Algoritmos e Programação de Computadores

Condicionais

Prof. Lucas Boaventura
lucas.boaventura@unb.br

Introdução

- Ao compilar um programa, são gerados comandos sequenciais, executados um após o outro
- Existem situações em que, para atingir o objetivo do algoritmo, é necessário tomar decisões baseadas em algum valor

Se / Então

- Em C, existem algumas formas de tratar as decisões
- A palavra-chave “if” é utilizada para isso:
  - if (verificação)
    {
  - ...
    }

Se / Então

- A condição utiliza frequentemente os operadores relacionais que estudamos na última aula
- Maior >. Maior ou igual >=
- Menor <. Menor ou igual <=
- Igual ==. Diferente !=

Se / Então

```c
#include <stdio.h>

int main()
{
    int i;

    scanf("%d", &i);
    if (i < 50)
        printf("Numero e' menor do que 50!\n");
    return 0;
}
```

Se / Então

- Esse código irá ler um número do teclado e imprimir a mensagem se o número for menor do que 50
- Quando o código tiver mais de uma linha, devemos usar chaves {} para limitar as instruções que serão executada pelo if

Introdução

```c
#include <stdio.h>

int main()
{
    int i;

    scanf("%d", &i);
    if (i < 50)
    {
        printf("Numero e' menor do que 50!\n");
        printf("Um segundo printf...\n");
    }
    return 0;
}
```

Se / Então

- A palavra-chave “else” pode ser usada para executar o trecho de código caso a expressão seja falsa
- Estrutura conhecida como “Se / Então”
- Como faríamos para criar um algoritmo para detectar se o programa digitado pelo usuário é par ou ímpar?

Se / Então

```c
#include <stdio.h>

int main()
{
    int i;

    scanf("%d", &i);

    if (i % 2 == 1)
        printf("Numero impar\n");
    else
        printf("Numero par\n");
    return 0;
}
```

Lembrete: % é o operador de resto

Operadores Relacionais

- Na aula passada vimos que para representar valores booleanos (V ou F):
- No C, são utilizados valores inteiros:
  - Valor 0: falso
  - Valor diferente de 0: verdadeiro
- Já os operadores relacionais retornam:
  - Valor 0 para falso
  - Valor 1 para verdadeiro

Operadores Relacionais

- Por isso é comum alterar o código omitindo alguns operadores relacionais
- No exemplo anterior, a linha:
  - if (i % 2 == 1)
- Pode ser substituída por:
  - if (i % 2)

Aninhados

- Pode-se combinar vários ifs em um mesmo código
- Para deixar o código claro, recomenda-se usar chaves quando aninhar vários ifs

```c
#include <stdio.h>

int main()
{
    int i;

    scanf("%d", &i);

    if (i % 2)
    {
        if (i > 10)
            printf("Numero impar maior que 10\n");
        else
            printf("Numero impar menor ou igual a 10\n");
    }
    else
    {
        printf("Numero par\n");
    }
    return 0;
}
```

Aninhados

- Se não utilizar chaves, pode-se causar confusão sobre a qual “if” pertence
- O “else” sempre pertence ao último “if”:
  - if (cond1)
  - if (cond2)
  - printf(“cond1 e 2 V\n”);
  - else //else de cond2, nao de cond1
  - printf(“cond1 V, cond2 F\n”);

Aninhados

```
if (cond1)
{
    if (cond2)
        printf("cond1 e 2 verdadeiros\n");
    else //else de cond2, nao de cond1
        printf("cond1 V, cond2 F\n");
}
else
    printf("cond1 falsa\n");
```

Else if

- Para analisar uma série de condições, não é necessário aninhar múltiplos ifs, podemos utilizar o operador “else if”

Else if

```c
#include <stdio.h>

int main()
{
    int i;

    scanf("%d", &i);

    if (i % 3 == 1)
        printf("Nao divisivel. Resto 1\n");
    else if (i % 3 == 2)
        printf("Nao divisivel. Resto 2\n");
    else
        printf("Divisivel por 3.\n");
    return 0;
}
```

Switch

- Se a operação for analisada na mesma variável, podemos uma outra construção, com as palavras-chave “switch” e “case”
- Dentro do switch, colocamos diversos “case”, um para cada valor desejado

```c
#include <stdio.h>

int main()
{
    int i;

    scanf("%d", &i);
    switch (i % 3)
    {
        case 0:
            printf("Divisivel por 3.\n");
            break;
        case 1:
            printf("Nao divisivel. Resto 1\n");
            break;
        case 2:
            printf("Nao divisivel. Resto 2\n");
            break;
    }

    return 0;
}
```

Switch

- Note que as operações “case” são seguidas de um “break”
- Se esquecer de colocar, o próximo “case” será executado!
- ESTE É O ÚNICO uso aceitável de “break” na nossa disciplina

Escopo de Variáveis

- Muito cuidado!!!
- NUNCA utilize o ; após o “if” combinado com as chaves
- Isso significa que o “if” não executa instruções caso seja verdadeiro. E o código dentro das {} sempre será executado

Escopo de Variáveis

```c
#include <stdio.h>

int main()
{
    int i;

    scanf("%d", &i);
    if (i < 50); //Este ; esta MUITO ERRADO
    {
        printf("Numero e' menor do que 50!\n");
        printf("Um segundo printf...\n");
    }
    return 0;
}
```

Escopo de Variáveis

- Para entender isso melhor, precisamos abordar um novo conceito chamado “Escopo de Variáveis”
- Quando uma variável é declarada, ela será vista dentro das chaves que ela está

Escopo de Variáveis

```c
#include <stdio.h>

int main()
{
    int i = 5;

    printf("Variavel i: %d\n", i);
    if (i > 0)
    {
        int j = 5;

        printf("Variavel j: %d\n", j);
    }
    return 0;
}
```

Escopo de Variáveis

- Nas versões mais modernas do C, é possível declarar em qualquer momento a variável
- A partir da declaração, até o final do escopo, a variável é visível dentro do seu código

Escopo de Variáveis

```c
#include <stdio.h>

int main()
{
    int i = 5;

    printf("Variavel i: %d\n", i);
    if (i > 0)
    {
        int j = 5;

        printf("Variavel j: %d\n", j);
    }
    return 0;
}
```

Escopo de i

Escopo de j

Escopo de Variáveis

- Quando o escopo de uma variável termina (ou seja, depois do ‘}’) a variável não pode mais ser usada
- Isto causará um erro de compilação, mesmo erro que causaria se usasse uma variável sem declarar antes!

Escopo de Variáveis

```c
#include <stdio.h>

int main()
{
    int i = 5;

    printf("Variavel i: %d\n", i);
    if (i > 0)
    {
        int j = 5;
    }
    printf("Variavel j: %d\n", j); //NAO compila
    return 0;
}
```

error: ‘j’ undeclared (first use in this function)

Escopo de Variáveis

- É possível (mas não é recomendado) utilizar as chaves dentro do código para criar um novo escopo!
- Essa construção é aceita pela linguagem C

Escopo de Variáveis

```c
#include <stdio.h>

int main()
{
    int i = 5;

    {
        int j = 3;
        printf("Variavel j: %d\n", j);
    }

    printf("Variavel i: %d\n", i);
    return 0;
}
```

Saída:

Variavel j: 3

Variavel i: 5

Escopo de Variáveis

- Quando utilizamos um ‘;’ após o if, o que nós estamos fazendo?
- Um if VAZIO, que não executa nenhuma instrução caso seja verdadeiro
- Abrimos um escopo que sempre será executado

Códigos Iguais

```c
#include <stdio.h>

int main()
{
    int i = 5;

    if (i < 0);
    {
        printf("NEGATIVO!\n");
    }
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    int i = 5;

    if (i < 0)
        ;

    {
        printf("NEGATIVO!\n");
    }
    return 0;
}
```

Códigos Iguais

```c
#include <stdio.h>

int main()
{
    int i = 5;

    if (i < 0)
    {

    }

    {
        printf("NEGATIVO!\n");
    }
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    int i = 5;

    if (i < 0)
    {

    }

    printf("NEGATIVO!\n");
    return 0;
}
```

Escopo de Variáveis

- O compilador pode te ajudar a detectar esse tipo de construção inválida
- Por isso sempre compilamos com a flag “-Wall”
- Ele irá produzir um aviso. O código será compilado, mas você foi avisado pelo compilador que uma construção estranha estava no código!

Escopo de Variáveis

```c
#include <stdio.h>

int main()
{
    int i = 5;

    if (i < 0);
    {
        printf("NEGATIVO!\n");
    }
    return 0;
}
```

```
[user@station code]$ gcc -Wall -o saida e8.c
e8.c: In function ‘main’:
e8.c:7:5: warning: this ‘if’ clause does not guard...
    7 |     if (i < 0);
      |     ^~
```

Dúvidas?

- lucas.boaventura@unb.br