

MC-102 — Aula 09

Comandos Repetitivos

Alexandre M. Ferreira

IC – Unicamp

31/03/2017

Roteiro

- 1 Exemplos com laços
 - Menu de Escolhas
 - Representação Binário-Decimal
 - Representação Decimal-Binário
- 2 Laços Encaixados
 - Equações Lineares Inteiras
 - Mega-Sena
- 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:

```
do{
    printf("1 - Cadastrar um produto\n");
    printf("2 - Buscar informações de produto\n");
    printf("3 - Remover um produto\n");
    printf("4 - Sair do programa\n");

    printf("\nEntre com a opção: ");
    scanf("%d", &opcao);

    //Faça o que for esperado conforme opção digitada

}while(opcao != 4);
```

Menu de Escolhas

```
int main(){
    int opcao;

    do{
        printf("1 - Cadastrar um produto\n");
        printf("2 - Buscar informações de produto\n");
        printf("3 - Remover um produto\n");
        printf("4 - Sair do programa\n");

        printf("\nEntre com a opção: ");
        scanf("%d", &opcao);

        if(opcao == 1)
            printf("Cadastrando....\n\n\n");
        else if(opcao == 2)
            printf("Buscando.....\n\n\n");
        else if(opcao == 3)
            printf("Removendo.....\n\n\n");
        else if (opcao == 4)
            printf("Seu programa será encerrado.\n\n\n");
        else
            printf("Opção Inválida!\n\n\n");

    }while(opcao != 4);
}
```

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 em binário.
- Ou seja, ao lermos $n = 111$ assumimos que este é um 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:

```
int main(){
    int n, digito;

    printf("\n Digite um número:");
    scanf("%d",&n);

    while( n != 0 ){
        digito = n%10;
        printf("%d\n", 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 C:

```
int main(){
    int n, digito, soma, pot;

    printf("Digite um número em binário:");
    scanf("%d",&n);
    soma = 0;
    pot = 1;
    while( n != 0 ){
        digito = n%10;
        n = n/10;
        soma = soma + (digito*pot);
        pot = pot*2;
    }
    printf("Valor em decimal: %d\n", 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 digito

$n = n/2$

Representação Decimal-Binário

Em C:

```
int main(){
    int n, digito;

    printf("Digite um número:");
    scanf("%d",&n);

    while( n != 0 ){
        digito = n%2;
        n = n/2;
        printf("%d\n", digito);
    }
}
```


Laços Encaixados

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

```
int main(){
    int i,j;

    for(i=1;i<=10;i++){
        for(j=1;j<=5;j++){
            printf("%d %d\n",i,j);
        }
    }
}
```

- O que será impresso por este programa?

Laços Encaixados

```
for(i=1;i<=10;i++){  
    for(j=1;j<=5;j++){  
        printf("%d %d\n",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=1;i<=10;i++){  
    for(j=1;j<=5;j++){  
        printf("%d %d\n",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 Inteiras

- Um uso comum de laços encaixados ocorre quando para cada um dos valores de uma determinada variável, precisamos gerar/checar algo sobre os 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 Inteiras

Em C:

```
int main(){
    int C;
    int x1, x2;

    printf("Digite o valor de C:");
    scanf("%d", &C);

    for(x1 = 0; x1 <= C; x1++){
        for(x2 = 0; x2 <= C; x2++){
            if(x1 + x2 == C)
                printf("%d + %d = %d\n",x1,x2,C);
        }
    }
}
```

Laços Encaixados: Equações Lineares Inteiras

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

```
int main(){
    int C;
    int x1, x2;

    printf("Digite o valor de C:");
    scanf("%d", &C);

    for(x1 = 0; x1 <= C; x1++){
        x2 = C - x1;
        printf("%d + %d = %d\n", 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 Inteiras

Em C:

```
int main(){
    int C;
    int x1, x2, x3;

    printf("Digite o valor de C:");
    scanf("%d", &C);

    for(x1 = 0; x1 <= C; x1++){
        for(x2 = 0; x2 <= C; x2++){
            for(x3 = 0; x3<= C; x3++){
                if(x1 + x2 + x3 == C)
                    printf("%d + %d + %d = %d\n",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:

```
int main(){
    int C;
    int x1, x2, x3;

    printf("Digite o valor de C:");
    scanf("%d", &C);

    for(x1 = 0; x1 <= C; x1++){
        for(x2 = 0; x2 <= C - x1; x2++){
            x3 = C - x1 - x2;
            printf("%d + %d + %d = %d\n",x1,x2,x3,C);
        }
    }
}
```

Laços Encaixados: Mega-Sena

- Na Mega-Sena, um jogo consiste de 6 números distintos com valores entre 1 e 60.

Problema

Imprimir todos os jogos possíveis da Mega-Sena.

Laços Encaixados: Mega-Sena

- Partimos da mesma idéia dos dados: gerar todos os possíveis valores para cada um dos 6 números do jogo.

```
int main(){
    int d1, d2, d3, d4, d5, d6;

    for(d1 = 1; d1 <= 60; d1++)
        for(d2 = 1; d2 <= 60; d2++)
            for(d3 = 1; d3 <= 60; d3++)
                for(d4 = 1; d4 <= 60; d4++)
                    for(d5 = 1; d5 <= 60; d5++)
                        for(d6 = 1; d6 <= 60; d6++)
                            printf("%d, %d, %d, %d, %d, %d\n",d1,d2,d3,d4,d5,d6);
}
```

- Qual a saída deste programa? Ele está correto?

Laços Encaixados: Mega-Sena

```
int main(){
    int d1, d2, d3, d4, d5, d6;

    for(d1 = 1; d1 <= 60; d1++)
        for(d2 = 1; d2 <= 60; d2++)
            for(d3 = 1; d3 <= 60; d3++)
                for(d4 = 1; d4 <= 60; d4++)
                    for(d5 = 1; d5 <= 60; d5++)
                        for(d6 = 1; d6 <= 60; d6++)
                            printf("%d, %d, %d, %d, %d, %d\n", d1, d2, d3, d4, d5, d6);
}
```

- As primeiras linhas impressas por este programa serão:

```
1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 2
1, 1, 1, 1, 1, 3
1, 1, 1, 1, 1, 4
1, 1, 1, 1, 1, 5
1, 1, 1, 1, 1, 6
1, 1, 1, 1, 1, 7
1, 1, 1, 1, 1, 8
1, 1, 1, 1, 1, 9
```

Laços Encaixados: Mega-Sena

- O programa anterior repete números, portanto devemos remover repetições.

```
int main(){
    int d1, d2, d3, d4, d5, d6;

    for(d1 = 1; d1 <= 60; d1++)
        for(d2 = 1; d2 <= 60; d2++)
            for(d3 = 1; d3 <= 60; d3++)
                for(d4 = 1; d4 <= 60; d4++)
                    for(d5 = 1; d5 <= 60; d5++)
                        for(d6 = 1; d6 <= 60; d6++)
                            if( (d1!=d2) && (d1!=d3) &&.....)
                                printf("%d, %d, %d, %d, %d, %d\n",d1,d2,d3,d4,d5,d6)
}
```

- Após incluir todos os testes para garantir que os números são distintos, temos a solução?

Laços Encaixados: Mega-Sena

- Não temos uma solução válida, pois o programa irá imprimir jogos como:

12, 34, 8, 19, 4, 45

34, 12, 8, 19, 4, 45

34, 12, 19, 8, 4, 45

- Todos estes jogos são um único jogo: 4, 8, 12, 19, 34, 45.
- Podemos assumir que um jogo é sempre apresentado com os números em ordem crescente.
- Dado que fixamos o valor de **d1**, **d2** necessariamente é maior que **d1**. Após fixar **d1** e **d2**, **d3** deve ser maior que **d2** etc.

Laços Encaixados: Mega-Sena

Solução correta:

```
int main(){
    int d1, d2, d3, d4, d5, d6;

    for(d1 = 1; d1 <= 60; d1++)
        for(d2 = d1 + 1; d2 <= 60; d2++)
            for(d3 = d2 + 1; d3 <= 60; d3++)
                for(d4 = d3 + 1; d4 <= 60; d4++)
                    for(d5 = d4 + 1; d5 <= 60; d5++)
                        for(d6 = d5 + 1; d6 <= 60; d6++)
                            printf("%d, %d, %d, %d, %d, %d\n", d1, d2, d3, d4, d5, d6);
}
```


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.