Universidade Federal do Espirito Santo
Centro Tecnologico

{introcomp}

Working 03 : Conceitos Básicos II

Objetivos:

- Dominar a construção de estruturas de seleção em C;
- Aperfeiçoarse na resolução de problemas, da primeira ideia ao programa pronto;

Prazo de Envio: sexta, 26/08, 23:55.

PET
Engenharia de Computação

1 INTRODUÇÃO

Até agora você programou utilizando variáveis, identificadores, expressões aritméticas, relacionais e lógicas, comandos de entrada e saída de dados. Mas estes são comandos básicos necessários para dominar C. O que é determinante para a programação de computadores e a geração de programas, sistemas complexos e inteligentes são os controles de fluxo! Na aula passada, aprendemos sobre um destes: as estruturas condicionais. Aprender a utilizálas da melhor maneira e saber o melhor formato de implementá-las de maneira eficiente pode ser o primeiro passo para projetar tecnologias rápidas e sofisticadas.

2 A ESTRUTURA DE SELEÇÃO IF

Em C, a estrutura de seleção simples obedece à seguinte regra de sintaxe:

```
1 if (<expressao logica>)
2 {
3     <sequencia de comandos>
4 }
```

Se, a expressão lógica for verdadeira, o computador executará a sequência de comandos de dentro do if. Porém, há também a estrutura de seleção dupla que utilizamos quando precisamos implementar o "caso contrário", ou seja, os comandos dos quais o computador irá executar no caso da expressão lógica ser falsa. Ela obedece à seguinte sintaxe:

```
1 if (<expressao logica>)
2 {
3     <sequencia de comandos>
4 }
5 else
6 {
7     <sequencia de comandos>
8 }
```

Há uma questão importantíssima para a implementação do 'caso contrário', que é o entendimento de qual seria o caso contrário presente numa expressão lógica. Por exemplo, qual seria o caso contrário da expressão a>3 a<5 ? Vamos pensar, se a=4, a expressão é verdadeira. Se a <= 3 ou a >= 5, a expressão é falsa. Portanto, o caso para que a expressão seja verdadeira é de que 'a' esteja entre os números 3 e 5. Já o caso contrário disso, é que 'a' esteja acima ou igual a 5 ou (||) abaixo ou igual a 3. Por isso, sempre verifique se a estrutura condicional que deseja "economizar" é válida para o caso contrário. Caso contrário (rsrs), não a economize.

3 DETALHES PARA PRODUTIVIDADE

Há alguns detalhes em C que aumentam a produtividade do programador. Por exemplo, se só há um comando num if ou else, basta fazer o seguinte:

```
if (<expressao logica>)
    <comando>
else
    <comando>
```

Mas atente-se ao fato de que um comando termina com um ';'. Outro detalhe que aumenta a produtividade é quando só há uma estrutura de if dentro do caso contrário. Exemplo:

```
if (<expressao logica>)
{
    <sequencia de comandos>
}
else
{
    if (<expressao logica>)
    {
        <sequencia de comandos>
    }
}
```

Observação: Apesar de parecer mais simples utilizar "if/else" de uma linha só caso se tenha somente um comando em seu escopo, recomendamos fortemente que para fim de aprendizado se use ‘‘ e ’’. Isso evita erros bobos e fortifica ainda mais o entendimento de escopo em estruturas.

4 OUTRAS ESTRUTURAS DE SELEÇÃO

Além do if, existe o switch. Sua sintaxe é a seguinte:

```
switch (expressao)
{
        case <valor1>:
                    <sequencia de comandos>
                    break;
        case <valor2>:
                    <sequencia de comandos>
                    break;
        .
        .
        .

        case <valorN>
                    <sequencia de comandos>
                    break;
        default:
                    <sequencia de comandos>
}
```

Agora vamos explicar como ela funciona. Primeiramente, você deve especificar uma expressão no cabeçalho, entre parênteses. Em seguida, o resultado dessa expressão é comparada com os valores especificados no case. Quando o valor da expressão for o mesmo que o valor determinado no case, o programa entrará nesse case e executará a sequência de comandos implementadas nele. Em seguida, é executado o comando break, pulando todo o resto da estrutura switch e executando o resto do programa.
O comando break possui esta função, quando ele é executado, o programa em execução sai da estrutura (seleção ou repetição) em que ele se encontra e continua a execução normal do programa.
Com isso, você especifica os case e, assim, o valor da expressão será comparada com o valor dos case até que um ou nenhum coincidir. Quando nenhum coincidir, o programa executará os comandos determinados no default. Porém, não é obrigatório o uso da instrução default.
Um exemplo para melhor fixação:

```
int main (){

    int candidato;
    printf ("Digite o n mero do seu candidato e tecle ENTER: ");
    scanf ("%d",&candidato);

    switch (candidato)
    {
            case 13:
                        printf ("Voce votou na Iriny Lopes!\n");
                        break;
            case 23:
                        printf ("Voce votou no Rezende!\n");
                        break;
            case 43:
                        printf ("Voce votou no Luiz Paulo!\n");
                        break;
            default:
                        printf ("Voce votou nulo!\n");
    }

    return 0;
}
```

Como pode-se observar, você digita o número do candidato e este número é passado para o switch. Com isso, caso o número seja 13, será executado as instruções do caso 13 e sairá do switch devido ao break. Observe que, neste caso, o valor default foi utilizado. Atentese também que não foi utilizado o break no valor default, já que ele é o último valor do comando switch.

Praticando

1. **(FÁCIL)**Faça um programa leia dois números inteiros do teclado e diga qual o maior deles.

2. **(FÁCIL)**Faça um programa que pergunte a idade de uma pessoa e diga se ela pode ou não doar sangue. Para doar sangue é necessário ter entre 18 e 67 anos.

3. **(MÉDIO)**Faça um programa que leia do teclado três números reais e verifique se o primeiro está contido no intervalo fechado(os limites estão contidos no intervalo) formado pelo segundo e terceiro.
   Atenção: o segundo número pode ser maior que o terceiro número.
   Exemplo: 5 2 10
   O número 5 está contido no intervalo, ou seja, ele é maior que 2 e menor que 10.

4. **(DIFÍCIL)**Faça um programa que tem como função verificar se o usuário digitou uma letra do teclado. Caso tenha digitado informe "Sim, é uma letra.", caso contrário "Não é uma letra." O programa deve considerar letras maiúsculas e minúsculas. OBS: Você poderá usar somente 1(UM) operador lógico OU(||).

5. **(DIFÍCIL)**Faça um programa que receba como entrada os coeficientes a,b,c (lidos do teclado) de uma equação do segundo grau do tipo ax + bx + c, e informe:

   - "Nenhuma raiz real", caso só possua raízes complexas (Delta menor que zero).
   - "Uma única raiz real", caso a equação só possua uma raíz real.
   - "Duas raízes reais", caso possua duas raízes reais.

Desafios

1. **(Olímpiada Brasileira de Informática – Nível 1, Fase 1)**

   Os computadores foram inventados para realizar cálculos muito rapidamente, e atendem a esse requisito de maneira extraordinária. Porém, nem toda conta pode ser feita num computador, pois ele não consegue representar todos os números dentro de sua memória. Em um computador pessoal atual, por exemplo, o maior inteiro que é possível representar em sua memória é 4.294.967.295. Caso alguma conta executada pelo computador dê um resultado acima desse número, ocorrerá o que chamamos de overflow, que é quando o computador faz uma conta e o resultado não pode ser representado, por ser maior do que o valor máximo permitido (em inglês overflow significa trasbordar). Por exemplo, se um computador só pode representar números menores do que 1023 e mandamos ele executar a conta 1022 + 5, vai ocorrer overflow.

Tarefa
Dados o maior número que um computador consegue representar e uma expressão de soma ou multiplicação entre dois inteiros, determine se ocorrerá overflow.
Entrada
A entrada contém um único conjunto de testes, que deve ser lido do dispositivo de entrada padrão (normalmente o teclado).
A primeira linha da entrada contém um inteiro N (1 N 500.000) representando o maior número que o computador consegue representar. A segunda linha contém um inteiro P (0 P 1000), seguido de um espaço em branco, seguido de um caractere C (que pode ser '+' ou '*', representando os operadores de adição e multiplicação, respectivamente), seguido de um espaço em branco, seguido de um outro inteiro Q (0 Q 1000). Essa linha representa a expressão P + Q, se o caractere C for '+', ou P Q, se o caractere C for '*'.
Saída
Seu programa deve imprimir, na saída padrão, a palavra 'OVERFLOW' se o resultado da expressão causar um overflow, ou a palavra 'OK' caso contrário. Ambas as palavras devem ser escritas com letras maiúsculas.

Exemplos:

Entrada	**Saida Correspondente**
10 5+5	OK
44 23*2	OVERFLOW
323500 42*35	OK

NO PRÓXIMO ENCONTRO...

No próximo encontro vamos continuar nossos estudos em estruturas para controle de fluxo. Veremos uma outra estrutura extremamente importante na construção de programas: a estrutura de repetição!