Programação de Computadores

INSTRUÇÃO DE DESVIO SWITCH

Judson Santiago

Introdução

- Uma tarefa muito comum em um programa é a verificação da entrada do usuário
  - Para isso é preciso identificar o que foi digitado:
    - Letras
    - Números
    - Pontuação
    - Etc.
- Esta verificação vai precisar de instruções de desvio
  - Para corrigir uma entrada errada

A Biblioteca cctype

- C++ herdou de C um pacote de funções que facilita a tarefa de saber se um caractere é uma letra, dígito ou pontuação
  - isalpha(ch)
    retorna *verdadeiro* se ch é uma letra e *falso* caso contrário
  - ispunct(ch)
    retorna *verdadeiro* apenas se ch é um caractere de pontuação
- Essas funções retornam um valor inteiro,
  mas a conversão para booleano é automática

A Biblioteca cctype

Função	Valor de retorno é verdadeiro se o argumento é
isalnum( )	alfanumérico (letra ou dígito)
isalpha( )	uma letra do alfabeto
isblank( )	um espaço ou tabulação
isspace( )	um espaço, tabulação ou nova linha
iscntrl( )	um caractere de controle
isdigit( )	um dígito decimal (`0-9`)
ispunct( )	um caractere de pontuação
islower( )	uma letra minúscula
isupper( )	uma letra maiúscula
tolower( )	Se o argumento é maiúsculo, retorna o caractere minúsculo correspondente
toupper( )	Se o argumento é minúsculo, retorna o caractere maiúsculo correspondente

A Biblioteca cctype

- Usar as funções da biblioteca cctype é mais conveniente e mais seguro do que usar os operadores lógicos

```
if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' &&  ch <= 'Z'))
```

```
if (isalpha(ch))
```

- O uso da cctype torna o código mais geral porque os operadores lógicos assumem que as letras têm códigos sequenciais (ASCII)

A Biblioteca cctype

```
#include <iostream>
#include <cctype>
using namespace std;
int main()
{
     cout << "Entre com o texto para análise (@ para sair):\n";
     int brancos = 0, digitos = 0, chars = 0, pont = 0, outros = 0;
     char ch;

     cin.get(ch);                          // lê o primeiro caractere
     while (ch != '@')                     // testa o caractere sentinela
     {
         if (isalpha(ch)) chars++;         // é uma letra do alfabeto?
         else if (isspace(ch)) brancos++;  // é um caractere de espaço?
         else if (isdigit(ch)) digitos++;  // é um dígito?
         else if (ispunct(ch)) pont++;     // é uma pontuação?
         else outros++;
         cin.get(ch);                      // lê o próximo caractere
     }
     cout << chars << " letras, " << digitos << " digitos " << brancos << " espaços, "
          << pont << " pontuações e " << outros << " outros.\n";
}
```

Classifica.cpp

A Biblioteca cctype

- Saída do Programa:

```
Entre com o texto para análise (@ para sair):
João "ex-aluno" Zinho, renomado programador,
escreveu seu primeiro programa em 2012.@
64 letras, 4 dígitos, 10 espaços, 6 pontuações e 0 outros.
```

- A contagem de espaços inclui os caracteres de nova linha
- Hífens, aspas, vírgulas e pontos contam como caracteres de pontuação

O Operador ?:

- O operador condicional ternário **?:** pode substituir um if else

```
expressão1 ? expressão2 : expressão3
```

  - Se a expressão 1 é verdadeira, toda a expressão condicional tem o valor da expressão 2, caso contrário ela tem o valor da expressão 3

```
5 > 3 ? 10 : 12  // como 5 > 3, a expressão tem valor 10
3 == 9 ? 25 : 18 // como 3 != 9, a expressão tem valor 18
```

O Operador ?:

```cpp
#include <iostream>
using namespace std;

int main()
{
	cout << "Entre com dois inteiros: ";
	int a, b;
	cin >> a >> b;

	cout << "O maior entre " << a << " e " << b;
	cout << " é " << a > b ? a : b << endl;

	return 0;
}
```

Maior.cpp

O Operador ?:

- Saída do Programa:

```
Entre com dois inteiros: 25 28
O maior entre 25 e 28 é 28.
```

- A principal parte do programa é a instrução:

```
cout << a > b ? a : b;

// um if else produz o mesmo resultado
if (a > b)
  cout << a;
else
  cout << b;
```

O Operador ?:

- O uso do operador ?: deve ser feito com cautela
  - Pode gerar expressões completamente ilegíveis

```
const char x[2][20] = {"Pedro ", " a seu serviço"};
const char * y = "Orlando ";
for (int i = 0; i < 3; i++)
    cout << ( (i < 2) ? !i ? x[i] : y : x[1] );
```

  - Ele é mais adequado para expressões simples

```
char maiusculo = char(isupper(ch) ? ch : toupper(ch));
```

  - Para expressões complexas use o if else

A Instrução switch

- Em muitas aplicações se faz necessário escolher uma opção entre uma lista de opções

```
Como você qualifica os preços dessa loja?

1. Barato
2. Moderado
3. Caro
4. Abusivo

Digite o número correspondente: [ ]
```

A Instrução switch

- O problema da seleção pode ser resolvido usando uma sequência de if else's:

```
cout << "Digite o número correspondente: [ ]\b\b";
int escolha;
cin >> escolha;

if (escolha == 1)
   cout << "Os preços são baratos!";
else if (escolha == 2)
   cout << "Os preços são moderados!";
else if (escolha == 3)
   cout << "Os preços são caros!";
else if (escolha == 4)
   cout << "Os preços são abusivos!";
else
   cout << "Opção inválida!";
```

A Instrução switch

- A instrução switch permite selecionar mais facilmente uma opção em uma lista extensa

```
switch (escolha)
{
   case 1:  cout << "Os preços são baratos!";
            break;
   case 2:  cout << "Os preços são moderados!";
            break;
   case 3:  cout << "Os preços são caros!";
            break;
   case 4:  cout << "Os preços são abusivos!";
            break;
   default: cout << "Opção inválida!";
}
```

A Instrução switch

- A forma geral de uma instrução switch:

A expressão deve se reduzir a um valor inteiro

```
switch(expressão-inteira)
{
   case val1: instruções
   case val2: instruções

   ...

   default: instruções
}
```

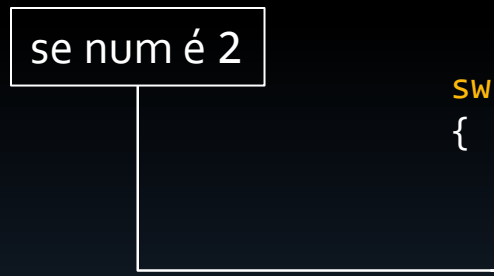
Os rótulos val1, val2, etc. devem ser constantes inteiras.

Tipicamente são valores int ou char ou constantes de uma enumeração

Se nenhum caso for válido, o caso default é selecionado

A Instrução switch

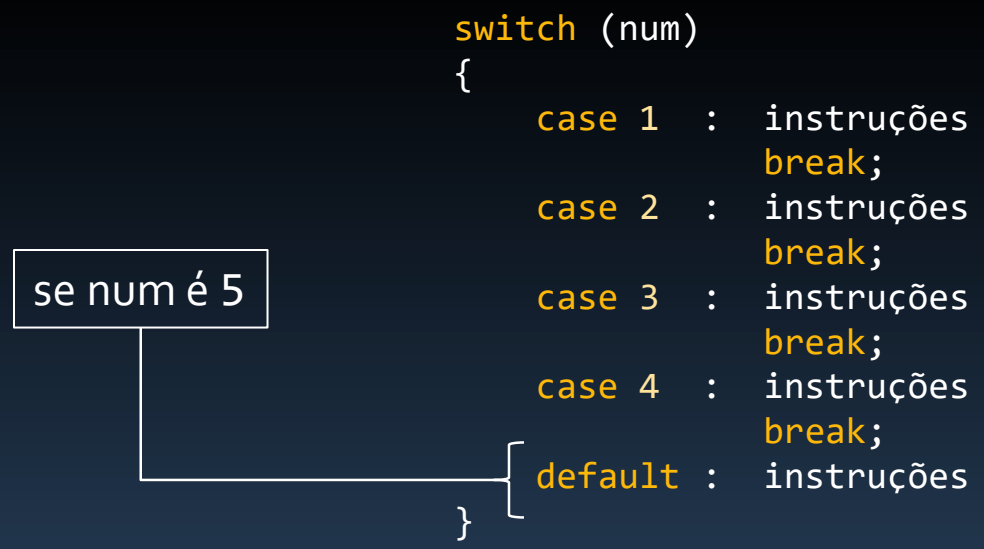
- Se a expressão-inteira é igual a um rótulo, a execução do programa pula para o rótulo selecionado




```
switch (num)
{
    case 1  :  instruções
               break;
    case 2  :  instruções
               break;
    case 3  :  instruções
               break;
    case 4  :  instruções
               break;
    default :  instruções
}
```

A Instrução switch

- Se a expressão-inteira não é igual a nenhum dos rótulos, a execução do programa pula para o rótulo default (opcional)



A Instrução switch

```
#include <iostream>
using namespace std;
void menu();
void relatorio();
void bajular();
int main()
{
     menu();

     int escolha;
     cin >> escolha;

     while (escolha != 5)
     {
        menu();
        cin >> escolha;
     }
}
```

```
switch (escolha)
{
    case 1  :  cout << "\a\n\n";
               break;
    case 2  :  relatorio();
               break;
    case 3  :  cout << "Eu estava doente.\n\n";
               break;
    case 4  :  bajular();
               break;
    default :  cout << "Opção inválida.\n\n";
}
```

Menu.cpp

Continua →

A Instrução switch

```
void menu()
{
   cout << "1) Alarme         2) Relatório\n"
           "3) Desculpa       4) Bajulação\n"
           "5) Sair\n"
           "Por favor, entre com uma opção: ";
}
void relatorio()
{
   cout << "Tem sido uma excelente semana para negócios.\n"
           "As vendas subiram 120%. Os gastos caíram 35%.\n\n";

}
void bajular()
{
   cout << "Seus empregados acham você o melhor chefe da indústria.\n"
           "Os seus sócios o consideram o melhor empresário do mercado.\n\n";
}
```

Menu.cpp

A Instrução switch

- Saída do Programa:

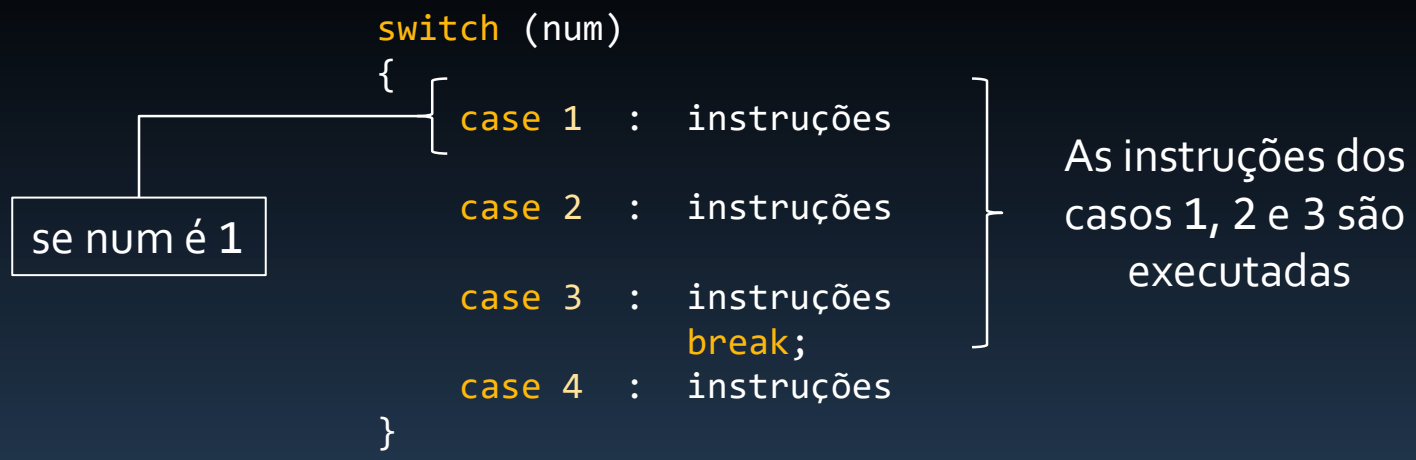
```
1) Alarme        2) Relatório
3) Desculpa      4) Bajulação
5) Sair
Por favor, entre com uma opção: 4
Seus empregados acham você o melhor chefe da indústria.
Os seus sócios o consideram o melhor empresário do mercado.

1) Alarme        2) Relatório
3) Desculpa      4) Bajulação
5) Sair
Por favor, entre com uma opção : 6
Opção inválida.

1) Alarme        2) Relatório
3) Desculpa      4) Babação
5) Sair
Por favor, entre com uma opção : 5
```

A Instrução switch

- Para fazer a execução parar ao final de um caso é preciso usar a instrução break
  - A instrução break faz com que a execução pule para a primeira instrução fora do switch



A Instrução switch

- C++ exige a presença do break para encerrar a execução de um caso porque omitir um break pode ser útil

```
char escolha;
cin >> escolha;

switch (escolha)
{
    case 'a' :
    case 'A' :  cout << "Você escolheu a letra A\n";
                break;
    case 'b' :
    case 'B' :  cout << "Você escolheu a letra B\n";
                break;

    default  :  cout << "Letra inválida.\n";
}
```

Usando Enumerações

```
#include <iostream>
using namespace std;
enum {vermelho, laranja, amarelo, verde};
int main()
{
     cout << "Entre com o código da cor: ";
     int cor; cin >> cor;
     while (cor >= vermelho && cor <= verde)
     {
         switch (cor)
         {
            case vermelho: cout << "Seu batom era vermelho.\n"; break;
            case laranja:  cout << "Sua roupa era laranja.\n"; break;
            case amarelo:  cout << "Seus sapatos eram amarelos.\n"; break;
            case verde :   cout << "Seus olhos eram verdes.\n"; break;
         }
         cout << "Entre com o código da cor: ";
         cin >> cor;
     }
     cout << "Tchau!\n";
}
```

Cores.cpp

Usando Enumerações

- Saída do Programa:

```
Entre com o código da cor: 3
Seus olhos eram verdes.
Entre com o código da cor: 0
Seu batom era vermelho.
Entre com o código da cor: 8
Tchau!
```

- O cin não reconhece tipos criados pelo programador por isso o programa precisa ler a cor como um valor inteiro (ou sobrescrever operator>>)

switch *versus* if else

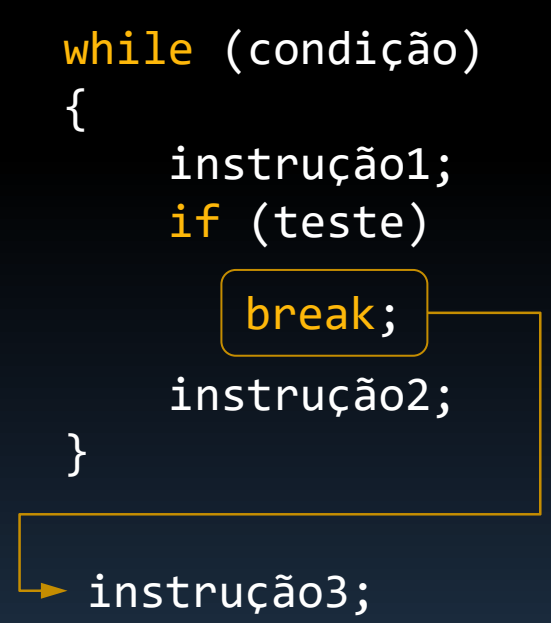
- Ambos permitem fazer uma seleção a partir de uma lista de alternativas, porém:
  - Faixas de valores
    - O if else trata faixas de números inteiros ou pontos flutuantes
    - Os casos do switch não podem ser valores ponto flutuantes
  - Lista extensa
    - O switch produz código menor e de execução mais rápida quando a lista de alternativas é extensa

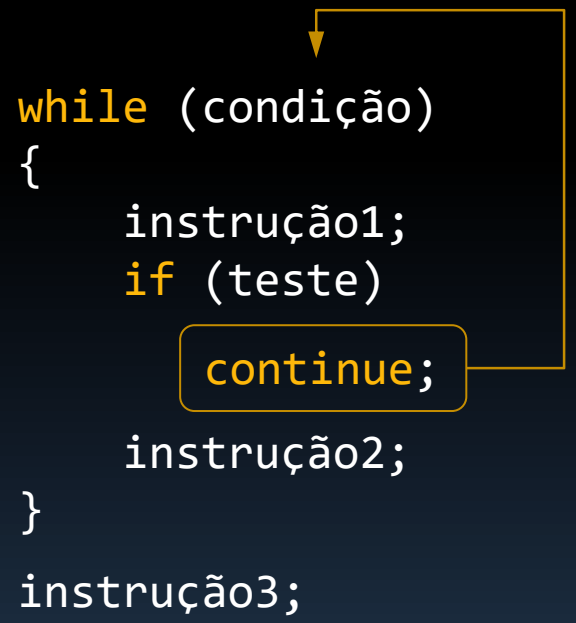
break e continue

- As instruções break e continue permitem ao programa pular partes de código
  - break:
    - Pode ser usado no switch ou em laços de repetição
    - Faz com que o programa saia do switch ou do laço
  - continue:
    - É usado em laços: faz o programa pular o resto do corpo e iniciar um novo ciclo do laço

break e continue

- A estrutura da instrução break e continue:





break e continue

```
#include <iostream>
using namespace std;
int main()
{
     char linha[80];
     int espacos = 0;
     cout << "Entre com uma linha de texto:\n";
     cin.getline(linha, 80);
     cout << "Linha completa: " << linha << endl;
     cout << "Linha até o primeiro ponto: ";
     for (int i = 0; linha[i] != '\0'; ++i)
     {
        cout << linha[i];      // mostra caractere
        if (linha[i] == '.')
            break;             // encerra laço se for um ponto
        if (linha[i] != ' ')
            continue;          // pula o resto do laço se não for um espaço
        espacos++;
     }
     cout << "\n" << espacos << " espaços\n";
}
```

Lanche.cpp

break e continue

- Saída do Programa:

```
Entre com uma linha de texto:
Vamos lanchar agora. Você paga!
Linha Completa: Vamos lanchar agora. Você paga!
Linha até o primeiro ponto: Vamos lanchar agora.
2 espaços
```

- Este programa não precisava usar break ou continue

```
for (int i = 0; (linha[i] != '\0') && (linha[i] != '.'); ++i)
{
        cout << linha[i];
        if (linha[i] == ' ')
            espacos++;
}
```

Resumo

- É possível fazer desvios com as instruções if e switch
  - O switch é ideal para longas listas de opções
  - O if para as demais situações
- O operador ?: pode substituir o if else
  - Ideal para expressões pequenas e simples
- As instruções break e continue fazem desvios dentro de laços
  - Quebram a lógica da programação estruturada
  - É sempre possível reescrever o código para eliminá-las