SCC0222 – Laboratório de ICC I

Erros Comuns

Luiz Eduardo
Rosane Minghim

Vitor Dellinocente
Barbara Cortes e Souza
Rafael Nakanishi
Marco Antonio

Variáveis

- Escolha nomes representativos, não muito grandes

```c
#include <stdio.h>

int main() {
    int a, b, i, j, v, p, d;
    float s, t, m;

}
```

```c
#include <stdio.h>

int main() {
    int altura, peso, i, j, velocidade, peso,distancia;
    float somaIdade, totalPessoas, mediaIdade;

}
```

Variáveis

- Declarar variáveis que não são utilizadas
  - Memória inútil
  - Clareza de código
- Uso incorreto de máscaras nas funções de I/O

```
#include <stdio.h>

int main() {
    int num1, num2;
    double soma;

    scanf("%d %d, &num1, &num2);
    soma = num1 + num2;

    printf("Soma: %f", soma);
}
```

Condicionais: IF

- Comparações no **IF** são individuais

```
if (a || b || c || d < 10) {
     ...
}
```

```
if (a < 10 || b < 10 || c < 10 || d < 10) {
     ...
}
```

Condicionais: IF

- Eficiência nas comparações

```
if (imc < 17)
    printf ("Muito abaixo do peso.");
if (imc >= 17 && imc < 18.5)
    printf ("Abaixo do peso.");
if (imc >= 18.5 && imc < 25) {
    printf ("Peso normal.");
```

```
if (imc < 17)
    printf ("Muito abaixo do peso.");
else if (imc >= 17 && imc < 18.5)
    printf ("Abaixo do peso.");
else if (imc >= 18.5 && imc < 25)
    printf ("Peso normal.");
```

Condicionais: SWITCH

- Cada ***case*** compara a variável com uma constante do mesmo tipo

```
char operator;
...
swicth (operator) {
    case +: ...
    case -: ...
```

```
char operator;
...
swicth (operator) {
    case '+': ...
    case '-': ...
```

Condicionais: SWITCH

- Instrução ***default***
  - Não compara valores
  - É uma boa prática colocar

```
char operator;
...
swicth (operator) {
    case ‘+’: ...; break;
    case ‘-’: ...; break;
    case ‘x’: ...; break;
    case ‘/’: ...; break;
    default: printf(“Operação inválida”);
}
```

Condicionais: SWITCH

- Dentro de um ***case***, não é preciso comparar novamente a variável

```
char operator;
...
swicth (operator) {
    case '+':
        if (operator == '+')
            ...;
        break;
    case '-': ...; break;
    case 'x': ...; break;
    case '/': ...; break;
    default: printf("Operação inválida");
}
```

Laços: FOR

- Identificar quando é melhor usar **WHILE**

```
for ( ; ; ) {
    scanf("%d", &num);
    if (num == -1)
        break;

    vet[num]++;
}
```

```
scanf("%d", &num);
while (num != -1) {
    vet[num]++;
    scanf("%d", &num);
}
```

Laços: WHILE

- Eficiência nas comparações

```
f = 0;
while (f != -1) {
    scanf("%d ", &f);
    if (f != -1)
        vet[f]++;

    if (f == -1)
        break;
}
```

```
scanf(“%d ”, &f);
while (f != -1) {
    vet[f]++;
    scanf("%d ", &f);
}
```

Declaração de arrays

```
int main() {
    int n;
    int vet[n];
    ...
}
```

```
int main() {
    int vet[100];
    ...
}
```

```
#define MAX_VET 100;

int main() {
    int vet1[MAX_VET];
    float numeros[MAX_VET];
    ...
}
```

```
// C99/C11

int main() {
    int n;
    scanf("%d", &n);
    int vet[n];
    ...
}
```

OUTROS

- Importar bibliotecas não utilizadas
- Colocar “**return 0**” no final da função “**main**”
- Não confundir “==“ com “=“

```
if (a = 2) {
    ...
}
```

```
if (a == 2) {
    ...
}
```

Documentação

- Um programa bem documentado não é aquele que possui muitos comentários no meio do código
- Em geral, devemos documentar todas as funções do código
  - Quando cabível, o arquivo como um todo que contém código
- O mais importante é dizer o que a função faz e não como ela faz
  - Entradas e Saídas
  - Descrição da relação entre Entrada e Saída

Documentação

- Documentando uma função

```
/* A função abaixo recebe um inteiro n >= 1 e um vetor v
e devolve o valor de um elemento máximo de v[0..n-1]. */
int max (int v[], int n) {
    int j, x = v[0];
    for (j = 1; j < n; j++)
        // x é um elemento máximo de v[0..j-1].
        if (x < v[j])
            x = v[j];

    return x;
}
```

Documentação

- Documentando o arquivo

```
/* Programa.c
   Este programa apenas escreve uma mensagem de
   boas-vindas na tela. Utilizado como exemplo em aula
   do curso de BCC-ICMC.

   Autor: Luiz Eduardo
   Data: 05/24/18
   Universidade de São Paulo - USP
*/
#include <stdio.h>

int main () {
    printf("Hello, World");
    return 0;
}
```

Layout do Código

- Programas precisam ser entendido por pessoas, não só pelo computador
- Um bom *layout* utiliza corretamente
  - Espaços entre palavras e símbolos
  - Identação

Layout do Código

- Um bom exemplo

```
int funcao (int n, int v[]) {
    int i, j;
    i = 0;
    while (i < n) {
        if (v[i] != 0)
            i = i + 1;
        else {
            for (j = i + 1; j < n; j++)
                v[j-1] = v[j];
            n = n - 1;
        }
    }
    return n;
}
```

Layout do Código

- Um mau exemplo (inconsistência no padrão adotado)

```
int funcao ( int n,int v[]){
    int i,j;
    i=0;
    while(i<n){
         if(v[i] !=0)
             i = i +1;
         else
         {
             for (j=i+1; j<n;j++)
                 v[j-1]=v[j];
             n=n- 1;
         }
    }
    return n;
}
```

Layout do Código

- Outro mau exemplo (identação)

```
int funcao (int n, int v[]) {
int i, j;
    i = 0;
  while (i < n) {
if (v[i] != 0)
i = i + 1;
    else {
    for (j = i + 1; j < n; j++)
    v[j-1] = v[j];
       n = n - 1;}
}
  return n;
}
```

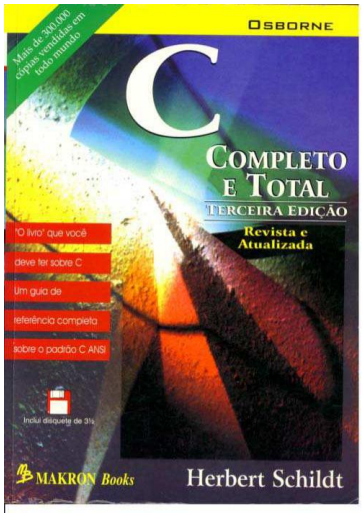
Código

- Pensando na internacionalização, é interessante que todo o programa seja codificado em Inglês
  - Nome das variáveis
  - Documentação

- Porém, nada impede que seja utilizado o Português
  - Manter a consistência (não misturar)

Sugestão de leitura

C – Completo e Total, 3ed.
Herbert Schildt.
Makron Books, 1996.



Algoritmos em linguagem C.
Paulo Feofiloff.
Elsevier, 2009.
https://www.ime.usp.br/~pf/algoritmos