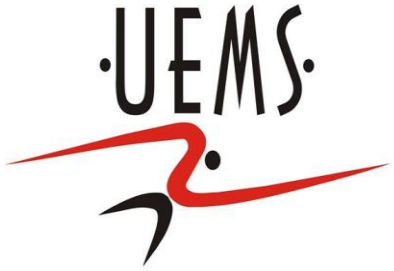


PCI- Estrutura de Repetição II

Profa. Mercedes Gonzales Márquez

Estrutura de Repetição

Exemplo 1: Imprimir os n primeiros números inteiros positivos.	Exemplo 2: Faça um algoritmo que determine os quadrados de um conjunto de números inteiros positivos.

int i, n;
scanf("%d",&n);
for (i=1; i<=n;i++)
printf("\n %d",i);
	int num;
	scanf("%d",&num);
	for (; num>0;){
	printf("%d\n",num*num);
	scanf("%d",&num);
	}

Estrutura de Repetição

Exemplo 3: Imprimir os divisores de um número natural n

Para isso, passamos por todos os números naturais i até o número n e verificamos se cada um deles é um divisor de n. Se o resto da divisão de n por i for zero, então estamos diante de um divisor de n.

```
int i, n;
printf("Entre com um inteiro positivo: ");
scanf("%d", &n);
for (i = 1; i <= n; i++)
    if (n % i == 0) /* Se o resto for zero, então i é divisor*/
        printf("%d\n", i);
```

Estrutura de Repetição

Sobre o exemplo 3 algumas observações se fazem necessárias:

(A) Se um número inteiro X possui um divisor Y menor que sua raiz quadrada, o quociente da divisão de X por Y será maior que a raiz quadrada de X e será, também, um divisor de X.

(B) Se um número inteiro X possui um divisor Y igual a sua raiz quadrada, o quociente da divisão de X por Y será o próprio divisor Y.

Estrutura de Repetição

A implementação anterior é uma das implementações mais ineficientes possíveis, pois vários dos candidatos da lista [2; 3; : : : ; n – 1] podem ser definitivamente descartados. Por exemplo, nenhum número
maior que n/2 pode ser divisor de n, pois isso significaria que n também tem um divisor entre 1 e 2. Ainda mais, se só precisamos verificar se n é primo (e não achar todos os seus divisores), basta verificar os possíveis divisores d <= *raiz(n)* (pois, se d for realmente divisor, n/d>=raiz(n) também será divisor).
Mas, no momento, para o propósito atual, vamos nos contentar com a maneira ineficiente.

Estrutura de Repetição

Exemplo 4: Faça um programa que dado um inteiro positivo N, imprima se ele é um número primo ou não.

Para testar se um número n é primo devemos verificar se ele tem um divisor não-trivial, isto é, um que não seja igual a 1 ou n. Assim, verificamos entre todos os candidatos (a princípio, [2; 3; : : : ; n – 1], mas essa lista ainda pode ser bem refinada, como comentado no exercício anterior) se há algum divisor de n. Se não acharmos nenhum, é porque n é primo.

Estrutura de Repetição

Exemplo 4: Faça um programa que dado um inteiro positivo n, imprima se ele é um número primo ou não.

```
int n, i, /* candidato a divisor */
divisores=0; /* contador de divisores */
printf("Entre com um inteiro positivo: ");
scanf("%d", &n);
for (i = 2; i < n; i++) /* testa todos os candidatos a divisor */
   if (n % i == 0)
      divisores++;
if (divisores == 0)   /* não achamos nenhum divisor dentre os
                         possíveis candidatos , então é primo */
    printf("O numero %d e´ primo\n", n);
else
    printf ("O numero %d nao e´ primo\n", n);
```

Estrutura de Repetição – Comandos break e continue

Comando break

O comando break faz com que a execução de um laco de repetição seja finalizada, passando a execução para o próximo comando após o laco. Exemplo:

```
int i;
for (i = 1; i <= 5; i++) {
    if (i > 3)
        break;
    printf("%d\t", i);
}
printf("Fim\n");
```

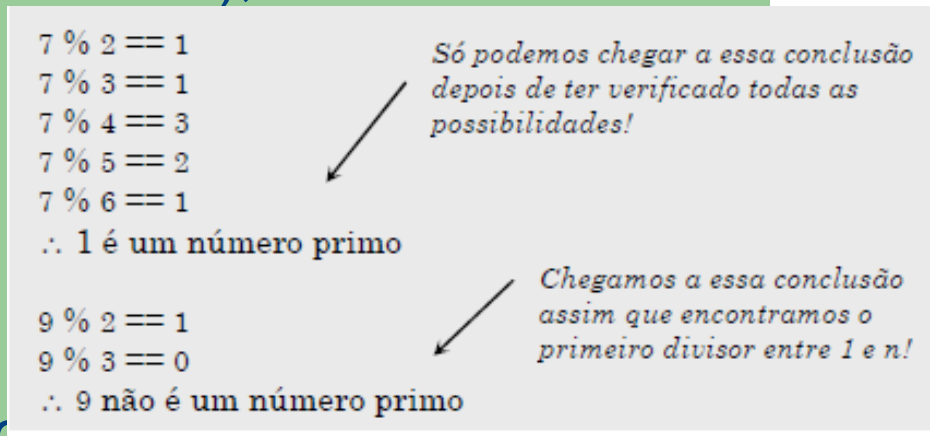
O que será impresso?

1 2 3 Fim

Estrutura de Repetição
Comandos break e continue

Exemplo 1: Faça um programa que dado um inteiro positivo n, imprima se ele é um número primo ou não.

```
int n, i;
printf("\nDigite um número natural: ");
scanf("%u", &n);
for(i=2; i<=n-1; i++)
     if( n%i == 0 )
         break;
if( i==n )
    printf("\nO número é primo");
else
   printf("\nO número não é primo");
```



Estrutura de Repetição – Comandos break e continue

Comando continue

O comando continue faz com que a execução da iteração corrente do laco de repetição seja finalizada, passando a execução para a próxima iteração do laco. Exemplo:

```
int i;
for (i = 1; i <= 5; i++) {
    if (i == 3)
        continue;
    printf("%d\t", i);
}
printf("Fim\n");
```

O que será impresso?

1 2 4 5 Fim

Combinação de Estruturas: Estrutura Condicional dentro de Repetições

- Exemplo 1: Escrever um programa que receba dois números inteiros positivos, e determine o produto dos mesmos, utilizando o seguinte método de multiplicação:

• dividir, sucessivamente, o primeiro número por 2, até que se obtenha 1 como quociente;
• paralelamente, dobrar, sucessivamente, o segundo número;
• somar os números da segunda coluna que tenham um número ímpar na primeira coluna. O total obtido é o produto procurado.

Exemplo: 9 x 6

9	6→	6
4	12	
2	24	
1	48→	+48
		54

Combinação de Estruturas: Estrutura Condicional dentro de Repetições

```
int i,a,b,pro=0;
printf (“Informe dois inteiros: “);
scanf (“%d %d”, &a,&b);
while (a>=1) {
   if (a%2!=0)
          pro+=b; /* Acumulador*/
   a/=2; /* atualização de a*/
   b*=2;/* atualização de b*/
}
printf (“O produto de %d e %d e’ %d”,a,b,pro);
```

Combinação de Estruturas- Estrutura Repetição dentro de Condicional

Agora veremos o contrário da combinação anterior, isto é, um comando de repetição no escopo do comando de desvio condicional.

Exemplo 2: Imprimir o Maximo Divisor Comum (MDC) entre dois números dados.

Combinação de Estruturas- Estrutura Repetição dentro de Condicional

O conceito matemático de máximo divisor comum entre dois números dados a e b envolve a fatoração de cada numero como um produto de fatores primos, escolhendo-se os fatores primos que se repetem com a potência mínima.

Exemplo: Calcular o MDC entre 72 e 135.

72 = 2*2*2*3*3

135 = 3*3*3 *5

Da teoria conclui-se que o MDC entre 72 e 135 é 3*3, pois o 3 e o único fator primo que se repete em ambos os números de entrada, e a menor potência comum é 2.

Combinação de Estruturas-
Estrutura Repetição dentro de Condicional

O máximo divisor comum entre dois números naturais não nulos e distintos é igual ao máximo divisor entre o menor e o resto da divisão do maior pelo menor.

Em símbolos, se x e y são números naturais tais que $x > y > 0$, e se

x	y
r	q

então $\boxed{mdc(x,y) = mdc(y,r)}$.

Baseado nesse resultado temos que:
A ideia é dividir o maior número pelo menor, e depois fazer divisões sucessivas do último divisor pelo último resto, até obtermos uma divisão exata.
Exemplo:

1320 | 35	35 | 25	25 | 10	10 | 5
25 | 37	10 | 1	5 | 2	0 | 2

Estrutura Repetição dentro de Condicional

```
Algoritmo <mdc>
inteiro:x, y, resto
Inicio
leia (x,y)
se ((x <> 0) e (y <> 0)) então
      resto ← mod(x,y)
      enquanto (resto <> 0) faça
            x ← y;
            y ← resto ;
            resto ← mod(x,y)
      fim enquanto
      escreva( “mdc =”, y)
senão
      escreva (“ algoritmo nao funciona para entradas nulas.”)
fim
```

Combinação de Estruturas- Estrutura Repetição Aninhadas

Vamos ver problemas para os quais os algoritmos exigem o aninhamento de repetições.

Exemplo 3: Imprimir as tabuadas do 1 ao n.

Estrutura Repetição Aninhadas

```
inteiro: i , j,n
Inicio
i ← 1
leia (n)
enquanto (i <= n) faça
      j ← 1
      enquanto  (j <= 10) faça
            escreva ( i ,'x' , j ,“= ”, i*j ) ; ( comando mais interno )
            j ← j + 1
      fim enquanto
      i ← i + 1
fim enquanto
fim
```

Combinação de Estruturas- Estrutura Repetição Aninhadas

Exemplo 4: Imprimir o valor do fatorial de todos os números entre 1 e n, sendo n fornecido pelo usuário.

O fatorial de n e assim definido:
n= n x (n -1) x (n - 2) x ... x 3 x 2 x1

No exercício 6 vimos o algoritmo para o fatorial de um único número n e agora veremos o algoritmo para o fatorial de um conjunto de números.

Estrutura Repetição Aninhadas

```
/* Fatorial de um único inteiro n*/
inteiro: i , n, fat
inicio
leia (n)
fat ← 1 ( inicializacao do acumulador )
i ← n
enquanto (i >= 1) faça
      fat ← fat  * i
      i ← i – 1
fim enquanto
escreva( fat )
fim
```

Estrutura Repetição Aninhadas

```
Algoritmo <fatorial>
/* Fatorial de um conjunto de inteiros*/

inicio
leia (n)
cont← 1
enquanto (cont<=n) faça
      fat ←1 ( inicializacao do acumulador )
      i ← cont
      enquanto (i >= 1) faça
            fat ← fat  * i
            i ← i – 1
      fim enquanto
      escreva( fat )
      cont ←cont+1
fim enquanto
fim
```

Combinação de Estruturas- Estrutura Repetição Aninhadas

O algoritmo funciona, mas e extremamente ineficiente e repleto de cálculos redundantes. Perceba a seguinte propriedade de fatorial:

$$n!=nx(n-1)!$$

Ou seja quando você calcular o fatorial de n você pode aproveitar o calculo do fatorial do número anterior, isto é, n-1.

Assim sendo, temos uma versão mais eficiente do algoritmo anterior.

Combinação de Estruturas- Estrutura Repetição Aninhadas

```
Algoritmo <fatorial>
inteiro: i , n, fat, cont
inicio
leia (n)
cont← 1
fat← 1
enquanto (cont<=n) faça
      fat ← fat * cont
      escreva ( fat )
      cont ← cont + 1;
fim enquanto
fim
```