.

Selecionar opção 2 e clicar em enter.

Figura 31. Imagem do teste de conversão de decimal para binário

```
C:\Users\Ana\Documents\GitHub\01---hadsonwork-01-2021

#####
##BEM VINDO AO CONVERSOR DE BASES!##
#####
Digite a Opcao de conversao desejada

1 - Binario para Decimal
2 - Decimal para Binario
3 - Decimal para Octal
4 - Octal para Decimal
0 - Sair

Resposta: 1
Digite numero binario: 01010000
[1010000] em binario = [80] em decimal
#####
##BEM VINDO AO CONVERSOR DE BASES!##
#####
Digite a Opcao de conversao desejada

1 - Binario para Decimal
2 - Decimal para Binario
3 - Decimal para Octal
4 - Octal para Decimal
0 - Sair

Resposta: 2
```

Fonte: Print feito durante o projeto

Digitar 80 e clicar em enter.

Figura 32. Imagem do teste de conversão de decimal para binário

```
-----  
##BEM VINDO AO CONVERSOR DE BASES!##  
  
Digite a Opcao de conversao desejada  
  
1 - Binario para Decimal  
2 - Decimal para Binario  
3 - Decimal para Octal  
4 - Octal para Decimal  
0 - Sair  
  
Resposta: Digite o numero decimal: 80
```

Fonte: Print feito durante o projeto

A aplicação retornará com o resultado da conversão.

Figura 33. Imagem do teste de conversão de decimal para binário

```
-----  
##BEM VINDO AO CONVERSOR DE BASES!##  
  
Digite a Opcao de conversao desejada  
  
1 - Binario para Decimal  
2 - Decimal para Binario  
3 - Decimal para Octal  
4 - Octal para Decimal  
0 - Sair  
  
Resposta: Digite o numero decimal: 80  
[80] em decimal = [1010000] em binario  
  
-----  
##BEM VINDO AO CONVERSOR DE BASES!##  
  
Digite a Opcao de conversao desejada  
  
1 - Binario para Decimal  
2 - Decimal para Binario  
3 - Decimal para Octal  
4 - Octal para Decimal  
0 - Sair  
  
Resposta:
```

Fonte: Print feito durante o projeto

## **Referências**

ALVES, Gustavo Furtado de Oliveira. As 10 conversões numéricas mais utilizadas na computação. 2018. Disponível em: <<https://dicasdeprogramacao.com.br/as-10-conversoes-numericas-mais-utilizadas-na-computacao/>>. Acesso em: 13 mar. 2021.

CÓRDOVA JUNIOR, Ramiro Sebastião. FUNDAMENTOS COMPUTACIONAIS. Porto Alegre: SAGAH, 2018. 194 p.

GARCIA, Ney. Lógica De Programação De Computadores: Introdução à Construção de Algoritmos. Joinville: Clube de Autores (managed), 2011. 163 p.

GASPAROTTO, Henrique Machado. Lógica de programação: introdução a algoritmos e pseudocódigo. 2017. Disponível em: <<https://www.devmedia.com.br/logica-de-programacao-introducao-a-algoritmos-e-pseudocodigo/37918>>. Acesso em: 13 mar. 2021.