DeVry Brasil | UNIFAVIP

Linguagem de Programação C
Aula 3

Professor: Rodrigo Rocha

Comandos Básicos

- Atribuição
- Controle de fluxo
  - if
  - switch
- Repetição
  - for
  - while
  - do
  - break
  - continue

Atribuição

- `int a=1, b, c; // Inicialização`
- `b=2; // Atribuição de constante`
- `b=a; // Atribuição de variável`
- `c=a+b; // Atribuição de expressão aritmética`

Atribuição de strings

- Dada a string **char s[10]**, **s** representa a posição de memória do caracter **s[0]**, ou seja, não podemos atribuir um valor string a ele
- Para Atribuir um valor ou uma variável a uma string utilizamos a função **strcpy**(destino, origem)
- A s funções de manipulação de string estão na biblioteca string.h
- Exemplo

```
char nome[30], x[30];
strcpy(nome, “teste”); // Atribui “teste” a nome
x=nome; // Coloca em x o endereço de memória de nome
```

Controle de fluxo: if

- Comando básico de controle de fluxo e tomada de decisão
- Formato geral:

```
if (condição) // Formato simplificado
  declaração

if (condição) // Formato completo
  declaração
else
  declaração
```

Comando: if - Exemplo

```
int div(int x, y) {
   int z=0;
   if (y != 0)
      z=(int) x/y;
   return(z);
}

int par(int x) {
   int r=0; // 0 = false
   if (x % 2 == 0) {
      printf("É par");
      r=1; // #0 = true
   }
   return(r);
}
```

```
int max (int x,y) {
   if (x>y)
      return(x);
   else
      return(y);
    }
```

Controle de fluxo: switch

- Comando de controle de fluxo e tomada de decisão
- Permite melhor performance em estruturas condicionais mais complexas
- Permite um código mais "limpo"

- Formato geral:

```
switch (variável) {
  case constante_1:
    declaração_1;
    break;
  case constante_2:
    declaração_2;
    break;
  ...
  case constante_n:
    declaração_n;
    break;
  default
    declaração_default;
  }
```

Comando: switch - Exemplo

```
if (a==1) {
  b=1;
}else if (a==2) {
  b=3;
}else if (a==3) {
  b=5;
}else if (a==4) {
  b=7;
}else
  b=0;
```

```
switch (a) {
  case 1:
    b=1;
    break;
  case 2:
    b=3;
    break;
  case 3:
    b=5;
    break;
  case 4:
    b=7;
    break;
  default
    b=0;
}
```

Repetição: for

- O loop (laço) **for** é usado para repetir um comando, ou bloco de comandos, diversas vezes, de maneira que se possa ter um bom controle sobre o loop

- Formato geral:

```
for (inicialização; condição; incremento)
  declaração;
```

- A declaração no comando for também pode ser um bloco ({ }) e neste caso o **;** pode ser omitido

- Todos os argumentos do comando for são opcionais

Repetição: for

- O melhor modo de se entender o loop **for** é ver de que maneira ele funciona "por dentro". O loop **for** é equivalente a se fazer o seguinte:

```
inicialização;
comando_if:
if (condição) {
  declaração;
  incremento;
  "Volte para o comando_if"
}
```

Comando: for - Exemplo

```
#include <stdio.h>

int main() {
   int count;
   for(count=1;count<=100;count=count+1)
     printf("%d ",count);
   return(0);
}
```

Repetição: while

- Executa um laço, comando ou bloco de comandos, enquanto uma condição for **verdadeira**

- Formato geral:

```
while(condição) declaração;
```

- Cuidado com o comando:

```
a = 9;
while (a<10); {
  a--;
}
```

Ele gera um loop infinito

Repetição: while

- O loop **while** é equivalente a se fazer o seguinte:

```
inicio_loop:
if (condição) {
  declaração;
  "Volte para o inicio_loop"
}
```

Comando: while - Exemplo

```
#include <stdio.h>

int main() {
   int count=1;
   while (count<=100) {
      printf("%d ",count);
      count++;
   }
   return(0);
}
```

Repetição: do-while

- Executa um laço, comando ou bloco de comandos, enquanto uma condição for **verdadeira**
  - Mesma definição do comando: while?
- Formato geral:

```
do {
   declaração;
} while (condição);
```

Repetição: do-while

- O loop **do-while** é equivalente a se fazer o seguinte:

```
inicio_loop:
declaração;
if (condição)
   "Volta para o inicio_loop"
```

Comando: do-while - Exemplo

```
#include <stdio.h>

int main() {
   int count=1;
   do {
      printf("%d ",count);
      count++;
   } while (count<100);
   return(0);
}
```

Comando: break

- Comando de controle de fluxo, utilizado dentro de comandos de loop
- Força a parada de um comando de laço (for, while ou do-while)
- Sai do laço e continua o processamento a partir do próximo comando após o laço
- Formato geral:

```
break;
```

Comando: break - Exemplo

```
#include <stdio.h>

int main() {
   int count=0;
   do {
      count++;
      if(count > 50) break;
      printf("%d ",count);
   } while (count<100);
   return(0);
}
```

COMANDO: CONTINUE

- Comando de controle de fluxo, utilizado dentro de comandos de loop
- Interrompe uma interação de um laço (for, while ou do-while)
- Interrompe o fluxo normal do laço e retorna para o início de uma nova interação
- No caso do comando **for** incrementa antes de voltar a nova interação
- Formato geral:

```
continue;
```

Comando: continue - Exemplo

```
#include <stdio.h>

int main() {
   int count=0;
   do {
      count++;
      if(count > 50 && count < 70) continue;
      printf("%d ",count);
   } while (count<100);
   return(0);
}
```

Exercícios

- Elaborar um programa para ler três números x, y e z, sem utilizar vetor, escrevê-los em ordem crescente. Tente fazer para quatro números: x, y, z e t.

- Elaborar um programa para imprimir todas as letras (de A a Z). Utilize um comando for para percorrer todas as letras.

- Elaborar um programa para ler uma relação de números, sem utilizar vetor, e escrever:
  - O valor do menor número
  - O valor do maior número
  - O valor médio

  Considere número = 0 para terminar a leitura e utilize o comando while para controle do loop.

- Elaborar um programa em C para ler um número entre 1 e 10 e escreve-lo na tela. Utilize um do-while para controlar o loop, caso o número seja inválido.