MC-102 — Aula 11
Comandos Repetitivos e Vetores

Instituto de Computação – Unicamp

11 de Abril de 2013

Roteiro

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 depois:

```
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

```
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 = 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: 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.

Informações Extras: break, continue e goto

O comando **break** faz com que a execução de um laço seja terminada, passando a execução para o o próximo comando depois do final do laço.

```
int i;
for(i = 1; i<= 10 ; i++){
          if(i >= 5)
             break;
          printf("%d\n",i);
}
printf("Terminou o laço");
```

Informações Extras: Laços e o comando continue

O comando **continue** faz com que a execução de um laço seja alterada para final do laço.

```
int i;
for(i = 1; i<= 10 ; i++){
          if(i == 5)
             continue;
          printf("%d\n",i);
}
printf("Terminou o laço");
```

Informações Extras: Laços e o comando continue

O comando **goto** faz com que a execução seja desviada para um ponto específico no programa. É uma estrutura de repetição primitiva, considerada pela maioria uma má prática de programação.

```
marca1:
...
sentença(s);
...
goto marca1;
```

Informações Extras: Laços e o comando continue

```
int main(int argc, char *argv[]) {
        int numero = 1;
        inicio_repeticao:
                if (numero > 10) {
                        goto fim_repeticao;
                }
                printf("%d " , numero);
                numero++;
                goto inicio_repeticao;
        fim_repeticao:
        return 0;
}
```

Vetores

Como armazenar 3 notas?

```
float nota1, nota2, nota3;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);
printf("Nota do aluno 2: ");
scanf("%f", &nota2);
printf("Nota do aluno 3: ");
scanf("%f", &nota3);
```

Vetores

Como armazenar 100 notas?

```
float nota1, nota2, nota3, /* .... */ nota100;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);
printf("Nota do aluno 2: ");
scanf("%f", &nota2);

/* ... */

printf("Nota do aluno 100: ");
scanf("%f", &nota100);
```

Vetores — Definição

Coleção de variáveis do mesmo tipo referenciada por um nome comum.

(Herbert Schildt)

Características:

- Acesso por meio de um índice inteiro.
- Posições contíguas na memória.
- Tamanho pré-definido.
- Índices fora dos limites podem causar comportamento anômalo do programa.

Declaração de um vetor

<tipo> identificador [<tamanho do vetor>];

Exemplo

```
float notas[100];
int medias[100];
```

Usando um vetor

Após declarada uma variável do tipo vetor, pode-se acessar uma determinada posição do vetor utilizando um valor inteiro.

identificador [<posição>];

- O acesso de um vetor em uma posição específica tem o mesmo comportamento que uma variável simples.
- A primeira posição de um vetor tem índice 0.
- A última posição de um vetor tem índice *<tamanho do vetor>* - 1.

Exemplo

```
int nota[10];
int a;
nota[5] = 95;
a = nota[5];
```

Usando um vetor

identificador [<posição>];

- Você pode usar valores inteiros para acessar uma posição do vetor.
- O valor pode ser inclusive uma variável inteira.

Exemplo

```
int g, vet[10];
for(g=0; g<10; g++)
    vet[g]=5*g;
```

Vetores

- Na memória:

```
int d;
int vetor[5];
int f;
```

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-

Vetores

- Ao executar `vetor[3]=10;`:

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
					10		

Vetores

- O que ocorre se forem executados os comandos:

```
vetor[5]=5;
vetor[-1]=1;
```

Vetores

- Ao executar

```
vetor[3]=10;
vetor[5]=5;
vetor[-1]=1;
```

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
	1				10		5

- Isto irá causar um erro no seu programa pois você está alterando valores de outras variáveis.
- Em muitos casos o seu programa será encerrado (**Segmentation Fault**).

Exercício

1. Faça um programa que lê 10 números inteiros e calcula a média.
2. Faça um programa que lê 100 números inteiros e verifica se há números repetidos na seqüência lida.