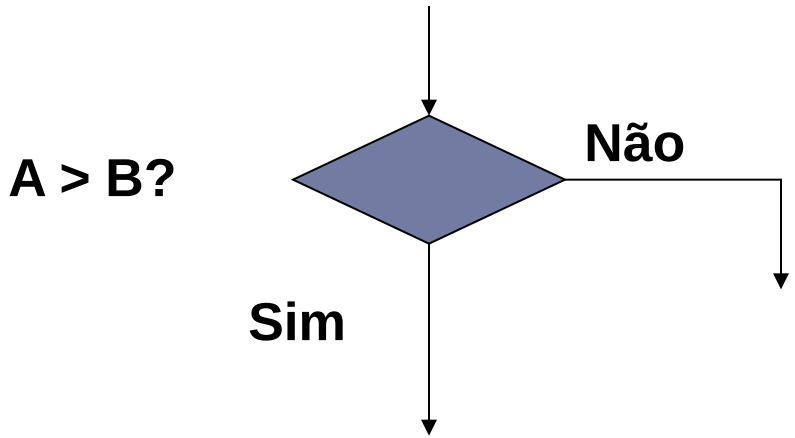
Comandos de Controle Condicional

Prof. Bruno Travençolo
Baseado em slides do Prof. André Backes

Fluxogramas

- Condição ou Decisão
  - Representado por losangos
  - Normalmente contém uma pergunta do tipo Sim/Não ou um teste de Verdadeiro/Falso.
  - Mudança no fluxo



Comando if

- Em linguagem C, o comando if é utilizado quando for necessário escolher entre dois caminhos, ou quando se deseja executar um comando sujeito ao resultado de um teste.

Comando if

- A forma geral de um comando **if** é:

  if (expressão)
  instrução

  (em inglês)
  if (expression)
  statement

- A expressão, na condição, será avaliada:
  - Se ela for zero (falsa), a declaração não será executada;
  - Se a condição for diferente de zero (verdadeira) a declaração será executada.

Exemplo if

```c
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    }
    return 0;
}
```

Verificando a sintaxe

if (expression)
statement

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    }
    return 0;
}
```

Verificando a sintaxe

if (expression)
statement

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    }
    return 0;
}
```

Verificando a sintaxe

if (expression)
statement

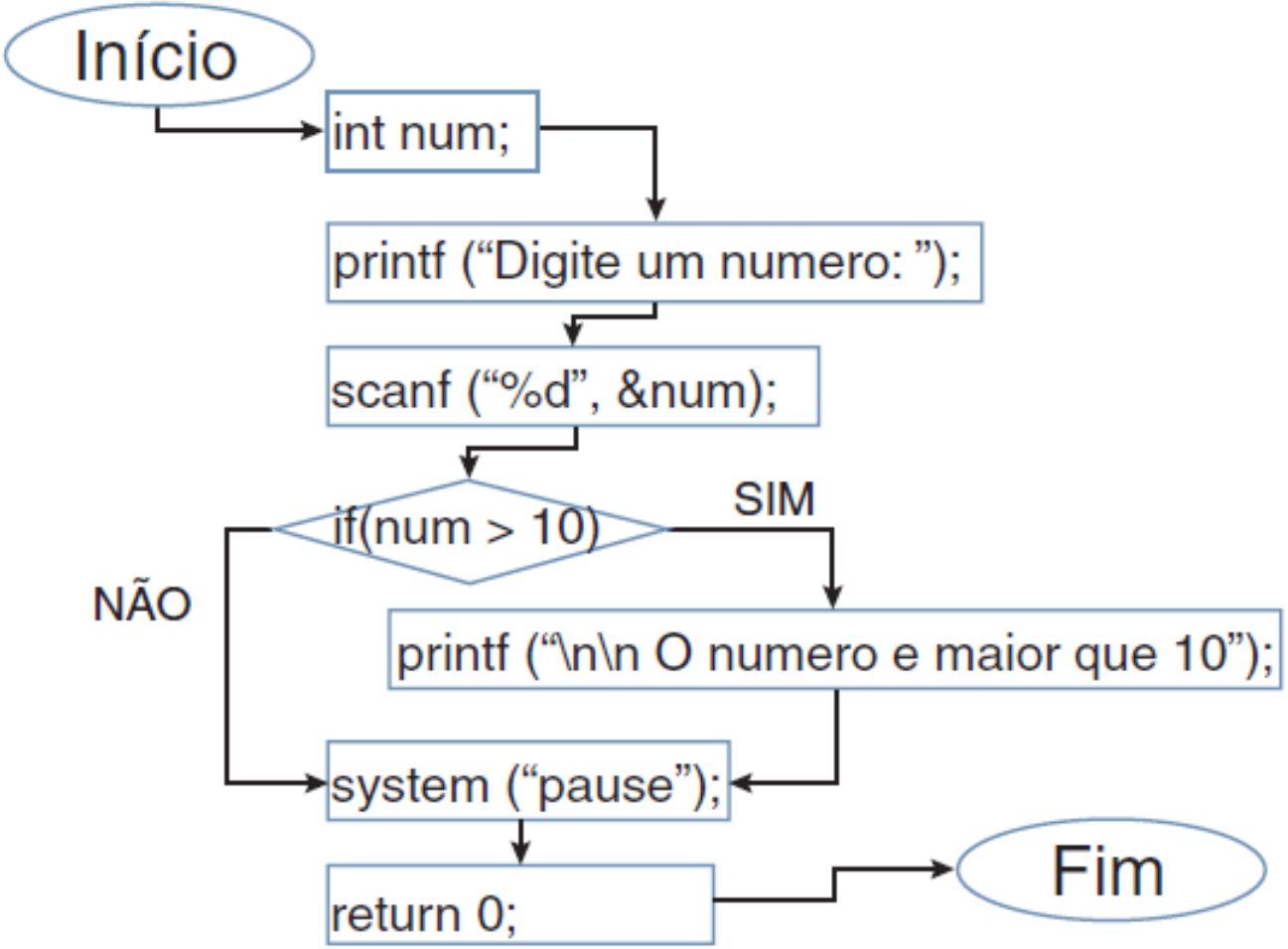
```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    }
    return 0;
}
```

Verificando a sintaxe

if (expression)
statement

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    }
    return 0;
}
```

Exemplo if


Início
int num;
printf (“Digite um numero: ”);
scanf (“%d”, &num);
if(num > 10)
SIM
NÃO
printf (“\n\n O numero e maior que 10”);
system (“pause”);
return 0;
Fim

Condição do if

if (expression)
    statement

- A *expression* (expressão) pode ser definida usando operadores matemáticos, lógicos e relacionais
  - +,-, *, /, %
  - &&, ||
  - >, <, >=, <=, ==, !=
- Ex:
  - (x > 10 && y <= x-1)

Condição do if

- Tabela verdade

a	b	!a	!b	a && b	a || b
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

Observação sobre a sintaxe

if (expression)
statement

- O que um *statement* (instrução)?
  - É uma única instrução da linguagem
    - Um *statement* termina com um sinal de ponto e vírgula **;**

```
prinft("Hello World!");
```

  - Ou é um conjunto de instruções delimitada por chaves, o que é chamado de Bloco de Instruções
    - Block Delimiter: **{ }**
    - Dentro de um bloco podemos colocar mais de uma instrução

```
{
  prinft("Hello World!");
  prinft("Hello World Again!");
}
```

Comando if

- Pode-se usar chaves **{ }** para delimitar o bloco de instruções que pertence ao if

```
if (num > 10) {
  printf ("\n\n O numero eh maior que 10");
}
```

- As chaves **devem** ser usadas no caso de mais de uma instrução:

```
if (nota >= 60) {
    printf ("A nota é maior ou igual a 60 \n") ;
    printf (“O aluno está aprovado!") ;
}
```

- As chaves podem ser ignoradas se a instrução for única.

```
if (num > 10)
  printf ("\n\n O numero e maior que 10") ;
```

Exercício

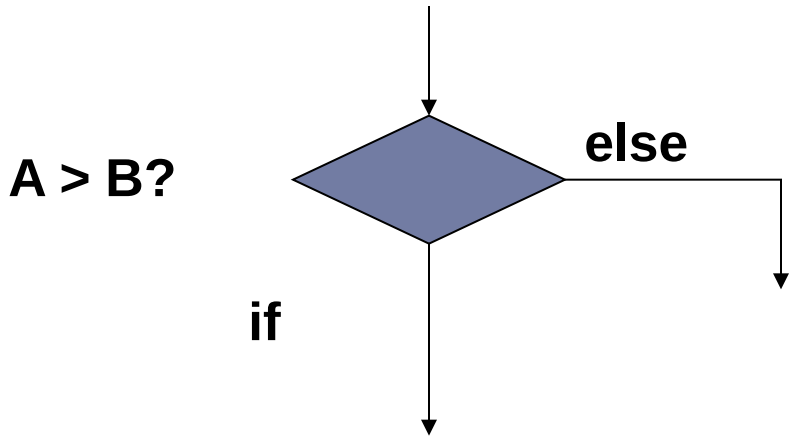
- Dada o valor da nota de um aluno, monte a expressão if que verifica se ele precisará fazer a sub. O aluno deverá fazer sub se sua nota for maior ou igual a 30 e menor do que 60.

Exercício

```c
#include <stdio.h>
#include <stdlib.h>

int main (){
  int num;
  printf ("Digite a nota: ");
  scanf ("%d",&num);

  if ((num >= 30) && (num < 60)){
    printf ("O aluno deve fazer a prova sub \n");
  }

  return 0;
}
```

Comando else

- O comando else pode ser entendido como sendo um complemento do comando if.
  - Se o **if** diz o que fazer quando a condição é verdadeiro, o **else** tratá da condição falsa.



Comando else

- O comando if-else tem a seguinte forma geral:

```
if (expressão)
    instrução1
else
    instrução2
```

(em inglês)

```
if (expression)
    statement1
else
    statement2
```

Comando else

- A expressão da condição será avaliada:
  - Se ela for diferente de zero (verdadeiro), a instrução1 será executada.
  - Se for zero (falso) a instrução2 será executada.
- Note que quando usamos a estrutura if-else, uma das duas declarações será executada.
- Não há obrigatoriedade em usar o else

Exemplo if-else

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    } else {
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

Analisando a sintaxe

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    } else {
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

```
if (expression)
        statement1
else
        statement2
```

Analisando a sintaxe

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    } else {
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

```
if (expression)
        statement1
else
        statement2
```

Analisando a sintaxe

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    } else {
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

```
if (expression)
        statement1
else
        statement2
```

Analisando a sintaxe

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    } else {
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

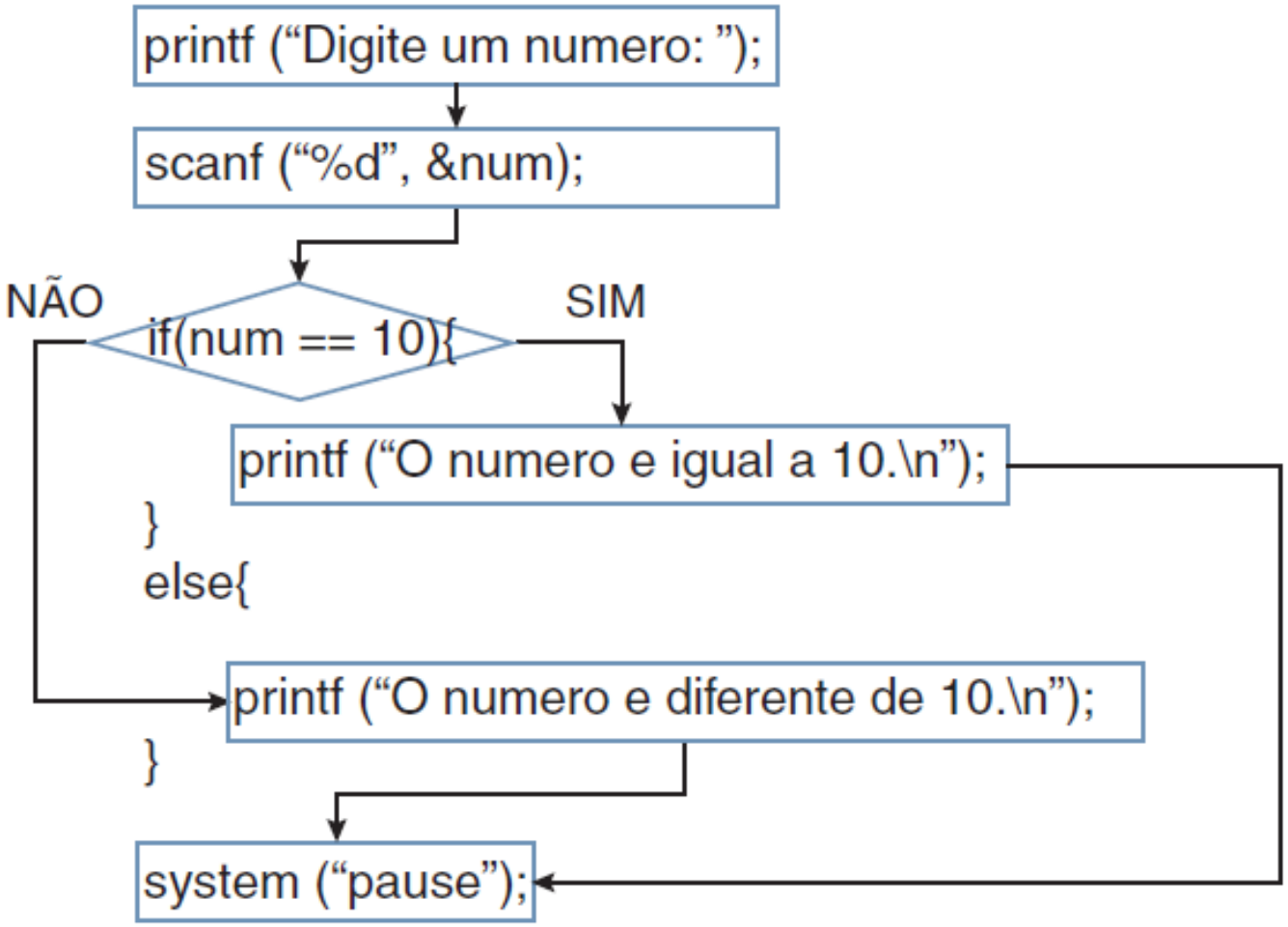
```
if (expression)
        statement1
else
        statement2
```

Analisando a sintaxe

```
if (expression)
        statement1
else
        statement2
```

```
int main()
{
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if (num==10){
        printf("O numero eh igual a 10.\n");
    } else {
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

Exemplo if-else



Comando if-else

- Como no caso do comando if, as chaves podem ser ignoradas se a instrução contida no **else** for única.

```
if (num==10){
   printf("O numero eh igual a 10.\n");
} else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

- ou

```
if (num==10){
    printf("O numero eh igual a 10.\n");
} else { // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

Comando if-else

- O statement do if é independe do statement do else.

```
if (num==10) // if sem usar chaves
   printf("O numero eh igual a 10.\n");
else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");

-----------------------------------------------

if (num==10) // if sem usar chaves
     printf("O numero eh igual a 10.\n");
else { // else com chaves
     printf("O numero eh diferente de 10.\n");
}

-----------------------------------------------
if (num==10){// if com chaves
    printf("O numero eh igual a 10.\n");
} else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");

-----------------------------------------------
if (num==10){// if com chaves
     printf("O numero eh igual a 10.\n");
 } else { // else com chaves
     printf("O numero eh diferente de 10.\n");
}
```

Em todos os casos existe a possibilidade de não usar chaves pois o statment do if possui somente um comando e o statement do else também.

Comando if-else

A sequência de comandos de **if** é independente da sequência de comandos de **else**. Cada comando tem o seu próprio conjunto de chaves ({ }).

Uso das chaves no comando if-else

	Certo	Errado

01
02
03
04
05
06
	`if(condicao){`
	`  sequencia de comandos;`
	`}`
	`else{`
	`  sequencia de comandos;`
	`}`
		`if(condicao){`
		`  sequencia de comandos;`
		`else`
		`  sequencia de comandos;`
		`}`

Aninhamento de if

- O **if** aninhado é simplesmente um **if** dentro da declaração de um outro **if** externo.
  - A estrutura if-else-if é apenas uma extensão da estrutura if-else.
- O único cuidado que devemos ter é o de saber exatamente a qual **if** um determinado **else** está ligado.

Aninhamento de if

```
if(condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

Aninhamento de if

- O programa começa a testar as condições começando pela 1 e continua a testar até que ele ache uma expressão cujo resultado dê diferente de zero (verdadeiro). Neste caso ele
  - executa a seqüência de comandos correspondente.
  - Só uma seqüência de comandos será executada, ou seja, só será executada a seqüência de comandos equivalente à primeira condição que der diferente de zero.
  - A última seqüência de comandos (default) é a que será executada no caso de todas as condições darem zero (falso) e é opcional.

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

if (expression)
statement1
else
statement2

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

if (expression)
statement1
else
statement2

Observe que no *statement2* há somente um comando – que é um outro comando **if**. Este, por sua vez, terá vários outros comandos internos. Mas para o *statement2* do **if** atual, ele só enxerga 1 comando (if), pois isso não é preciso abrir chaves

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

Comandos do statement 2 do primeiro if

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

Comandos do statement 2 do primeiro if

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

Comandos do statement 2 do primeiro if

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
}
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

Comandos do statement 2 do primeiro if

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
} else
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

Comandos do statement 2 do primeiro if

```
if (expression)
        statement1
else
        statement2
```

Aninhamento de if

```
if (condição 1) {
   seqüência de comandos 1;
} else if(condição 2) {
   seqüência de comandos 2;
} else if
. . .
else if(condição_n) {
   seqüência de comandos n;
} else{
   seqüência de comandos default;
}
```

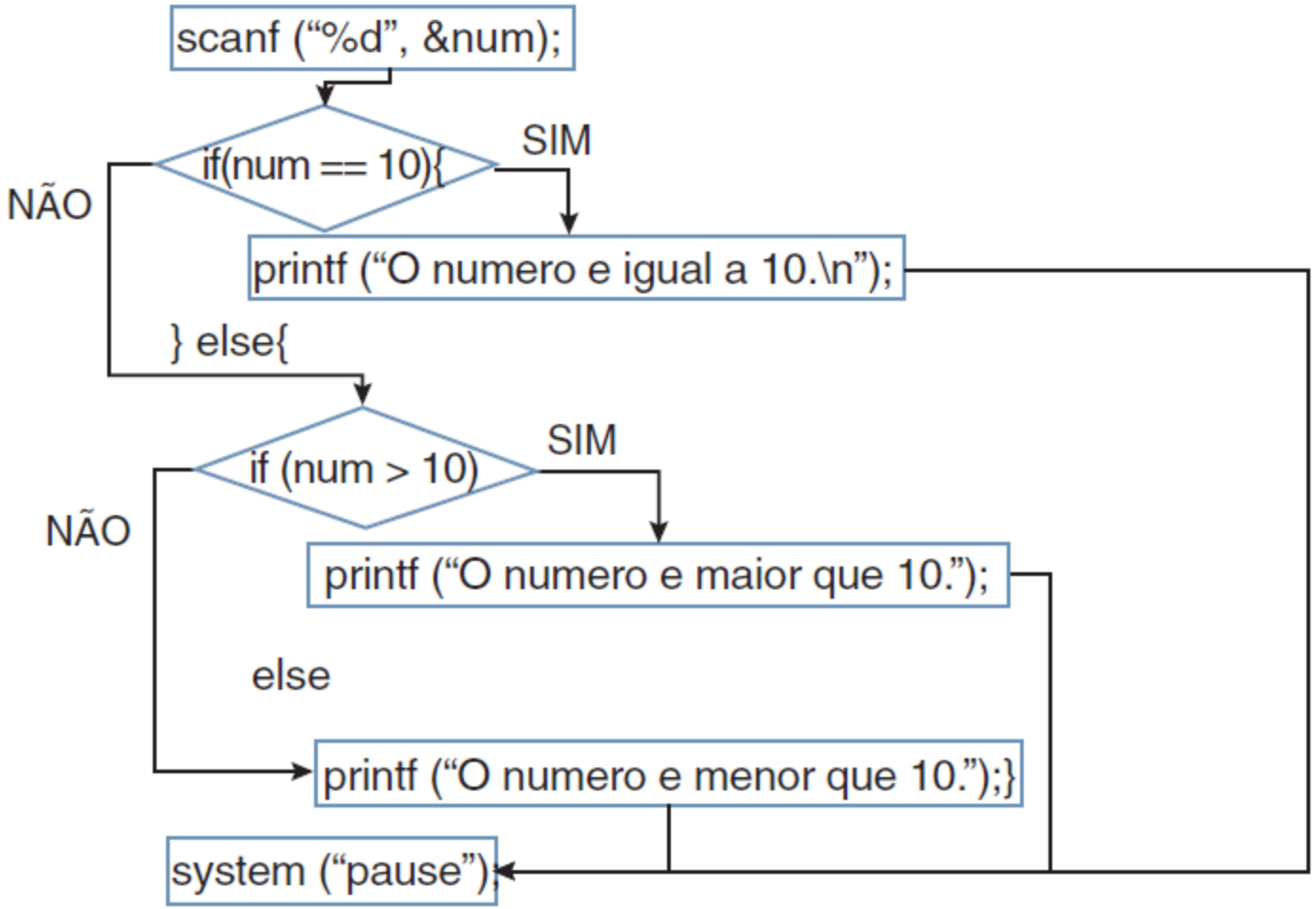
Comandos do statement 2 do primeiro if

```
if (expression)
        statement1
else
        statement2
```

Exemplo aninhamento

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int num;
05    printf("Digite um numero: ");
06    scanf("%d", &num);
07    if(num == 10){
08       printf("O numero e igual a 10.\n");
09    } else{
10      if(num > 10)
11           printf("O numero e maior que 10.\n");
12      else
13           printf("O numero e menor que 10.\n");
14    }
15    system("pause");
16    return 0;
17  }
```

Exemplo aninhamento



Aninhamento de if

- Observe sempre a correspondência entre if's e else's

```
if (cond1)
    if (cond2)
       comando if2;
    else
       comando if1;
```

Errado, pois o comando if1 está associado ao segundo if, e não ao primeiro

```
if (cond1) {
    if (cond2)
    comando if2;
} else
    comando if1;
```

Correto. Agora o comando if1 está associado ao primeiro if

Aninhamento de if

- Não existe aninhamento de else's
  - Para cada else deve existir um if anterior, mas nem todo if precisa ter um else.

```
if (cond1)
    comando if1;
    else
    comando else1;
    else
    comando else2;
```

Errado

Exercício

- Dada o valor da nota de um aluno, monte o conjunto de if’s e else’s que verifica se ele foi aprovado, reprovado ou precisará fazer a sub.

Exercício

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d < 30){
    printf (“Aluno reprovado”);
  } else {
    printf (“O aluno deve fazer a prova sub”);
  }
}
```

Erros encontrado - #1

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
}
  if (d < 30){
    printf (“Aluno reprovado”);
  } else {
    printf (“O aluno deve fazer a prova sub”);
  }
```

Erros encontrado - #1

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d < 30){
    printf (“Aluno reprovado”);
  } else {
    printf (“O aluno deve fazer a prova sub”);
  }
}
>> Sem o else, o aluno com nota >= 60 tem que
fazer sub, mesmo aprovado!
```

Erros encontrado - #2

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d < 30); {
    printf (“Aluno reprovado”);
  } else (“O aluno deve fazer a prova sub”);{
  }
}
```

Erros encontrado - #2

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d < 30); {
    printf (“Aluno reprovado”);
  } else {
    printf(“O aluno deve fazer a prova sub”);
  }
}
```

>> O “;” após o if finaliza o comando

>> faltou o comando printf e também o local do comando estava errado

Erros encontrado - #3

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d > 60){
  printf (“Aluno aprovado");
} else {
  if (d < 30){
    printf (“Aluno reprovado”);
  } else {
    printf (“O aluno deve fazer a prova sub”);
  }
}
```

Erros encontrado - #3

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d < 30){
    printf (“Aluno reprovado”);
  } else {
    printf (“O aluno deve fazer a prova sub”);
  }
}
```

>> Maior ou igual é muito diferente de Maior. Muitos alunos reprovariam se o código errado fosse utilizado!

Erros encontrado - #4

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d >= 30 && d < 60){
    printf (“O aluno deve fazer a prova sub”);
  } else {
    printf (“Aluno reprovado”);
  }
}
```

Erros encontrado - #4

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d >= 30 && d < 60){
    printf (“O aluno deve fazer a prova sub”);
  } else {
    printf (“Aluno reprovado”);
  }
}
```

>> Não chega a ser um erro, mas é um teste desnecessário. Com certeza no segundo IF a nota é menor que 60. Muitos alunos cometeram esse erro

Erros encontrado - #5

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else (d < 30){
    printf (“Aluno reprovado”);
  } else {
    printf (“O aluno deve fazer a prova sub”);
  }
```

Erros encontrado - #5

```
printf ("Digite a nota: ");
scanf ("%d",&d);

if (d >= 60){
  printf (“Aluno aprovado");
} else {
  if (d < 30){
    printf (“Aluno reprovado”);
  } else {
    printf (“O aluno deve fazer a prova sub”);
  }
}
```

>> Não coloque condição depois do else. Se quiser fazer algum teste, abra outro IF

Exercício

- Construir a seqüência de if-else para escrever o nome do dígito lido
  - ‘0’ -> “zero”;
  - ‘1’ -> “um”;
  - etc.

Exercício

```
char ch;
scanf(“%c”,&ch);
if (ch == ‘0’) printf(“Zero”);
else if (ch==‘1’) printf(“Um”);
else if (ch==‘2’) printf(“Dois”);
else if ...
else if (ch==‘9’) printf(“Nove”);
else printf(“Nao era um digito!”);
...
```

Expressão Condicional

- Quando o compilador avalia uma condição, ele quer um valor de retorno para poder tomar a decisão.
- Esta expressão não necessita ser uma expressão no sentido convencional.
- Uma variável sozinha pode ser uma "expressão" e esta retornar o seu próprio valor.

Expressão Condicional

- Isto quer dizer que teremos as seguintes expressões:

```
int num;
if (num!=0)
if (num==0)
```

- equivalem a

```
int num;
if (num)
if (!num)
```

Importante

- Símbolo de atribuição **=** é diferente, muito diferente, do operador relacional de igualdade **==**

```c
int Nota;

Nota == 60; // Nota é igual a 60?
Nota = 50;  // Nota recebe 50

// Erro comum em C:
// Teste se a nota é 60
// Sempre entra na condição
if (Nota = 60) {
  printf("Você passou raspando!!");
}

// Versão Correta
if (Nota == 60) {
  printf("Você passou raspando!!");
}
```

Importante

- Símbolo de atribuição **=** é diferente, muito diferente, do operador relacional de igualdade **==**
- Por que sempre entra na condição?

```
if (Nota = 60) {
  printf("Você passou raspando!!");
}
```

- Ao fazer Nota = 60 (“Nota recebe 60”) estamos atribuindo um valor inteiro à variável Nota.
- O valor atribuído **60 é diferente de Zero**. Como em C os booleanos são números inteiros, então vendo Nota como booleano, essa assume **true**, uma vez que é diferente de zero

Limpando o buffer

- Antes de usar um scanf com “%c” faça
- **setbuf**(**stdin**, **NULL**);
- Esse comando limpa o buffer de entrada

```
char letra;
setbuf(stdin,NULL);
scanf("%c",&letra);
```

O comando switch

- O comando switch é próprio para se testar uma variável em relação a diversos valores pré-estabelecidos.
  - Parecido com if-else-if, porém não aceitas expressões, apenas constantes.
  - O switch testa a variável e executa a declaração cujo *case* corresponda ao valor atual da variável.

O comando switch

```
switch (expressão) {
case valor1:
   comandos1;
   break;
case valor2:
   comandos2;
   break;
   ....
case valor k:
   comandos k;
   break;
default:
   comandos default;
}
```

```
switch (expression) {
  case const1:
       statement1
       break;
  case const2:
       statement2
       break;
....
  case const_k:
       statement_k;
       break;
  default:
     statement;
}
```

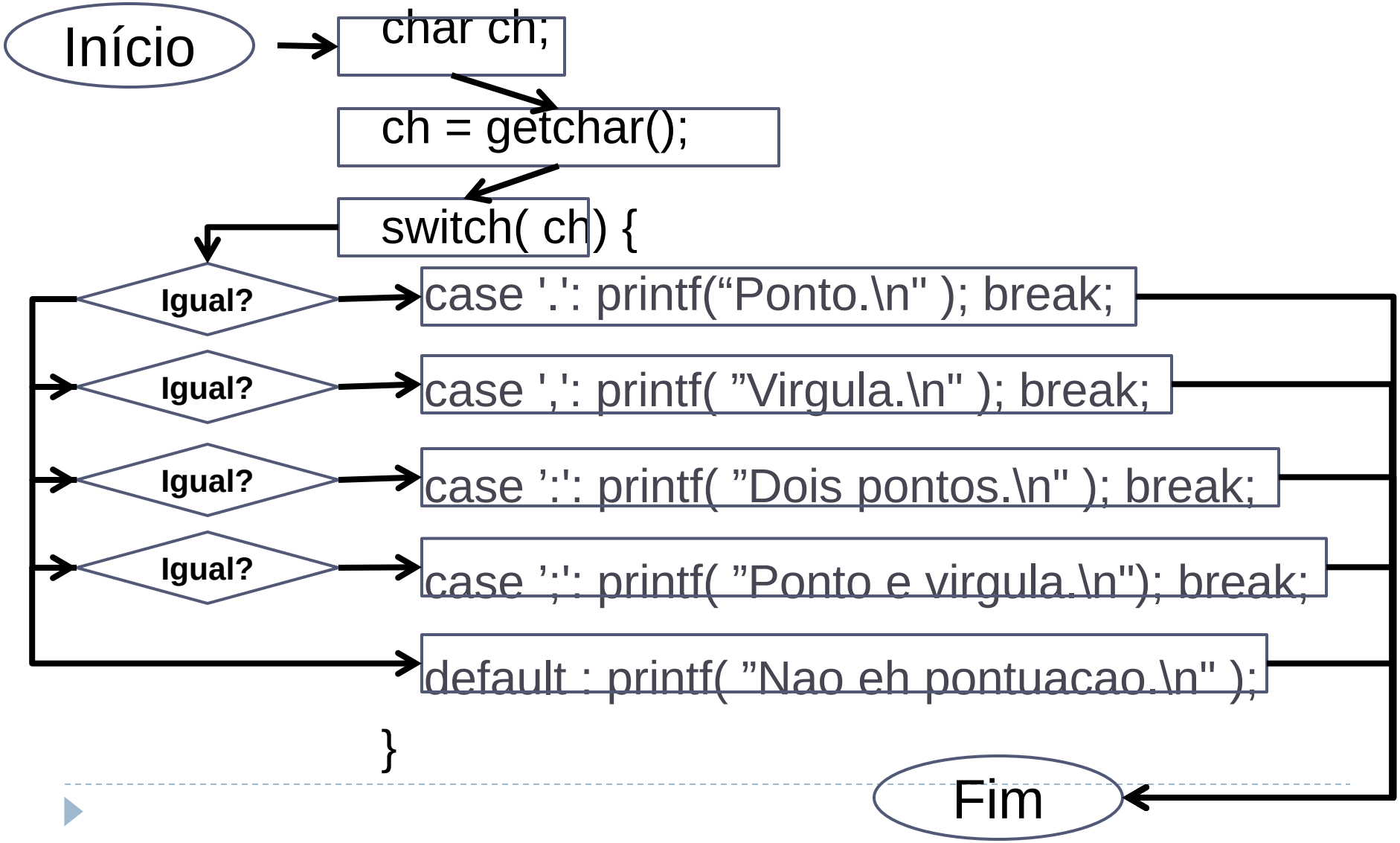
O comando switch

- O comando switch
  - Avalia o valor de `expression` com os valores associados às cláusulas `case` em seqüência;
  - Quando o valor associado a uma cláusula é igual ao valor de `expression` os respectivos comandos são executados até encontrar um `break`.
- A declaração `default` é opcional e será executada apenas se a expressão `expression` que está sendo testada não for igual a nenhuma das constantes presentes nos `case`.

O comando switch

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    char ch;
05    printf("Digite um simbolo de pontuacao: ");
06    ch = getchar();
07    switch( ch ) {
08        case '.': printf("Ponto.\n" ); break;
09        case ',': printf("Virgula.\n" ); break;
10        case ':': printf("Dois pontos.\n" ); break;
11        case ';': printf("Ponto e virgula.\n"); break;
12        default : printf("Nao eh pontuacao.\n" );
13    }
14    system("pause");
15    return 0;
16  }
```

O comando switch



O comando switch

- O comando break
  - Faz com que o switch seja interrompido assim que uma das seqüência de comandos seja executada.
  - Não é essencial. Se após a execução da declaração não houver um break, o programa continuará executando o próximo comando case.
  - Isto pode ser útil em algumas situações, mas tenha cuidado.

Mais um comando de entrada

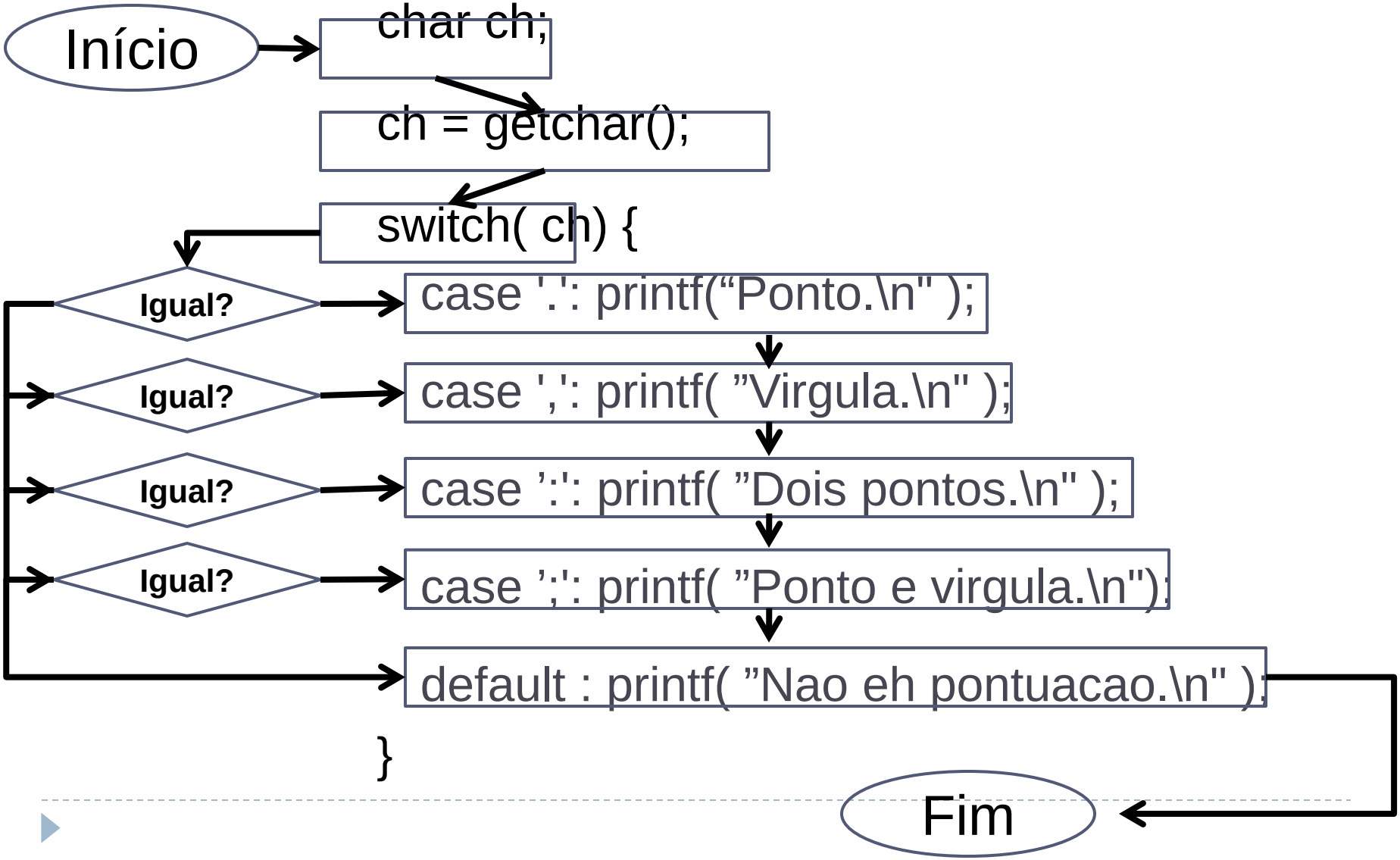
- **getchar()**
  - Comando que realiza a leitura de um único caractere

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    char c;
05    c = getchar();
06    printf("Caractere: %c\n", c);
07    printf("Codigo ASCII: %d\n", c);
08    system("pause");
09    return 0;
10  }
```

O comando switch sem break

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    char ch;
05    printf("Digite um simbolo de pontuacao: ");
06    ch = getchar();
07    switch( ch ) {
08        case '.': printf("Ponto.\n" );
09        case ',': printf("Virgula.\n" );
10        case ':': printf("Dois pontos.\n");
11        case ';': printf("Ponto e virgula.\n");
12        default : printf("Nao eh pontuacao.\n" );
13    }
14    system("pause");
15    return 0;
16  }
```

O comando switch sem break



O comando switch

```
int num;
scanf("%d",&num);

switch( num ) {
        case 0: printf("0"); /* 0123456789 */
        case 1: printf("1"); /* 123456789 */
        case 2: printf("2"); /* 23456789 */
        case 3: printf("3"); /* 3456789 */
        case 4: printf("4"); /* 456789 */
        case 5: printf("5"); /* 56789 */
        case 6: printf("6"); /* 6789 */
        case 7: printf("7"); /* 789 */
        case 8: printf("8"); /* 89 */
        case 9: printf("9"); /* 9 */
}
```

Exercício

- Construir o switch para escrever o nome do dígito lido
  - 0 -> “zero”;
  - 1 -> “um”;
  - etc.

Exercício

```
switch( num ) {
        case 0: printf("Zero"); break;
        case 1: printf("Um"); break;
        case 2: printf("Dois"); break;
        case 3: printf("Tres"); break;
        case 4: printf("Quatro"); break;
        case 5: printf("Cinco"); break;
        case 6: printf("Seis"); break;
        case 7: printf("Sete"); break;
        case 8: printf("Oito"); break;
        case 9: printf("Nove"); break;
}
```

Material Complementar

- Vídeo Aulas
  - Aula 13: Comando If
  - Aula 14 : Comando Else
  - Aula 15: Aninhamento If-Else
  - Aula 16: Operador Ternário(?)
  - Aula 17: Comando Switch

- Slides Adicionais (poderão ser vistos em uma aula futura)

O Operador ?

- Também conhecido como operador ternário
- A expressão condicional “? :” é uma simplificação do if-else utilizada tipicamente para atribuições condicionais

O Operador ?

- Uma expressão como

```
if (a > 0)
    b = -150;
else
    b = 150;
```

- pode ser simplificada usando-se o operador ? da seguinte maneira:

```
b = a > 0 ? -150 : 150;
```

Exercício

- Dado dois números x e y, retorne o maior na variável z:
  - Usando if-else
  - Usando o operador ternário

Exercício

	Usando if-else	Usando operador ternário
01	`#include <stdio.h>`	`#include <stdio.h>`
02	`#include <stdlib.h>`	`#include <stdlib.h>`
03	`int main(){`	`int main(){`
04	`  int x,y,z;`	`  int x,y,z;`
05	`  printf("Digite x:");`	`  printf("Digite x:");`
06	`  scanf("%d",&x);`	`  scanf("%d",&x);`
07	`  printf("Digite y:");`	`  printf("Digite y:");`
08	`  scanf("%d",&y);`	`  scanf("%d",&y);`
09	`  if(x > y)`	`  z = x > y ? x : y;`
10	`      z = x;`	`  printf("Maior = %d\n",z);`
11	`  else`	`  system("pause");`
12	`      z = y;`	`  return 0;`
13	`  printf("Maior = %d\n",z);`	`}`
14	`  system("pause");`	
15	`  return 0;`	
16	`}`	

O Operador ?

- O operador ? é limitado
  - não atende a uma gama muito grande de casos.
- mas pode ser usado para simplificar expressões complicadas. Uma aplicação interessante é a do contador circular.
  - index = (index== 3) ? index= 0: ++index;