Programando com recursão
Professor Wladimir A. Tavares

Recursão com números

Um número N pode ser representado utilizando dígitos em uma base b, onde os dígitos variam de 0 até $b-1$. Por exemplo, na base decimal (base 10), os dígitos disponíveis são 0 a 9. Para determinar os dígitos de um número N na base 10, podemos aplicar as seguintes operações matemáticas:

```
digito_menos_significativo = N % 10
parte_superior = N / 10
```

- A operação **resto da divisão por 10 (% 10)** retorna o dígito menos significativo do número.
- A operação **divisão inteira por 10 (/ 10)** remove o dígito menos significativo, ”encurtando”o número.

O seguinte programa em C++ demonstra como extrair e exibir os dígitos de um número N na base 10:

```
int N;
scanf("%d", &N);
while (N > 0) {
    printf("%d\n", N % 10);
    N = N / 10;
}
```

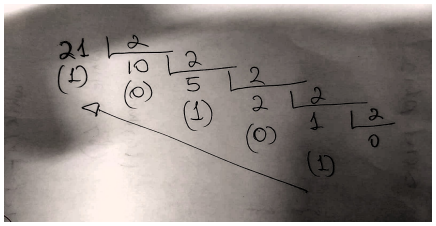
Por exemplo, se $N = 456$, o programa executará os seguintes passos:

```
int N = 456;
printf("%d\n", N % 10); // 6
N = N / 10;             // N = 45
printf("%d\n", N % 10); // 5
N = N / 10;             // N = 4
printf("%d\n", N % 10); // 4
N = N / 10;             // N = 0
```

Se quisermos encontrar os dígitos de um número N em uma base b qualquer, basta substituir o divisor fixo 10 pela base b. O programa abaixo ilustra essa generalização:

```
int N, B;
scanf("%d %d", &N, &B);
while (N > 0) {
    printf("%d\n", N % B);
    N = N / B;
}
```

Por exemplo, para $N = 456$ na base $b = 2$ (binária), o programa exibirá os dígitos binários na ordem inversa (do menos significativo ao mais significativo).



No processo de representação numérica de um número N, seguimos o padrão de:

1. Extrair o dígito menos significativo com a operação de resto ($\%b$).
2. Dividir o número por b (descartando o dígito extraído).
3. Repetir o processo até que o número seja reduzido a zero.

Essa abordagem é aplicável para qualquer base $b > 1$, e a sequência de dígitos extraídos pode ser invertida para obter a representação final do número na base desejada.

Observe que podemos usar um recurso poderoso da programação para resolver esse problema: a recursão. Com a recursão, um problema maior é decomposto em versões menores de si mesmo, permitindo que a mesma lógica seja aplicada repetidamente. Esse processo continua até alcançarmos uma instância suficientemente simples, que pode ser resolvida diretamente de forma trivial.

No problema de exibir os dígitos de um número, temos as seguintes regras:

- Se N ¡ B então N tem apenas um dígito, que é o próprio dígito.
- Caso contrário, N%B é o dígito menos significativo e o novo problema será N/B.

```
void mostra_digito(int N, int B){
    if(N < B){
        printf("%d", N);
    }else{
        printf("%d", N%B);
        mostra_digitos(N/B, B)
    }
}
```

O programa acima mostra os dígitos de um número de maneira invertida, mas se você mudar a ordem dos comandos, ele vai imprimir os dígitos da forma direta.

```
void mostra_digito(int N, int B){
    if(N < B){
        printf("%d", N);
    }else{
        mostra_digitos(N/B, B)
        printf("%d", N%B);
    }
}
```

Exercícios

1. Dado um número inteiro positivo, calcule a soma de seus dígitos.

   Exemplo:

   Entrada: 1234

   Saída: 10 $(1 + 2 + 3 + 4)$

   ```
   void soma_digito(int N, int B){
       if(N < B){
           return N;
       }else{
           return N%B + soma_digitos(N/B, B)
       }
   }
   ```

2. **Maior Dígito** Escreva uma função que retorne o maior dígito de um número inteiro. **Exemplo:**

   Entrada: 56789

   Saída: 9

```
int maior_digito(int N, int B){
    if(N < B){
        return N;
    }else{
        int digito_menos_significativo = N%B;
        int maior_digito = maior_digitos(N/B, B);
        return digito_menos_significativo > maior_digito ?
        digito_menos_significativo : maior_digito;
    }
}
```

3. **Número a partir de uma Lista de Dígitos**

   Crie uma função que receba uma lista de dígitos e retorne o número inteiro correspondente.

   Exemplo:

   Entrada: $[1, 2, 3, 4]$

   Saída: 1234

```
int lista_digito_para_numero(int v[], int size, int B){
    if(size==1){
        return v[0];
    }else{
        int digito_menos_significativo = v[size-1];
        return lista_digito_para_numero(v, size-1, B)*B + digito_menos_significativo;
    }
}
```

Explicação do Programa

O programa transforma uma lista de dígitos ($v[]$) em um número inteiro, considerando que os dígitos estão em uma base B. Vamos explicar o funcionamento de forma bem intuitiva.

Ideia Principal

A função utiliza **recursão** para construir o número aos poucos, começando pelo dígito mais significativo (aquele na posição mais à esquerda da lista). A cada chamada recursiva, ela resolve um "problema menor", até chegar no caso mais simples, quando há apenas um dígito.

Passo a Passo

(a) **Caso base (quando a lista tem apenas um dígito):**

- Se a lista tem tamanho 1 ($size == 1$), o número correspondente é simplesmente o único dígito da lista ($v[0]$).
- Por exemplo, se a lista for $[5]$, o número é 5.

(b) **Caso geral (quando a lista tem mais de um dígito):**

- Pegamos o último dígito da lista, chamado de "dígito menos significativo"($v[size - 1]$).
- Chamamos a função novamente para processar o restante da lista ($size-1$), ignorando o último dígito.
- Multiplicamos o número obtido da parte restante por B (para deslocar os dígitos para a esquerda, na base B) e somamos o "dígito menos significativo".

Exemplo com a Base 10

Vamos ver como o programa funciona para a lista de dígitos $[1, 2, 3, 4]$ na base $B = 10$.

- **Primeira chamada:** $lista\_digito\_para\_numero(v = [1, 2, 3, 4], size = 4, base = 10)$
  - Último dígito: 4
  - Chamamos a função para o restante: $[1, 2, 3]$
  - Resultado: $lista\_digito\_para\_numero([1, 2, 3], 3, 10) \times 10 + 4$
- **Segunda chamada:** $lista\_digito\_para\_numero(v = [1, 2, 3], size = 3, base = 10)$
  - Último dígito: 3
  - Chamamos a função para o restante: $[1, 2]$
  - Resultado: $lista\_digito\_para\_numero([1, 2], 2, 10) \times 10 + 3$
- **Terceira chamada:** $lista\_digito\_para\_numero(v = [1, 2], size = 2, base = 10)$
  - Último dígito: 2
  - Chamamos a função para o restante: $[1]$
  - Resultado: $lista\_digito\_para\_numero([1], 1, 10) \times 10 + 2$
- **Quarta chamada (caso base):** $lista\_digito\_para\_numero(v = [1], size = 1, base = 10)$
  - A lista tem um único dígito: 1. Retorna 1.

Construção do Resultado

Agora, remontamos as chamadas recursivas:

- A última chamada retorna 1.
- Na penúltima, temos: $1 \times 10 + 2 = 12$.
- Na terceira, temos: $12 \times 10 + 3 = 123$.
- Na segunda, temos: $123 \times 10 + 4 = 1234$.

Portanto, o número correspondente à lista $[1, 2, 3, 4]$ na base 10 é 1234.

Intuição

- A função quebra o problema em partes menores, trabalhando com um número cada vez menor até chegar no caso mais simples (um único dígito).
- Depois, reconstrói o número completo multiplicando os valores anteriores pela base B e somando o dígito atual.

4. **Reversão de Dígitos**

   Crie uma função que receba um número inteiro e retorne o número com seus dígitos invertidos.

   Exemplo:

   Entrada: 1234

   Saída: 4321

5. **Substituição de Dígitos** Dado um número inteiro e dois dígitos, substitua todas as ocorrências do primeiro dígito pelo segundo.

   Exemplo:

   Entrada: 12321, substituir 2 por 9 Saída: 19391