

# MC-102 — Aula 08

## Comandos Repetitivos

Instituto de Computação – Unicamp

26 de Março de 2018

# Roteiro

## 1 Exemplos com laços

- Menu de Escolhas
- Representação Binário-Decimal
- Representação Decimal-Binário

## 2 Laços Encaixados

- Equações Lineares

## 3 Exercícios

# Menu de Escolhas

- Em programas de computador, é comum a apresentação de um menu de opções para o usuário.
- Vamos fazer um menu com algumas opções, incluindo uma última para encerrar o programa.

# Menu de Escolhas

O programa terá as seguintes opções:

- **1** - Cadastrar um produto.
- **2** - Buscar informações de produto.
- **3** - Remover um produto.
- **4** - Sair do Programa.

Após realizar uma das operações, o programa volta para o menu.

# Menu de Escolhas

O comportamento do seu programa deveria ser algo como:

```
opcao = 5
while opcao != 4:
    print("1 - Cadastrar um produto")
    print("2 - Buscar informações de produto")
    print("3 - Remover um produto")
    print("4 - Sair do programa")

    opcao = int(input("\nEntre com a opção: "))

#Faça o que for esperado conforme opção digitada
```

# Menu de Escolhas

```
opcao = 5
while opcao != 4:
    print("1 - Cadastrar um produto")
    print("2 - Buscar informações de produto")
    print("3 - Remover um produto")
    print("4 - Sair do programa")

    opcao = int(input("\nEntre com a opção: "))
    if(opcao == 1):
        print("Cadastrando....\n")
    elif(opcao == 2):
        print("Buscando....\n")
    elif(opcao == 3):
        print("Removendo....\n")
    elif(opcao == 4):
        print("Seu programa será encerrado.\n")
    else:
        print("Opção inválida!\n")
```

# Representação Binário-Decimal

- Já sabemos que um computador armazena todas as informações na representação binária.
- É útil saber como converter valores binário em decimal e vice versa.
- Dado um número em binário  $b_n b_{n-1} \dots b_2 b_1 b_0$ , este corresponde na forma decimal à:

$$\sum_{i=0}^n b_i \cdot 2^i$$

- Exemplos:

$$101 = 2^2 + 2^0 = 5$$

$$1001110100 = 2^9 + 2^6 + 2^5 + 2^4 + 2^2 = 512 + 64 + 32 + 16 + 4 = 628$$

- OBS: Em uma palavra no computador um bit é usado para indicar o sinal do número:  $-$  ou  $+$ .

# Representação Binário-Decimal

- Vamos supor que lemos do teclado um inteiro binário.
- Ou seja, ao lermos  $n = 111$  assumimos que este é o número binário (e não cento e onze).
- Como transformar este número no correspondente valor decimal (7 neste caso)??
- Basta usarmos a expressão:

$$\sum_{i=0}^n b_i \cdot 2^i$$



# Representação Binário-Decimal

Um passo importante é conseguir recuperar os dígitos individuais do número:

- Note que  $n \% 10$  recupera o último dígito de  $n$ .
- Note que  $n // 10$  remove o último dígito de  $n$ , pois ocorre a divisão inteira por 10.

Exemplo: Com  $n = 345$ , ao fazermos  $n \% 10$  obtemos 5. E ao fazermos  $n // 10$  obtemos 34.

# Representação Binário-Decimal

- Para obter cada um dos dígitos de um número  $n$  podemos fazer algo como:

Leia  $n$

Enquanto  $n \neq 0$  faça

$\text{digito} = n \% 10$

    Imprima o digito

$n = n // 10$

# Representação Binário-Decimal

O programa abaixo imprime cada um dos dígitos de  $n$  separadamente:

```
n = int(input("Digite um número:"))
while n != 0 :
    digito = n%10
    print(digito)
    n = n//10
```

# Representação Binário-Decimal

- Usar a fórmula  $\sum_{i=0}^n b_i \cdot 2^i$ , para transformar um número em binário para decimal.
- Devemos gerar as potências  $2^0, \dots, 2^n$ , e multiplicar cada potência  $2^i$  pelo  $i$ -ésimo dígito.
  - ▶ Calcular as potência já sabemos (acumuladora **pot** ).
- Para armazenar a soma  $\sum_{i=0}^n b_i \cdot 2^i$  usamos uma outra variável acumuladora **soma**.

```
Leia n
pot = 1
soma = 0
Enquanto n != 0 faça
    digito = n%10
    n = n/10
    soma = soma + (pot*digito)
    pot = pot * 2
```

# Representação Binário-Decimal

Em Python:

```
n = int(input("Digite um número:"))
soma = 0
pot = 1
while n != 0 :
    digito = n%10
    soma = soma + (pot*digito)
    pot = pot*2
    n = n//10
print("Valor em decimal é: ", soma)
```

# Representação Decimal-Binário

- Dado um número em decimal, vamos obter o correspondente em binário.
- Qualquer decimal pode ser escrito como uma soma de potências de 2:

$$5 = 2^2 + 2^0$$

$$13 = 2^3 + 2^2 + 2^0$$

- Nesta soma, para cada potência  $2^i$ , sabemos que na representação em binário haverá um 1 no dígito  $i$ . Exemplo:  $13 = 1101$
- O que acontece se fizermos sucessivas divisões por 2 de um número decimal?

$$13/2 = 6 \text{ com resto } 1$$

$$6/2 = 3 \text{ com resto } 0$$

$$3/2 = 1 \text{ com resto } 1$$

$$1/2 = 0 \text{ com resto } 1$$

# Representação Decimal-Binário

- Dado  $n$  em decimal, fazemos repetidas divisões por 2, obtendo os dígitos do valor em binário:

$$13/2 = 6 \text{ com resto } 1$$

$$6/2 = 3 \text{ com resto } 0$$

$$3/2 = 1 \text{ com resto } 1$$

$$1/2 = 0 \text{ com resto } 1$$

Leia  $n$

Enquanto  $n \neq 0$  faça

$\text{digito} = n \% 2$

    Imprima  $\text{digito}$

$n = n // 2$

# Representação Decimal-Binário

Em Python:

```
n = int(input("Digite um número:"))
while n != 0 :
    digito = n%2
    n = n//2
    print(digito)
```



# Laços Encaixados

- Para resolver alguns problemas, é necessário implementar um laço dentro de outro laço.
- Estes são laços encaixados.

```
for i in range(1,11):  
    for j in range(1,6):  
        print(i, j)
```

- O que será impresso por este programa?

# Laços Encaixados

```
for i in range(1,11):  
    for j in range(1,6):  
        print(i, j)
```

- Fixado um valor para **i** no primeiro laço **for**, começa-se o segundo laço **for**, que varia o valor de **j** entre 1 e 5.
- No final deste segundo laço **for** voltamos para o primeiro laço onde a variável **i** assumirá seu próximo valor. Fixado este valor de **i** começa-se novamente o segundo laço **for**.

# Laços Encaixados

```
for i in range(1,11):  
    for j in range(1,6):  
        print(i, j)
```

- Será impresso:

```
1 1  
1 2  
1 3  
1 4  
1 5  
2 1  
...  
10 1  
10 2  
10 3  
10 4  
10 5
```

# Laços Encaixados: Equações Lineares

- Um uso comum de laços encaixados ocorre quando para cada um dos valores de uma determinada variável, precisamos gerar/checar algo com valores de outras variáveis.

## Problema

Determinar todas as soluções inteiras de um sistema linear como:

$$x_1 + x_2 = C$$

com  $x_1 \geq 0$ ,  $x_2 \geq 0$ ,  $C \geq 0$  e todos inteiros.

# Laços Encaixados: Equações Lineares Inteiras

## Problema

Determinar todas as soluções inteiras de um sistema linear como:

$$x_1 + x_2 = C$$

com  $x_1 \geq 0$ ,  $x_2 \geq 0$ ,  $C \geq 0$  e todos inteiros.

- Uma solução: para cada um dos valores de  $0 \leq x_1 \leq C$ , teste todos os valores de  $x_2$  possíveis e verifique quais deles são soluções.

Para cada  $x_1$  entre 0 e C faça

Para cada  $x_2$  entre 0 e C faça

Se  $x_1 + x_2 = C$  então imprima solução

# Laços Encaixados: Equações Lineares

Em Python:

```
C = int(input("Digite o valor da constante C:"))
for x1 in range(0,C+1):
    for x2 in range(0,C+1):
        if(x1 + x2 == C):
            print(x1, " + ", x2, " = ", C)
```

# Laços Encaixados: Equações Lineares

OBS: Note que fixado  $x_1$ , não precisamos testar todos os valores de  $x_2$ , pois este é determinado como  $x_2 = C - x_1$ .

```
C = int(input("Digite o valor da constante C:"))
for x1 in range(0,C+1):
    x2 = C - x1
    print(x1, " + ", x2, " = ", C)
```

Mas em um caso geral com  $n$  variáveis,

$$x_1 + x_2 + \dots + x_n = C$$

será preciso fixar  $(n - 1)$  variáveis para só então determinar o valor de  $x_n$ .

# Laços Encaixados: Equações Lineares Inteiras

## Problema

Quais são as soluções de  $x_1 + x_2 + x_3 = C$  com  $x_1 \geq 0$ ,  $x_2 \geq 0$ ,  $x_3 \geq 0$ ,  $C \geq 0$  e todas inteiras?

- Uma solução: para cada um dos valores de  $0 \leq x_1 \leq C$ , teste todos os valores de  $x_2$  e  $x_3$  e verifique quais deles são soluções.

Para cada  $x_1$  entre 0 e  $C$  faça

    Para cada  $x_2$  entre 0 e  $C$  faça

        Para cada  $x_3$  entre 0 e  $C$  faça

            Se  $x_1 + x_2 + x_3 = C$  então imprima solução



# Laços Encaixados: Equações Lineares

Em Python:

```
C = int(input("Digite o valor da constante C:"))
for x1 in range(0,C+1):
    for x2 in range(0, C+1):
        for x3 in range(0, C+1):
            if(x1 + x2 + x3 == C):
                print(x1, " + ", x2, " + ", x3, " = ", C)
```

# Laços Encaixados: Equações Lineares Inteiras

- Note que fixado  $x_1$ , o valor máximo de  $x_2$  é  $C - x_1$ .
- Fixados  $x_1$  e  $x_2$ , o valor de  $x_3$  é determinado como  $C - x_1 - x_2$ .
- Podemos alterar o programa com estas melhorias:

```
C = int(input("Digite o valor da constante C:"))
for x1 in range(0,C+1):
    for x2 in range(0, C+1-x1):
        x3 = C - x1 -x2
        print(x1, " + ", x2, " + ", x3, " = ", C)
```

# Exercício

- Na transformação decimal para binário, modifique o programa para que este obtenha o valor binário em uma variável inteira, ao invés de imprimir os dígitos um por linha na tela.
- Dica: Suponha  $n = 7$  (111 em binário), e você já computou  $x = 11$ , para "inserir" o último dígito 1 em  $x$  você deve fazer  $x = x + 100$ . Ou seja, você precisa de uma variável acumuladora que armazena as potências de 10: 1, 10, 100, 1000 etc.

# Exercício

- Implemente um programa que compute todas as soluções de equações do tipo

$$x_1 + x_2 + x_3 + x_4 = C$$

- Melhore o seu programa com as seguintes idéias.
  - ▶ Fixado  $x_1$ , os valores possíveis para  $x_2$  são  $0, \dots, C - x_1$ . Fixado  $x_1$  e  $x_2$ , os valores possíveis para  $x_3$  são  $0, \dots, C - x_1 - x_2$ . Fixados  $x_1, x_2$ , e  $x_3$ , então  $x_4$  é unicamente determinado.