



# Estruturas de Dados (ED)

**Aula Prática 2 (Parte 2):**

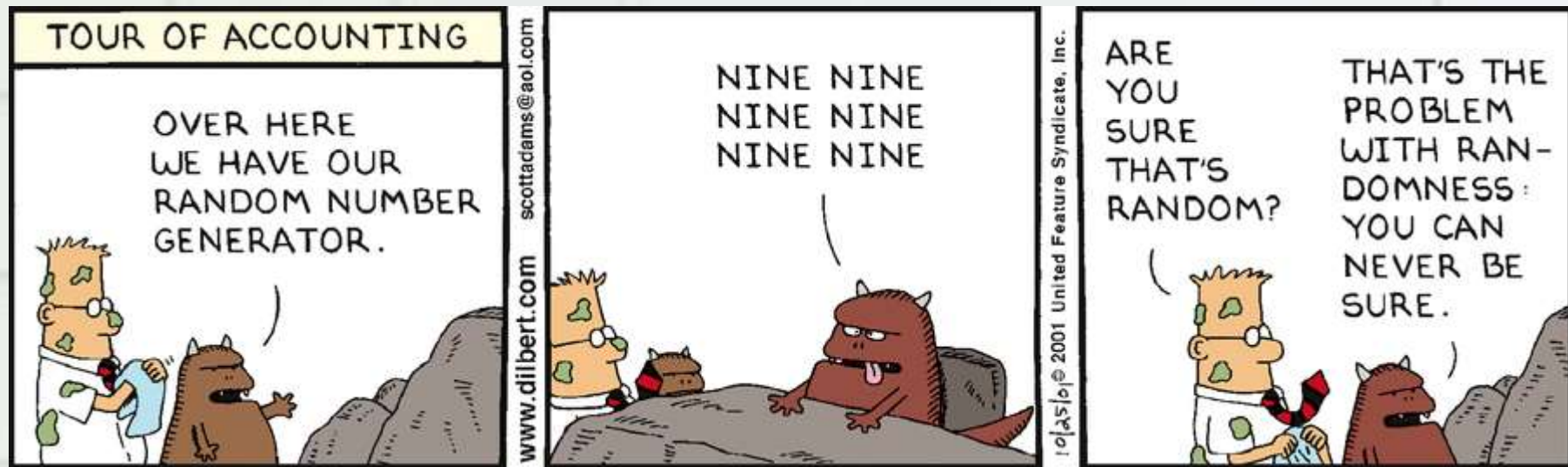
**Função rand() e comandos especiais**

**Comandos de Decisão / Estrutura de Repetição (Exercícios)**

**Prof. Daniel Baraldi Sesso**

**[daniel.sesso@docente.unip.br](mailto:daniel.sesso@docente.unip.br)**

# Números aleatórios



# Números aleatórios

- Sequências de números aleatórios (ou seja, imprevisíveis) são úteis em muitas aplicações.
- São úteis, em particular, para gerar dados de teste de programas.
  - ❑ Números verdadeiramente aleatórios são muito difíceis de obter (veja um exemplo de obtenção de número aleatório em [random.org](https://www.random.org)); por isso, devemos nos contentar com **números pseudoaleatórios**, gerados por algoritmos.
- Para simplificar a linguagem, omitiremos o "pseudo" no que segue.

## Função rand()

- A função **rand()** (o nome é uma abreviatura de *random*) da biblioteca **stdlib.h** gera números inteiros aleatórios.
- Cada invocação da função produz um inteiro aleatório no intervalo fechado **0 .. RAND\_MAX**.
  - ❑ A macro **RAND\_MAX** está definida na interface **stdlib.h** e é menor ou igual a **INT\_MAX**
    - **INT\_MAX** vale  $2^{31}-1$ , ou seja, 2147483647.
  - ❑ Para aplicações pouco exigentes, podemos supor que os números gerados por **rand** são mais ou menos uniformemente distribuídos no intervalo **0 .. RAND\_MAX**, ou seja, que cada número do intervalo tem mais ou menos a mesma probabilidade de ser gerado.

# Inteiros aleatórios

- Como obter um número inteiro aleatório num dado intervalo  $0 \dots N-1$ ?
- A primeira ideia é usar a expressão `rand() % N` (que dá o resto da divisão de `rand()` por `N`).
- Essa ideia seria razoável se `rand` produzisse números verdadeiramente aleatórios.
- Como os números produzidos por `rand` são apenas pseudoaleatórios, os últimos dígitos de cada número podem não ser aleatórios, e assim o resto da divisão por `N` pode não ser aleatório (poderia ser sempre ímpar, por exemplo).

# Inteiros aleatórios

```
secreto = rand() % 26 + 'a';
```

- Na expressão acima (do exemplo da aula anterior), **rand() % 26** resulta o resto da divisão do valor de **rand()** por 26.
- O resultado é um número inteiro entre 0 e 25 (resto da divisão por 26).
- A este número é somado o caractere a para gerar uma letra minúscula aleatória.

## Comandos especiais

**break, continue e goto**

# Comandos break e continue

- Os comandos **break** e **continue** são instruções que devem pertencer ao corpo de um laço **for**, **while** ou **do-while** e não podem ser utilizados em outras partes de um programa.
- O comando **break** tem um segundo uso junto ao comando **switch** (como vimos na aula anterior).
- O comando **break** causa a saída imediata de um laço; após isso, **o controle passa para a próxima instrução.**
- Se a instrução **break** pertencer a um conjunto de laços aninhados, **afetará somente o laço ao qual pertence e os laços internos a ele.**



# Comandos break e continue

- O comando **continue** força a próxima iteração do laço e pula o código que estiver abaixo.
- Nos laços **while** e **do-while**, um comando continue faz com que o controle do programa avalie imediatamente a expressão de teste e depois continue o processo do laço.
- No laço **for**, é executada a expressão de incremento e, em seguida, o teste.
- Ou seja, o **break** faz todo o laço parar.
- Já o **continue**, faz somente com que a iteração atual pare, pulando pra próxima iteração.

# Comandos break e continue

## Exemplo 1:

*Ache o primeiro número, entre 1 e 1 milhão que é divisível por 11, 13 e 17.*

Isso quer dizer que temos que encontrar o menor número que, quando dividimos ele por 11, por 13 e por 17 dê resto da divisão nulo.

Uma possível solução é a seguinte:

```
#include <stdio.h>

int main()
{
    int count,
        numero=0;

    for(count=1 ; count<=1000000 ; count++)
    {
        if(numero == 0)
            if((count%11==0) && (count%13==0) && (count%17==0))
                numero=count;
    }
    printf("O numero e: %d", numero);
}
```

## Comandos break e continue

- A variável **count** é a que irá contar de 1 até 1 milhão.
- O número que queremos será armazenado em **numero** e inicializamos com valor 0.
- Enquanto **numero** tiver valor 0, é porque o número que estamos procurando ainda não foi achado, pois quando ele for achado a variável **numero** irá mudar de valor e passará a ser o número que queremos.
- Note que esse valor ficará sempre armazenado na variável **numero**, pois agora o primeiro teste condicional nunca mais será satisfeito.

## Comandos break e continue

- Você vai ver que o número em questão é 2431.
- E o que acontece depois que descobrimos esse número?
- Ora, o laço continua a contar de 2431 até 1000000 desnecessariamente.
- Então o computador está gastando processamento à toa.

# Comandos break e continue

## Exemplo 1:

*Ache o primeiro número, entre 1 e 1 milhão que é divisível por 11, 13 e 17.*

Isso quer dizer que temos que encontrar o menor número que, quando dividimos ele por 11, por 13 e por 17 dê resto da divisão nulo.

```
#include <stdio.h>
int main()
{
    int count,
        numero=0;

    for(count=1 ; count<=1000000 ; count++)
    {
        if(numero == 0)
            if((count%11==0) && (count%13==0) && (count%17==0)){
                numero=count;
                break;
            }
    }
    printf("O numero e: %d", numero);
}
```

# Comandos break e continue

## Exemplo 2:

*Faça um aplicativo em C que some todos os números, de 1 até 100, exceto os múltiplos de 5.*

Fazemos o laço for percorrer de 1 até 100.

Testamos se cada número deixa resto 0 quando dividido por 5.

Caso deixe, é porque é múltiplo de 5 e não devemos somar.

**Para não somar, simplesmente pulamos essa iteração.**

Porém, se não for múltiplo de 5, é porque a iteração não foi pulada, ela continua, então vamos somar esse número na soma total.

# Comandos break e continue

## Exemplo 2:

*Faça um aplicativo em C que some todos os números, de 1 até 100, exceto os múltiplos de 5.*

```
#include <stdio.h>

int main()
{
    int count,
        soma;

    for(count=1, soma=0 ; count<=100 ; count++)
    {
        if( count%5 == 0)
            continue;

        soma = soma + count;
    }

    printf("Soma %d", soma);
}
```

# Comando goto

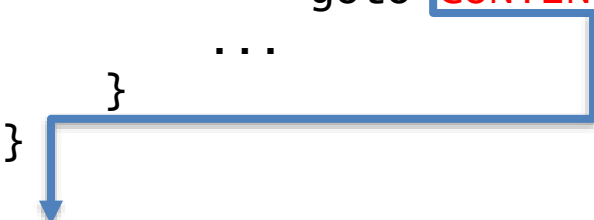
- O comando **goto** está disponível em C para fornecer alguma compatibilidade com outras linguagens de programação, mas sua utilização é desaconselhada.
- Esse comando causa o **desvio do controle do programa** para a instrução seguinte ao rótulo com o nome indicado, lembrando que um rótulo é um nome seguido de dois pontos (:).
- O comando **goto** e o rótulo correspondente devem estar no corpo da mesma função.
  - ❑ Em princípio, você nunca precisará usar **goto** em seus programas, mas, se tiver um programa em outra linguagem que deva ser traduzido para C rapidamente (p.ex. linguagem *Assembly*), o comando goto poderá ajudá-lo.



# Comando goto

Uma aplicação citada por Brian Kernighan & Dennis Ritchie (no livro “**The C Programming Language**”. 2ed. p.57–58.) é para escapar de laços muito aninhados:

```
for (i=0; i<P; i++) {  
    for (j=0; j<Q; j++) {  
        for (k=0; k<R; k++) {  
            ...  
            if (condicao)  
                goto CONTINUACAO;  
            ...  
        }  
    }  
}  
CONTINUACAO:  
    printf("o programa continua daqui...");  
    ...
```



Há um “salto” para outro ponto do programa

# Dúvidas



# EXERCÍCIOS



## Exercício 1

---

Faça um programa que leia três números inteiros e encontra o menor deles.

***Sugestão:*** Sejam 3 números  $A$ ,  $B$  e  $C$ . A ideia principal é: verificar se  $A$  é menor que  $B$  e  $C$  e se não for, verificar entre  $B$  e  $C$ .

## Exercício 2

Faça um programa que leia três números inteiros e colocá-los em ordem crescente.

**Sugestão:** *Sejam 3 números A, B e C. A ideia principal é:*

- ✓ *armazenar em A o menor valor*
- ✓ *armazenar em B o valor intermediário*
- ✓ *armazenar em C o maior valor*

## Exercício 3

---

Faça um programa que apresente se o número que o usuário digitou é divisível por 3 e por 5 ao mesmo tempo.

## Exercício 4

---

Num determinado Estado, para transferências de veículos, o DETRAN cobra uma taxa de 2,5% para carros fabricados antes de 2010 e uma taxa de 3,5% para os fabricados de 2010 em diante, taxa esta incidindo sobre o valor de tabela do carro.

Escreva um programa lê o ano e o preço do carro e a seguir calcula e imprime a taxa a ser paga.

## Exercício 5

Elabore um programa que calcule o que deve ser pago por um produto, considerando o preço normal de etiqueta e a escolha da condição de pagamento.

Utilize os códigos da tabela seguinte para ler qual a condição de pagamento escolhida e efetuar o cálculo adequado.

Código	Condições de pagamento
1	À vista em dinheiro ou cheque, recebe 10% de desconto
2	À vista no cartão de crédito, recebe 5% de desconto
3	Em 2 vezes, preço normal de etiqueta sem juros
4	Em 3 vezes, preço normal de etiqueta mais juros de 10%



## Exercício 6

---

Escreva um programa que lê dois números inteiros e calcula a multiplicação entre os números dados, sem o uso do operador \*, mas sim pela soma sucessiva de um deles. Exemplo:  $3 \times 4 = 3 + 3 + 3 + 3 = 4 + 4 + 4 = 12$ .

## Exercício 7

---

Escreva um programa que leia uma sequência de números inteiros, encontre e imprima o maior e o menor número. A entrada de um número negativo indica que sequência terminou.

## Exercício 8

---

Escreva um programa que receba 10 números inteiros, calcule e mostre a soma dos pares positivos.

## Exercício 9

---

Faça um programa que receba a idade, a altura e o peso de 5 pessoas, calcule e mostre:

- a) a quantidade de pessoas com idade superior a 50 anos;
- b) a média das alturas das pessoas com idade entre 10 e 20 anos;
- c) a porcentagem de pessoas com peso inferior a 40 quilos entre todas as pessoas analisadas.

## Exercício 10

A sequência de números de Fibonacci é a seguinte: os dois primeiros termos têm o valor 1 e cada termo seguinte é igual à soma dos dois anteriores.

1, 1, 2, 3, 5, 8, 13, 21, ...

Escreva um programa que solicite ao usuário o número do termo da sequência de Fibonacci e calcule o valor desse termo.

P.ex., se o número fornecido pelo usuário for 7, o programa deverá encontrar e imprimir na tela o valor '13'.

# BONS ESTUDOS!

