Unidade III - Fluxo de Controle

Disciplina Linguagens de Programação I
Bacharelado em Ciência da Computação da Uerj
Professores Guilherme Abelha e Gilson Costa

ANSI C

```c
#include  <stdio.h>
int main  ()
{
   printf("Hello, World!");
   return 0;
}
```

Que assuntos serão abordados nesta unidade?

- Estruturas básicas da linguagem C
  - comando
  - bloco
  - função
- Instruções de seleção de fluxo de execução
  - `if`
  - `switch`
  - `exp ? const : const`
- Instruções de repetição
  - `while`
  - `do-while`
  - `for`

Conceitos Fundamentais

Comandos e Blocos

; terminador de comando

{ } delimitadores de bloco

```
/* progPag039.c - print longest input line*/
int main()
{
   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

```
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
                return mid;  /* match found */
    }
    return -1; /* no match */
}
```

if – else

```
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
            return mid;  /* match found */
    }
    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

$$\|x\| = \begin{cases} x, & \text{se } x \geqslant 0 \\ -x, & \text{se } x < 0 \end{cases}$$

```
float x, modulo;
scanf("%f", &x);
modulo = x>=0 ? x : -x;
```

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

```c
#include <stdio.h>
int 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 **<u>após</u>** 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 U3.1 - Conversão de strings

Escreva a função `escape` capaz de converter os códigos ascii dos caracteres `{' ','\t', '\f','\v','\n','\r'}` quebrando-os em dois caracteres `'\\' + 'X'`.

OBS.:

Não é permitido o uso de `if`.

É obrigatório modularizar o código em três funções: uma para a leitura da sequência original, a segunda para a conversão e a terceira para fazer a impressão da string final

Fim