Introdução à Programação C

Fabio Mascarenhas - 2014.2

http://www.dcc.ufrj.br/~fabiom/introc

Repetição

- Muitos problemas em programação são resolvidos através da execução de inúmeros pequenos passos semelhantes entre si

- Sempre que temos um vetor, por exemplo, é natural querer fazer alguma coisa com cada elemento do vetor

- Algumas vezes o número de passos nem mesmo é conhecido de antemão, mas sim dado por algo que o usuário forneceu

- Toda linguagem fornece dois mecanismos básicos para repetir algo um certo número de passos: *laços* e *recursão*

O laço `while`

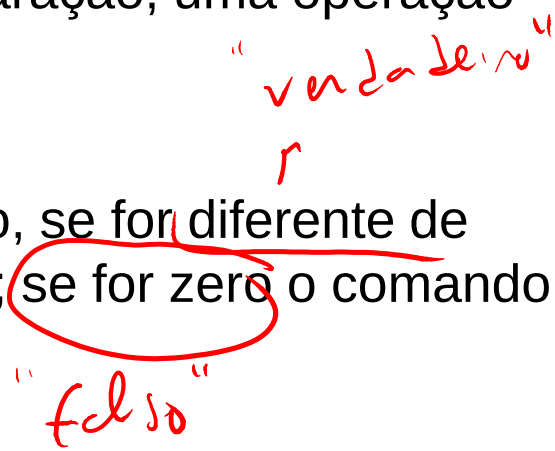
- Um *laço* é um comando que executa um bloco de comandos repetidamente até que alguma condição se torne verdade

- Em C, o laço mais simples é o comando `while`:

```
while(<condição>) {
    <comandos>
}
```

- A *condição* é qualquer expressão booleana: uma comparação, uma operação lógica, uma função predicado...

- A cada *iteração* do laço, o valor da condição é calculado, se for diferente de zero o bloco é executado, e tenta-se mais uma iteração; se for zero o comando `while` termina e a execução segue depois dele



Exemplo

- A função ganhou do jogo da forca conta o número de acertos examinando cada caractere:

```
static int ganhou() {
    int i = 0;
    int n = 0;
    while(i < NLETRAS) {
        if(acertos[i] != '_') {
            n = n + 1;
        }
        i = i + 1;
    }
    return (n == NLETRAS);
}
```

Retornando de um laço

- Um comando return dentro de um laço também encerra a função, abandonando o laço. Podemos reescrever ganhou para não precisar contar o número de acertos:

```
static int ganhou() {
    int i = 0;
    while(i < NLETRAS) {
        if(acertos[i] == '_') {
            return 0;
        }
        i = i + 1;
    }
    return 1;
}
```

- Se chegamos ao final do laço é que todas as letras já foram reveladas!

Percorrendo um vetor

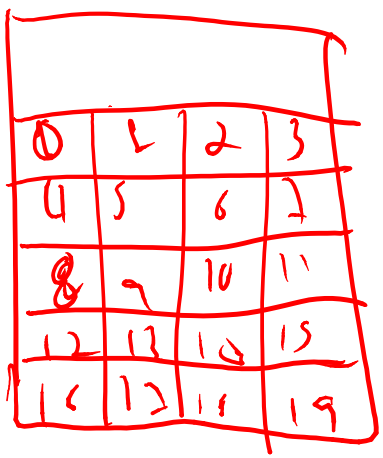
- Ambos os exemplos mostram como é um laço para percorrer os elementos de um vetor em ordem

- Usamos uma *variável de controle* com valor inicial 0, e somamos um a ela ao final de cada iteração; a condição do laço é se o valor da variável é menor que o tamanho do vetor

- Podemos usar esses laços para transformar várias tarefas repetitivas do nosso jogo de forca em laços, e deixá-lo independente do tamanho da palavra!

Exemplo - calculadora

- A interface da calculadora dos labs 2 e 3 tem vinte botões, e temos tarefas repetitivas:
  - Desenhar cada botão
  - Descobrir qual botão foi clicado, no início e no fim do clique
- Podemos usar vetores e laços para evitar que o código para cada uma dessas tarefas seja repetido vinte vezes!



Laço `for`

- O laço `for` em C é uma forma mais compacta de escrever um laço `while`:

```
for(<inicial>; <condição>; <atualização>) {
    <comandos>
}
```

- O comando `<inicial>` é executado antes da primeira iteração; a cada iteração o valor da `<condição>` é calculado, se for diferente de zero os `<comandos>` são executados, depois o comando `<atualização>` é executado, e começa uma nova iteração

- Isso corresponde ao seguinte comando `while`:

```
<inicial>;
while(<condição>) {
    <comandos>
    <atualização>;
}
```

Exemplo

- Voltando ao exemplo de ganhou, ele ficaria assim com um laço for:

```
static int ganhou() {
    int i;
    for(i = 0; i < NLETRAS; i++) {
        if(acertos[i] == '_') {
            return 0;
        }
    }
    return 1;
}
```

- A vantagem do for é reunir todos os comandos relacionados à variável de controle em um só lugar, evitando erros