

# MC-102 — Aula 08

## Comandos Repetitivos

Instituto de Computação – Unicamp

13 de Março de 2015

# Roteiro

- 1 Laços Encaixados
  - Números Primos
  - Dados
  - Mega-Sena
- 2 Exercícios

# Laços Encaixados: Primos

- A geração de números primos é uma parte fundamental em sistemas criptográficos como os utilizados em *internetbanking*.
- Já sabemos testar se um determinado número é ou não primo.
- Imagine que agora queremos imprimir os  $n$  primeiros números primos.
- O que podemos fazer?

## Laços Encaixados: Primos

- O programa abaixo verifica se o valor na variável **candidato** corresponde a um primo:

```
divisor = 2;
eprimo = 1;
while( (divisor <= candidato/2) && (eprimo) ){
    if(candidato % divisor == 0)
        eprimo = 0;
    divisor++;
}
if(eprimo){
    printf("%d, ", candidato);
}
```

# Laços Encaixados: Primos

Em um laço externo usamos uma variável contadora **primosImpressos**, que contará o número de primos impressos durante a execução do laço.

```
while(primosImpressos < n){  
  
    //trecho do código anterior que  
    //checa se candidato é ou não é primo  
  
    if(eprimo){  
        printf("%d, ", candidato);  
        primosImpressos++;  
    }  
    candidato++; //Testa próximo número candidato a primo  
}
```

# Laços Encaixados: Primos

Podemos usar o trecho de código que checa se um número é primo ou não.

```
int main(){
    int divisor, candidato, primosImpressos, n, eprimo;
    printf("\n Digite um numero inteiro positivo:");
    scanf("%d",&n);

    candidato = 2;
    primosImpressos = 0;
    while(primosImpressos < n){

        //trecho do código que checa
        //se candidato é ou não é primo

        if(eprimo){
            printf("%d, ", candidato);
            primosImpressos++;
        }
        candidato++; //Testa próximo número candidato a primo
    }
}
```

# Laços Encaixados: Primos

Código completo:

```
int main(){
    int divisor, candidato, primosImpressos, n, eprimo;

    printf("\n Digite um numero inteiro positivo:");
    scanf("%d",&n);

    candidato = 2;
    primosImpressos = 0;
    while(primosImpressos < n){
        divisor = 2;
        eprimo=1;
        while( (divisor <= candidato/2) && (eprimo) ){
            if(candidato % divisor == 0)
                eprimo = 0;
            divisor++;
        }
        if(eprimo){
            printf("%d, ", candidato);
            primosImpressos++;
        }
        candidato++; //Testa próximo número candidato a primo
    }
}
```

## Laços Encaixados: Primos

- Note que o número 2 é o único número par que é primo.
- Podemos alterar o programa para sempre imprimir o número 2:

```
int main(){
    int divisor, candidato, primosImpressos, n, eprimo;

    printf("\n Digite um numero inteiro positivo:");
    scanf("%d",&n);

    if(n > 0){
        printf("%d, ", 2);
        .....
    }
```

## Laços Encaixados: Primos

- Podemos alterar o programa para testar apenas números ímpares como candidatos a primo:

```
candidato = 3;
primosImpressos = 1;
while(primosImpressos < n){
    divisor = 2;
    eprimo=1;
    while( (divisor <= candidato/2) && (eprimo) ){
        if(candidato % divisor == 0)
            eprimo = 0;
        divisor++;
    }
    if(eprimo){
        printf("%d, ", candidato);
        primosImpressos++;
    }
    candidato = candidato + 2;//Testa próximo número candidato a primo
}
```

# Laços Encaixados: Primos

- Além disso não precisamos mais testar os divisores que são pares.
- Se o candidato só pode ser um número ímpar, ele não pode ser divisível por um número par, pois seria divisível por 2 também.
- Portanto basta testar divisores ímpares.

# Laços Encaixados: Primos

```
int main(){
    int divisor, candidato, primosImpressos, n, eprimo;
    printf("\n Digite um numero inteiro positivo:");
    scanf("%d",&n);
    if(n > 0){
        printf("%d, ", 2);
        candidato = 3;
        primosImpressos = 1;
        while(primosImpressos < n){
            divisor = 3;
            eprimo=1;
            while( (divisor <= candidato/2) && (eprimo) ){
                if(candidato % divisor == 0)
                    eprimo = 0;
                divisor = divisor + 2;
            }
            if(eprimo){
                printf("%d, ", candidato);
                primosImpressos++;
            }
            candidato = candidato + 2;//Testa próximo número candidato a primo
        }
    }
}
```

# Laços Encaixados: Dados

## Problema

Imprimir todas as possibilidades de resultados ao se jogar 4 dados de 6 faces.

- Para cada possibilidade do primeiro dado, devemos imprimir todas as possibilidades dos 3 dados restantes.
- Para cada possibilidade do primeiro e segundo dado, devemos imprimir todas as possibilidades dos 2 dados restantes....
- Você consegue pensar em uma solução com laços aninhados?

# Laços Encaixados: Dados

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

    printf("\nD1 D2 D3 D4\n");
    for(d1 = 1; d1 <= 6; d1++)
        for(d2 = 1; d2 <= 6; d2++)
            for(d3 = 1; d3 <= 6; d3++)
                for(d4 = 1; d4 <= 6; d4++)
                    printf("%d %d %d %d\n", d1, d2, d3, d4);
}
```

# 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.
- Problema: Os números do jogo devem ser distintos.

# 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++)
                            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

- Na verdade, 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**. E com **d1** e **d2** fixados, **d3** é 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

- Faça um programa que leia um número  $n$  e imprima  $n$  linhas na tela com o seguinte formato (exemplo se  $n = 6$ ):

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

# Exercício

- Faça um programa que leia um número  $n$  e imprima  $n$  linhas na tela com o seguinte formato (exemplo se  $n = 6$ ):

```
+ * * * * *
* + * * * *
* * + * * *
* * * + * *
* * * * + *
* * * * * +
```

# Exercício

- Um jogador da Mega-Sena é supersticioso, e só faz jogos em que o primeiro número do jogo é par, o segundo é ímpar, o terceiro é par, o quarto é ímpar, o quinto é par e o sexto é ímpar. Faça um programa que imprima todas as possibilidades de jogos que este jogador supersticioso pode jogar.