

# Unidade IV - Fluxo de Controle

Disciplina Linguagens de Programação I  
Bacharelado em Ciência da Computação da Uerj  
Professores Guilherme Mota e Leandro Marzulo

## ANSI C

```
#include <stdio.h>
main ()
{
    printf("Hello World!");
}
```

# **Conceitos Fundamentais**

# Comandos e Blocos

**;** terminador de comando  
**{ }** delimitadores de bloco

```
/* progPag039.c */
main() /* print longest input line; specialized version
*/
{
    int len; extern int max; extern char longest[];
    max = 0;
    while ((len = mygetline()) > 0)
        if (len > max)
        {
            max = len; copy();
        }
    if (max > 0) /* there was a line */
        printf("%s", longest);
    return 0;
}
```

# **Estruturas de seleção**

# `if - else`

- Estrutura de seleção
- Condiciona a execução de blocos de comandos ao valor de uma expressão
- Somente executa bloco a seguir se a expressão for verdadeira
- Se a condição for falsa será executado um comando (opcional) definido pelo usuário

# if - else

```
/* expressao != 0 ? */
```

```
    if (expressao)  
        comando-1;  
    else  
        comando-2;
```

```
/* note que é expressão e  
não condição*/
```

# if - else

```
/* bitcount: count 1 bits in x */  
int bitcount(unsigned x)  
{  
    int b;  
  
    for (b = 0; x != 0; x >>= 1)  
        if (x & 01)  
            b++;  
  
    return b;  
}
```

# else - if

```
if (expressao-1)
    comando-1;
else
    if (expressao-2)
        comando-2;
    else
        if (expressao-3)
            comando-3;
        else
            comando-4;
```



## `else - if` (Identação C mais comum)

```
if (expressao-1)
    comando-1;
else if (expressao-2)
    comando-2;
else if (expressao-3)
    comando-3;
else
    comando-4;
```

# if - else      else - if

```
int binsearch(int x, int v[], int n)
{
    int low, high, mid;
    low = 0;
    high = n - 1;
    while (low <= high)
    {
        mid = (low+high)/2;
        if (x < v[mid])
            high = mid - 1;
        else
            if (x > v[mid])
                low = mid + 1;
            else
                /* found match */
                return mid;
    }
    return -1; /* no match */
}
```

# Operador Ternário

- Seleção condicional entre duas expressões
- Se a expressão condicional for verdadeira, retorna o resultado da primeira expressão
- Senão, retorna o resultado da segunda expressão

# Operador Ternário

- Seleção condicional entre duas expressões
- Se a expressão condicional for verdadeira, retorna o resultado da primeira expressão
- Senão, retorna o resultado da segunda expressão

`exp-cond ? exp-sim : exp-não`

# Operador Ternário

```
/* exp-cond != 0 ? */
```

exp-cond ? exp-sim : exp-não

```
/* se != 0 retorna exp-sim */
```

```
/* se == 0 retorna exp-não */
```

# Operador Ternário

```
y = -1 ? 1+10 : -1-10;
```

```
printf("\ny=%d\n", y);
```

```
/* imprime: y=11 */
```

# switch - case

- Estrutura de decisão múltipla
- Condiciona a execução de blocos de comandos ao valor de uma expressão
- Define blocos para múltiplos valores da expressão
- Pode ainda ser definido um bloco de comandos *default*

# switch - case

```
#include <stdio.h>
main() /* count digits, white space, others */
{
    int c, i, nwhite, nother, ndigit[10]; nwhite = nother = 0;
    for (i = 0; i < 10; i++) ndigit[i] = 0;
    while ((c = getchar()) != EOF)
    {
        switch (c)
        {
            case '0': case '1': case '2': case '3': case '4':
            case '5': case '6': case '7': case '8': case '9':
                ndigit[c-'0']++;
                break;
            case ' ': case '\n': case '\t':
                nwhite++;
                break;
            default:
                nother++;
                break;
        }
    }
    printf("digits =");
    for (i = 0; i < 10; i++)
        printf(" %d", ndigit[i]);
    printf(", white space = %d, other = %d\n", nwhite, nother);
    return 0;
}
```



# **Estruturas de repetição**

# while

- Estrutura de repetição
- Executada enquanto expressão for diferente de 0
- O teste da expressão é executado antes do bloco
- Para evitar loop infinito as variáveis usadas na expressão precisam ser alteradas

```
while (expressao)
{
    comandos;
}
```

# while

```
int counter = 5;  
int factorial = 1;  
  
while (counter > 0)  
{  
    factorial *= counter--;  
}  
  
printf("factorial of 5 is  
%d\n", factorial);
```

# do-while

- Estrutura de repetição
- Executada enquanto expressão for diferente de 0
- O teste da expressão é executado **após** o bloco
- Para evitar loop infinito as variáveis usadas na expressão precisam ser alteradas

```
do
{
    comandos;
} while (expressão);
```

# do-while

```
int counter = 5;  
int factorial = 1;  
  
do  
{  
    factorial *= counter--;  
} while (counter > 0);  
  
printf("factorial of 5 is  
%d\n", factorial);
```

# for

- Estrutura de repetição
- Executada enquanto expressão condição for diferente de 0
- O teste da expressão é executado antes do bloco
- Inicializa + testa condição + executa bloco + incrementa

```
for (inicial; cond; increm)
{
    comandos;
}
```

# for

```
int count;  
int factorial = 1;  
  
for(count=5; count>0; count--)  
{  
    factorial *= count;  
}  
  
printf("factorial of 5 is  
%d\n", factorial);
```

# break - continue

- Interrompem o curso da repetição
- `break` pode ser usado no interior de laços `while`, `do-while` e `for` para força a saída
- `continue` pode ser usado no interior de laços `for` para saltar para o próximo valor do contador



# break

```
while ( i < 20 )
{
    i++;
    if ( i == 10 )
        break;
}
for (i=5; i<20; i++)
{
    if ( i == 10 )
        break;
}
```

# continue

```
for (i=5; i<20; i++)  
{  
    if ( i == 10)  
        continue;  
    printf("%d", i);  
}
```

# Exercício - Conversão de strings

- a) Escreva a função `escape` capaz de converter os códigos ascii dos caracteres { ' ', '\t', '\f', '\v', '\n', '\r' } na string `source` para as sequências '\X' (excluindo as aspas) correspondentes. O resultado da conversão deve ser armazenado na string `dest`. Use obrigatoriamente o comando `switch` ao invés de `if`.

```
escape(char source[], char dest[]);
```

- b) Faça da função `main()` um programa de teste que permita que você valide a função produzida no item anterior.

**Fim**