Linguagens de Programação I

Tema # 5

Estruturas de Controle

INTRODUÇÃO

- Estruturas do controle são utilizadas para especificar a ordem em que as instruções devem ser executadas.
- Tipos de estruturas de controle:
  1. estrutura de seqüência,
  2. estrutura de seleção,
  3. estrutura de repetição.
- A linguagem C tem apenas 7 estruturas de controle.

ESTRUTURA DE SEQUENCIA

- esta essencialmente inserida na linguagem C,
- a menos que seja especificado de outra forma, o computador executa automaticamente as instruções segundo a estrutura de seqüência, isto é, uma apos a outra, de acima para baixo.

ESTRUTURA DE SELEÇÃO

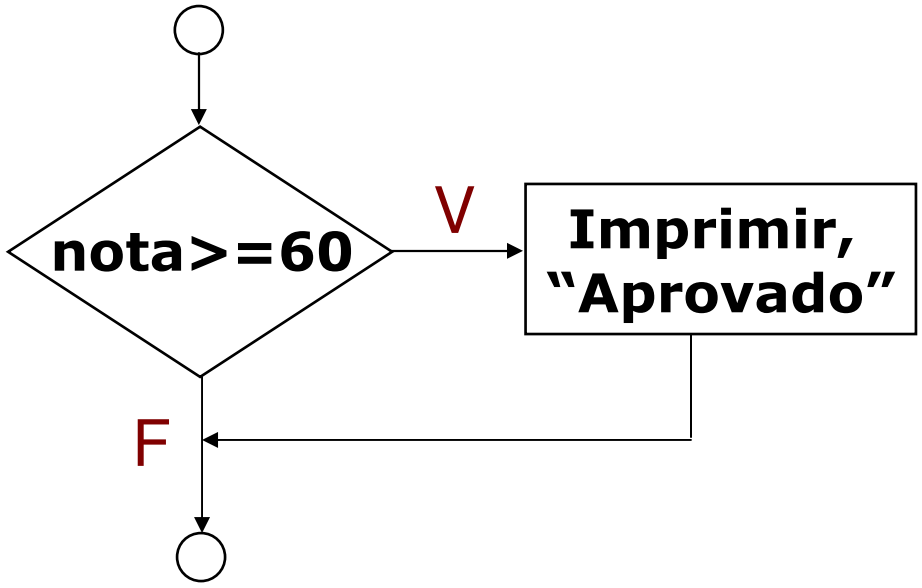
- permitem escolher blocos de instruções diferentes (caminhos diferentes durante a execução),
- a linguagem C fornece três tipos de estruturas de seleção:
  1. estrutura de seleção simples,
  2. estrutura de seleção dupla,
  3. estrutura de seleção múltipla.

ESTRUTURA DE SELEÇÃO

1. **Estrutura de seleção simples** ***if***, seleciona ou ignora um bloco de instruções.

- sintaxe:

if(condição)
{Instruções}




```
if (nota >= 60)
  printf("Aprovado\n");
```

ESTRUTURA DE SELEÇÃO

2. **Estrutura de seleção dupla** `if\else`, seleciona entre dois blocos de instruções diferentes.

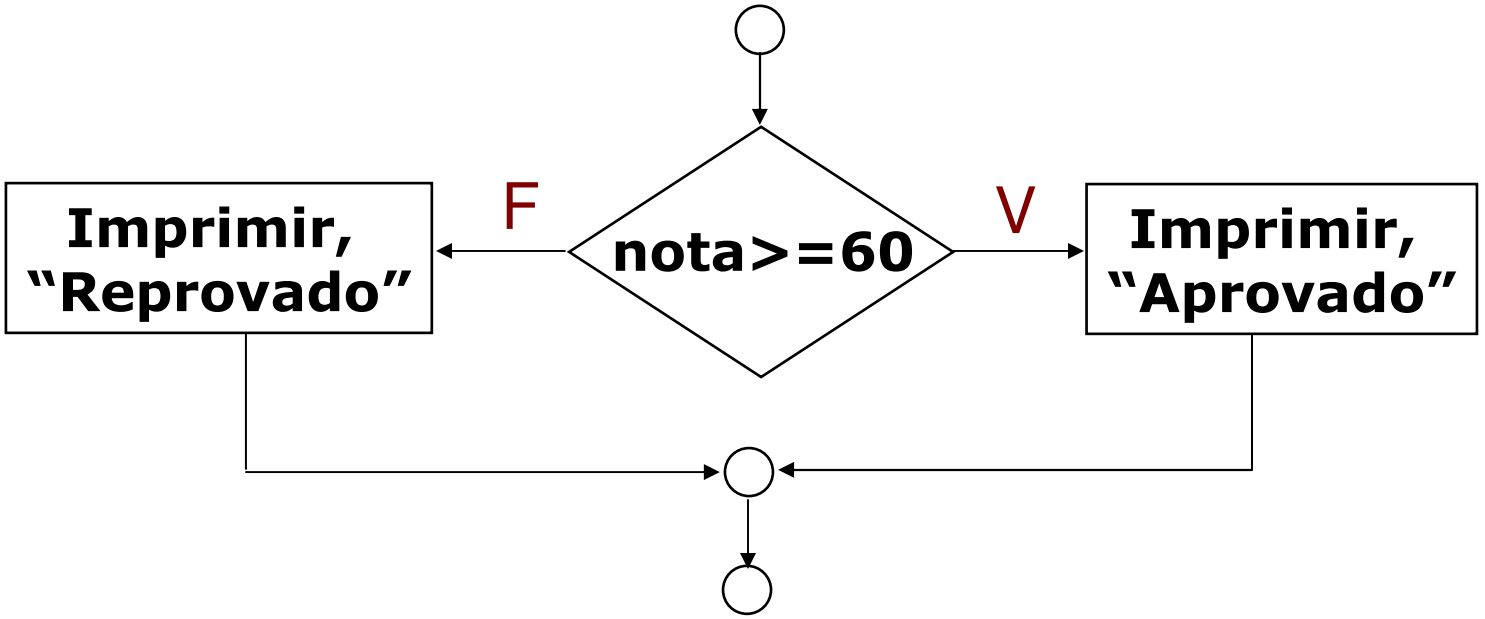
- sintaxe:

if(condição)
{Instruções}
else
{Instruções}

```
if (nota >= 60)
  printf("Aprovado\n");
else
  printf("Reprovado\n");
```

ESTRUTURA DE SELEÇÃO

2. Estrutura de seleção dupla . . .



3. Estrutura de seleção múltipla `switch`, seleciona entre muitos blocos de instruções diferentes.

ESTRUTURAS DE REPETIÇÃO

- permitem realizar uma instrução ou um bloco de instruções várias vezes.
- A linguagem C fornece três tipos de estruturas de repetição:
  1. estrutura **`while`**,
  2. estrutura **`do\while`**,
  3. estrutura **`for`**.

ESTRUTURAS DE REPETIÇÃO

- **Estrutura `while`**
- permite ao programador especificar que uma ação deve ser repetida enquanto uma determinada condição for verdadeira.
- Exemplo: lista de compras

Enquanto (houver itens em minha lista de compras)
comprar o próximo item
riscá-lo de minha lista

ESTRUTURAS DE REPETIÇÃO

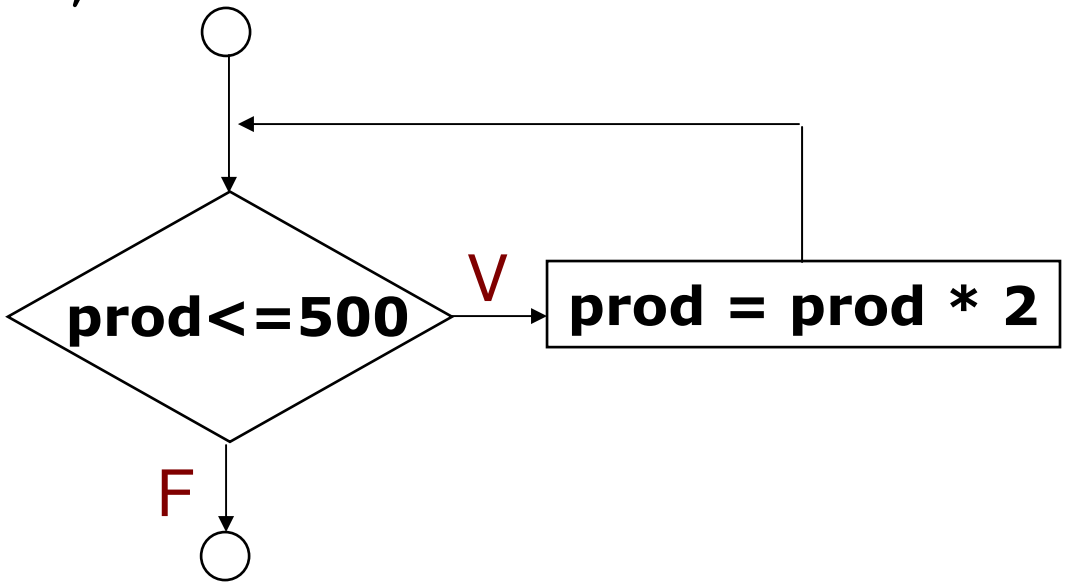
- **Estrutura `while ...`**
- sintaxe:

  while (condição){
  bloco de instruções
  }
- Execução de um ciclo `while`
- Considere o seguinte trecho de programa para encontrar a primeira potência de 2 maior que 500.

ESTRUTURAS DE REPETIÇÃO

- **Estrutura `while ...`**

```
prod = 2;
while (prod <= 500){
   prod = prod * 2;
   }
```



- Inicialização
- Ciclos Infinitos !!!

REPETIÇÃO CONTROLADA POR CONTADOR

- também chamada de repetição definida,
- é conhecido o número de vezes que o ciclo pode ser executado,
- Exemplo: uma turma de dez alunos fez um teste, leia as notas (inteiros de 0 a 100) da turma e determine a média da turma no teste.

```
/* Programa para calcular a media
   de uma turma de 10 alunos  */

#include <stdio.h>
#include <stdlib.h>
#define N 10

int main()
{
  int cont, nota, total, media;

  /* fase de inicialização */
  total = 0;
  cont = 1;

  /* fase de procesamento */
  while (cont <= N){
    printf("Entre com a nota: ");
    scanf("%d", &nota);
    total = total + nota;
    cont++;
  }
```

```
/* fase de resultados */
  media = total/N;
  printf("A media da turma e %d\n",media);

  system("PAUSE");
  return 0;
}
```

```
Entre com a nota: 90
Entre com a nota: 85
Entre com a nota: 100
Entre com a nota: 70
Entre com a nota: 55
Entre com a nota: 60
Entre com a nota: 38
Entre com a nota: 89
Entre com a nota: 97
Entre com a nota: 65
A media da turma e 74
Press any key to continue . . .
```

REPETIÇÃO CONTROLADA POR SENTINELA

- também chamada de repetição indefinida,
- sentinela ou flag: valor sinalizador utilizado para indicar o final de um processo (e. g. entrada de dados),
- o valor do sentinela deve ser escolhido fora do domínio dos dados de entrada.
- Exemplo: desenvolva um programa para calcular a média de uma turma num teste, o programa deve processar um numero arbitrário de alunos cada vez que for executado.

```
/* Programa para calcular a media de uma turma
   com numero arbitrario de alunos */
#include <stdio.h>
#include <stdlib.h>

int main()
{
  float media;
  int cont, nota, total;

  /* fase de inicialização */
  total = 0;
  cont = 0;

  /* fase de procesamento */
  printf("Entre com a nota (-1 para finalizar): ");
  scanf("%d", &nota);
  while(nota != -1){
    total = total + nota;
    cont = cont + 1;
    printf("Entre com a nota (-1 para finalizar): ");
    scanf("%d", &nota);
  }
```

```
  /* fase de resultados */
  if (cont){
    media = (float) total/cont;
    printf("A media da turma e %.2f\n", media);
  }
  else
    printf("Nenhum grau foi fornecido\n");

  system("PAUSE");
  return 0;
}
```

```
Entre com a nota (-1 para finalizar): 75
Entre com a nota (-1 para finalizar): 89
Entre com a nota (-1 para finalizar): 36
Entre com a nota (-1 para finalizar): 58
Entre com a nota (-1 para finalizar): 92
Entre com a nota (-1 para finalizar): 100
Entre com a nota (-1 para finalizar): 84
Entre com a nota (-1 para finalizar): -1
A media da turma e 76.29
Press any key to continue . . .
```

REPETIÇÃO

Controlada por contador **vs** **Controlada por sentinela**

OPERADORES DE ATRIBUIÇÃO COMPOSTOS

- considere: `int a=1, b=2, c=3, d=4, e=5;`

Operador	Exemplo	Operação	Resultado
`+=`	`a += 7`	`a = a + 7`	`8`
`-=`	`b -= 2`	`b = b - 2`	`0`
`*=`	`c *= 5`	`c = c * 5`	`15`
`/=`	`d /= 2`	`d = d / 2`	`2`
`%=`	`e %= 2`	`e = e % 2`	`1`

OPERADOR CONDICIONAL [?:]

- é utilizado para substituir a estrutura de seleção **`if/else`** em operações simples.
- Sintaxe:

(condição) ? instrução A : instrução B;

- Exemplos:

```
b = (a > 0) ? a++ : a--;

b = (c > 0) ? sqrt(c) : sqrt(-c);

(grau >= 60) ? printf("Aprdo.\n") : printf("Reprd.\n");
```

EXERCÍCIO

- procure erros nos seguintes fragmentos de programas:

```
 if (idade >65);
    printf("Idade e maior que 65\n");
  else
  printf("Idade e menor que 65\n");
```

```
int x = 1, total;

while (x<=10){
  total += x;
  ++x;
}
```

```
 while (y>0){
   printf("%d\n",y);
   ++y;
 }
```

```
While (x<=100)
  total += x;
  ++x;
```

```
int main()
{
  int num1, num2, res=0, i=0, s, s1=1, s2=1;
  printf("Digite Numero 1:");
  scanf("%d", &num1);
  printf("Digite Numero 2:");
  scanf("%d", &num2);
  if (num1<0){
    num1 = -num1;
    s1 = -1;
  }
  if (num2<0){
    num2 = -num2;
    s2 = -1;
  }
  s = s1 * s2;
  while(i<num2){
    res += num1;
    i++;
  }
  printf("Resultado %d\n", res*s);
  system("pause");
  return 0;}
```

REPETIÇÃO CONTROLADA POR CONTADOR

- Repetição controlada por contador exige:

1. O *nome* de uma variável de controle,
2. O *valor inicial* da variável de controle,
3. O *incremento* (ou *decremento*) pelo qual a variável de controle é modificada cada vez que o corpo do laço é realizado.
4. A condição que testa o *valor final* da variável de controle.

REPETIÇÃO CONTROLADA POR CONTADOR

- Exemplo de repetição controlada por contador:

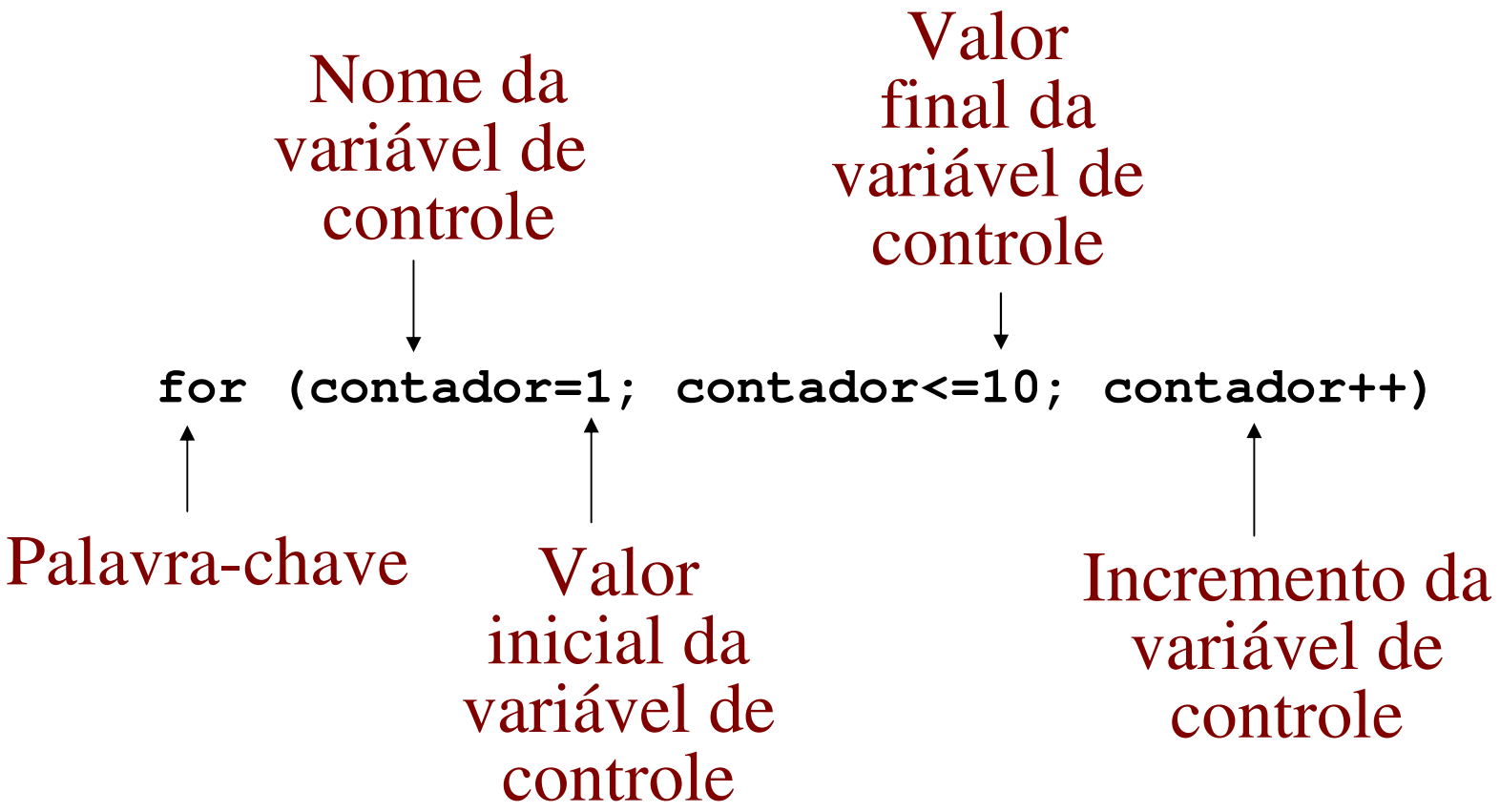
```
int main(){
  int contador=1;                    /* inicialização */
  while (contador<=10){              /* condição */
    printf("%d\n", contador);
    contador++;                      /* incremento */
  }

  return 0;
}
```

ESTRUTURA DE REPETIÇÃO **for**

- A estrutura de repetição **`for`** manipula automaticamente todos os detalhes da repetição controlado por contador.
- Repetimos o exemplo anterior utilizando **`for`**.

```
int main(){
  int contador;
  /* inicialização, condição e incremento
     estão incluídos no cabeçalho da estrutura */
  for (contador=1; contador<=10; contador++)
    printf("%d\n", contador);
  return 0;
}
```

ESTRUTURA DE REPETIÇÃO **for**



ESTRUTURA DE REPETIÇÃO **for**

- Sintaxe *for (expressão1; expressão2; expressão3)*
  {bloco de instruções}
- *expressão1* = inicialização
- *expressão2* = condição
- *expressão3* = incremento
- Execução da estrutura `for`
- Na maioria dos casos, existe uma equivalencia entre as estruturas `for` e `while`

```
expressão1
while(expressão2){
   instrução
   expressão3
}
```

ESTRUTURA DE REPETIÇÃO **for**

- Freqüentemente expressão1, expressão2 e expressão3 são expressões múltiplas separadas por vírgulas, onde a lista de expressões é avaliada da esquerda para a direita.
- As três expressões da estrutura for são opcionais e podem ser omitidas.
- Se a expressão2 for omitida, a linguagem C presume que a condição é verdadeira e cria um loop infinito.

ESTRUTURA DE REPETIÇÃO **for**

- A expressão1 pode ser omitida se a variável de controle for inicializada em outro lugar do programa.
- A expressão3 pode ser omitida se o incremento é calculado no corpo da estrutura `for` o se nenhum incremento se faz necessário.

ESTRUTURA DE REPETIÇÃO **for**

- Exemplos:

1. A inicialização, condição e continuação do loop podem conter operações aritméticas.

```
x = 2; y = 10
for(j=x; j<=4*x*y; j+=y/x)


for(j=2; j<=80; j+=5)
```

2. O "incremento" pode ser negativo, neste caso temos decremento ou contagem regressiva.

```
for(i=10; i>=0; i--)
  printf("%d\n", i);
```

ESTRUTURA DE REPETIÇÃO **for**

- Exemplos:

3. Se a condição de continuação for inicialmente falsa, o corpo do loop nunca é executado, a execução continua na próxima instrução após o loop.

```
for(i=k; i<k; i++)
  printf("%d\n", i);
```

4. Laço infinito.

```
for(;;){
  ch = getchar();
  if (ch=='A') break;
}
```

ESTRUTURA DE REPETIÇÃO **for**

- Exemplos:

5. Laço `for` com condição composta

```
flag = 1;
for(i=0; i<10 && flag; i++){
  printf("Digite um numero positivo:");
  scanf("%d", num);
  if (num<0) flag = 0;
}
```

6. Laço `for` como retardo de tempo

```
for(i=0; i<1000; i++);
```

ESTRUTURA DE REPETIÇÃO **for**

- Exemplos:

7. Laço `for` sem corpo

```
for(soma=0, num=2; num<=100; soma+=num, num+=2);
```

```
soma=0;
for(num=2; num<=100; num=num+2)
  soma+=num;
```

8. Laço `for` utilizando caracteres

```
for(char letra='A'; letra<='Z'; letra++)
  printf("%d %c", letra, letra);
```

ESTRUTURA DE REPETIÇÃO **for**

65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N

79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z

ESTRUTURA DE REPETIÇÃO **for**

- Exercícios

1. Fazer a variável de controle assumir valores de 1 a 100 em incrementos de 1.
2. Fazer a variável de controle assumir os múltiplos de 7 desde 7 até 70.
3. Fazer a variável de controle assumir os valores da seguinte seqüência: 20, 17, 14, 11, 8, 5, 2.
4. Fazer a variável de controle assumir os valores da seguinte seqüência: 2, 4, 8, 16, 32, 64, 128, 256, 512.

ESTRUTURA DE REPETIÇÃO **for**

- Exercícios

Uma pessoa investe 1000 reais em uma conta de poupança que rende juros do 5 por cento. Admitindo que todos os juros são deixados em depósito na conta, calcule e imprima a quantia na conta ao final de cada ano, ao longo dos anos. Use a fórmula de juros compostos:

$$a = p(1+r)^n$$

p, quantia invertida originalmente,
r, taxa anual de juros,
n, numero de anos,
a, quantia existente em depósito ao final do ano n.

Resposta

```
/* Calculando juros compostos */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
  int ano;
  double quant, princ=1000.0, taxa=.05;

  printf("%4s%21s\n", "Ano", "Saldo na conta");
  for(ano=1; ano<=10; ano++){
    quant = princ*pow(1.0+taxa, ano);
    printf("%4d%21.2f\n", ano, quant);
  }
return 0;
}
```

Resposta

Ano	Saldo na conta
1	1050.00
2	1102.50
3	1157.63
4	1215.51
5	1276.28
6	1340.10
7	1407.10
8	1477.46
9	1551.33
10	1628.89

ESTRUTURAS DE SELEÇÃO

- Estrutura de seleção simples (`if`)
- Estrutura de seleção múltipla (`if/else`)
- **Estrutura de seleção múltipla**: um algoritmo pode conter uma série de decisões nas quais uma variável ou expressão e testada separadamente para cada um dos valores constantes que ela pode assumir, e com base nisso diferentes ações são tomadas.
- **Exemplo**: *Crie um programa para processar as notas de uma exame (A, B, C, D, E), informe a quantidade de alunos que obtiveram cada nota.*

```
int main()
{
  char nota;
  int Ca = 0, Cb = 0, Cc = 0, Cd = 0, Ce = 0;

  printf("Entre com as notas.\n");
  printf("Entre com os carateres EOF para finalizar.\n");
  while((nota=getchar()) != EOF ){
    if((nota=='A')||(nota=='a')) Ca++;
    else
      if((nota=='B')||(nota=='b')) Cb++;
      else
        if((nota=='C')||(nota=='c')) Cc++;
        else
          if((nota=='D')||(nota=='d')) Cd++;
          else
            if((nota=='E')||(nota=='e')) Ce++;
            else
              if((nota=='\n')||(nota==' '));
              else
                printf("Nota incorreta.\n");
  }
```

```
  printf("\nOs totais de cada nota sao:\n");
  printf("A: %d\n", Ca);
  printf("B: %d\n", Cb);
  printf("C: %d\n", Cc);
  printf("D: %d\n", Cd);
  printf("E: %d\n", Ce);

  system("PAUSE");
  return 0;
}
```

- Função **`getchar()`**, lê um caractere do teclado e retorna esse caractere, no exemplo ele é armazenado na variável **`nota`**.
- ***EOF***, End of File, constante simbólica definida pelo padrão ANSI, em MS-DOS pode ser gerada utilizando `Ctrl-z`.

```
while((nota=getchar()) != EOF ){
  switch(nota){
    case 'A': case 'a':
      Ca++;
      break;
    case 'B': case 'b':
      Cb++;
      break;
    case 'C': case 'c':
      Cc++;
      break;
    case 'D': case 'd':
      Cd++;
      break;
    case 'E': case 'e':
      Ce++;
      break;
    case '\n': case ' ':
      break;
    default:
      printf("Nota incorreta.\n");
      break;
  }
}
```

Exemplo anterior utilizando a estrutura de seleção múltipla **`switch`**

ESTRUTURA DE SELEÇÃO MULTIPLA

switch

- Sintaxe:

```
switch(expressão){
   case constante1:
        seqüência de comandos
        break;
   case constante2:
        seqüência de comandos
        break;
  case constante3:
        seqüência de comandos
        break;
  .
  .
  .
   default:
        seqüência de comandos
}
```

ESTRUTURA DE SELEÇÃO MULTIPLA

switch

- O `switch` testa sucessivamente o valor de uma expressão contra uma lista de **constantes inteiras**, quando o valor coincide os comandos associados àquela constante são executados.
- Embora `case` seja uma palavra reservada da linguagem ela não pode se utilizada fora da estrutura `switch`.
- O `switch` solo pode testar igualdade.
- Duas constantes case no mesmo `switch` não podem ter valores idênticos.

ESTRUTURA DE SELEÇÃO MULTIPLA

switch

- Quando constantes caracteres são usadas no comando `switch` elas são automaticamente convertidas a inteiros.
- O comando `break`, é um comando de desvio, quando encontrado a execução continua na primeira linha de código apos o `switch`.
- O comando `default` é executado se nenhuma coincidência for detectada.

ESTRUTURA DE SELEÇÃO MULTIPLA

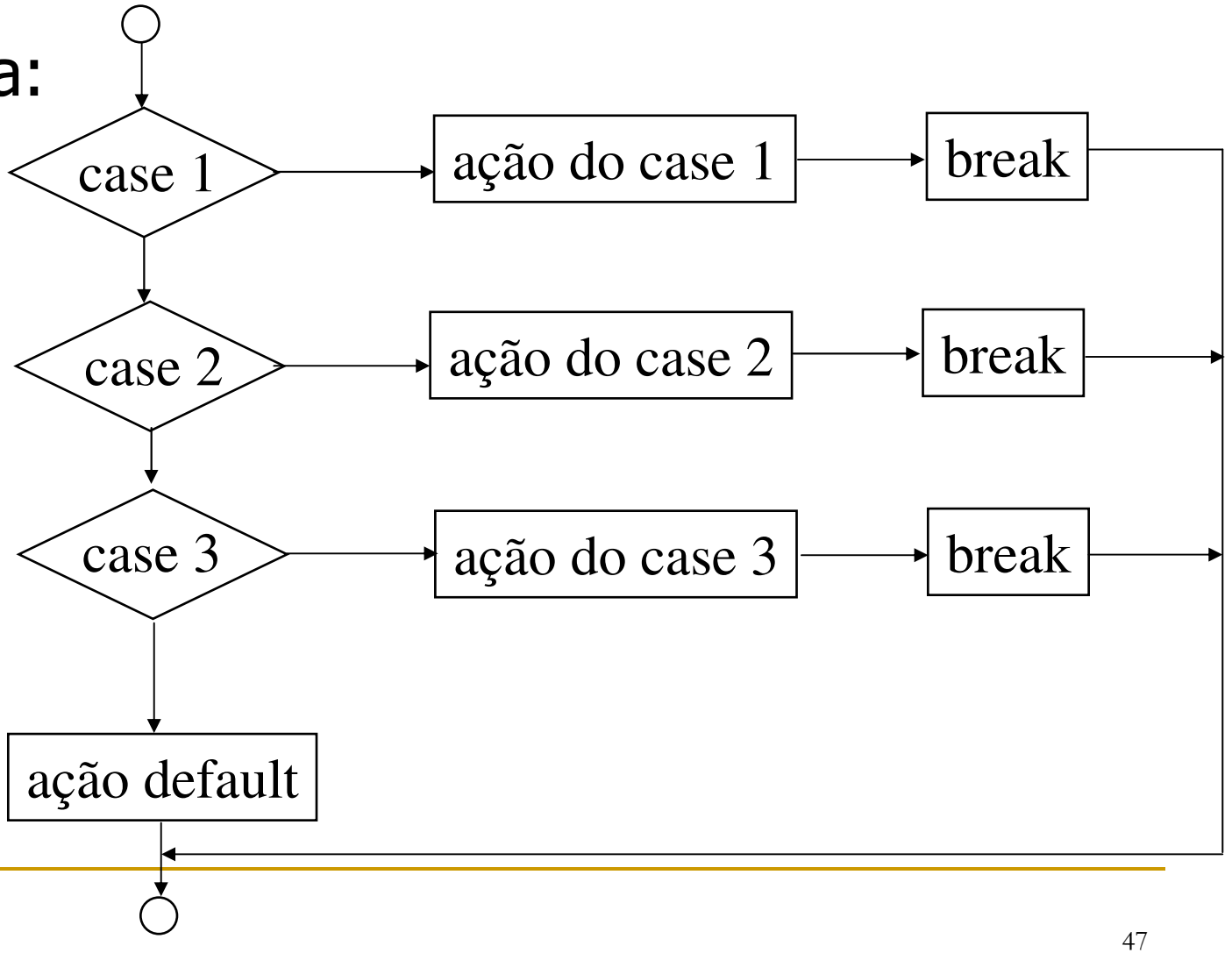
switch

- O `default` é opcional, se não estiver presente nenhuma ação será realizada se todos os testes falharem.
- Tecnicamente, os comandos `break` são opcionais, se o `break` for omitido, a execução continua pelos próximos comandos até que um `break` ou o fim do `switch` seja encontrado. A omissão do `break` pode levar a erros de lógica na execução.

ESTRUTURA DE SELEÇÃO MULTIPLA

switch

- Fluxograma:



ESTRUTURA DE REPETIÇÃO

do/while

- A estrutura **`do/while`** é similar a estrutura **`while`**.
- No **`while`** a condição de continuidade do laço é testada no inicio antes do corpo da estrutura ser executado.
- No **`do/while`** a condição de continuidade do laço é testada depois do corpo do laço se executado.
- O laço **`do/while`** é executado pelo menos uma vez.
- Sintaxe:

```
do{
   instrução
}while(condição)
```

ESTRUTURA DE REPETIÇÃO

do/while

- **Exemplos:**

```
int main(){
   int contador=1;
   while (contador<=10){
     printf("%d\n", contador);
     contador++;
   }
   return 0;
}
```

```
int main(){
   int contador=1;
   do{
     printf("%d\n", contador);
   }while(++contador<=10)
   return 0;
}
```

Criação de um menu

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

int main()
{
  char op;

  do{
    system("CLS");
    printf("(A) Exibir a listagem do diretorio.\n");
    printf("(B) Alterar a hora do sistema.\n");
    printf("(C) Alterar a data do sistema.\n");
    printf("(S) Sair.\n");
    printf("Escolha:");
    op = toupper(getchar());
```

```
switch(op){
      case 'A':  system("DIR");
                 system("PAUSE");
                 break;
      case 'B':  system("TIME");
                 system("PAUSE");
                 break;
      case 'C':  system("DATE");
                 system("PAUSE");
                 break;
      case 'S':
      case '\n':
      case ' ':  break;
      default:   printf("Opcao incorreta!!!\n");
                 system("PAUSE");
    }
  }while((op!='S'));

  system("PAUSE");
  return 0;
}
```

COMANDOS DE DESVIO

- A linguagem C tem quatro comandos que realizam desvio incondicional: **`return`**, **`break`**, **`continue`** e **`exit`**.
- **`return`**:
  1. é usado para retornar o valor de uma função.
  2. ele faz com que a execução volte ao ponto onde a chamada a função foi feita.
  3. é utilizada para terminar a execução de uma função.
  4. se for a função **`main`**, termina a execução do programa.

COMANDOS DE DESVIO

- **`break`**:

1. usado para terminar um case em um comando **`switch`**,
2. ou para forçar a terminação imediata de um laço,
3. em ambos casos o controle do programa retorna a próxima instrução após a estrutura (laço ou **`switch`**).

COMANDOS DE DESVIO

- **`continue`:**

1. utilizado para forçar a próxima iteração de um laço,
2. no laço **`for`**, o teste condicional e a expressão de incremento são executados,
3. nos laços **`while`** e **`do/while`** o controle do programa passa ao teste condicional (incremento?)

COMANDOS DE DESVIO

- **Exemplos:**

```
for(t=0; t<100; t++){
  if(t%2) continue;
  printf("%d",t);
}
```

```
for(t=0; t<100; t++){
  printf("%d",t);
  if(t==10) break;
}
```

```
i=0
while(i<10){
  printf("Digite Numero positivo");
  scanf("%d", &num);
  if (num<0) break;
  i++;
}
```

```
t=0;
while(t<100){
  if(t%2) continue;
  printf("%d",t);
  t++
}
```

A FUNÇÃO **exit()**

- A função `exit()` pode ser utilizada para sair de um programa, provocando a terminação imediata da execução do programa e retornando o controle ao sistema operacional.
- Sintaxe:

void exit(int codigo_de_retorno);

- **Exemplo:**

```
if (num<0) exit(-1);
```