BOAS PRÁTICAS DE PROGRAMAÇÃO*

Fernando Alva Manchego – estagiário PAE
e-mail: falva@icmc.usp.br

SCC0202 ALGORITMOS E ESTRUTURA DE DADOS I - 29/08/2011
*MATERIAL ELABORADO A PARTIR DO "GUIA DE CODIFICAÇÃO" – AUTORA: CAROLINA SCARTON

Boas Práticas de Programação

- Convenções/Estilo de Codificação/Programação
- Conjunto de diretrizes para escrever código fonte em uma linguagem de programação determinada.
  - Nome de variáveis
  - Nome de funções
  - Identação
  - Comentários
  - ...

Boas Práticas de Programação

- Definido pela equipe de programadores.
- Documentos oficiais dos criadores da linguagem ou do projeto:
  - Java: http://www.oracle.com/technetwork/java/codeconv-138413.html
    "Code Conventions for the Java Programming Language"
  - GNU: http://www.gnu.org/prep/standards/
    "GNU coding standards"

Padronização de Idiomas

- Definir um idioma → para TODO o programa!!!
- Comentários DEVEM seguir o idioma escolhido
- Opta-se por inglês para TADs clássicos → Pilha, Fila,...

Nomenclatura de Variáveis

- Notação Húngara: *Convenção de nomenclatura de identificadores, no qual o nome da variável ou função indica o seu tipo.*

"tNomeVariavel"

- t: tipo da variável

Tipo	Letra
int	i
float	f
double	d
char	c

Exemplos: iLinha, fSalario

Nomenclatura de Variáveis

- Modificadores: primeira letra (minúsculo) do modificador + primeira letra do tipo

Modificadores	Letra
unsigned	u
short	s
long	l

Exemplos: uiIdade, siNota

Nomenclatura de Variáveis

- Ponteiros: adicionar p (minúsculo) antes do tipo

puIdade

- Palavras que definem o nome → primeira letra maiúscula

puiTamanhoVetor

- Se tipo criado (typedef) → não aplicar modificadores
  - Somente p se for ponteiro: Matrix* pMatrix

Nomenclatura de Variáveis

- Outros pontos:
  - Nomes devem ser significativos!!!!
    - Evitar: i, aux,...
  - Não utilizar caracteres especiais, acentos ou cedilhas
    - Evitar problemas com editores de texto

Nomenclatura de Funções

- Modelo:

  NomeBiblioteca_NomeFuncao()

  - NomeBiblioteca: nome do arquivo .h
    - Auxiliar o programador (Eclipse, CodeBlocks) → auto-completar!
  - NomeFuncao: nome da função
  - Exemplos: Conjunto.h → Conjunto_CriaConjunto

Nomenclatura de Constantes

- Todo em maiúsculo → pode abreviar:

  #define TAM 100

- Constantes de erro → ERRO_NOME

  #define ERRO_PONTEIRO_NULO 1

Comentários

- Explicar trechos pouco claros
- Explicar trechos com alta complexidade
- Manter o idioma definido!

Cabeçalhos de funções

- Cada função da biblioteca deve ter um cabeçalho
- Reproduzir os cabeçalhos no .h e no .c
- Deve conter:
  - Nome da função
  - Descrição
  - Parâmetros, descrição e função (entrada/saída)
  - Retorno
  - ...
- **Mais detalhes: próxima aula!!**

Construtor/Destrutor

- Construtor: inicializar as variáveis da biblioteca (alocação de memória)
  - Facilita desalocação → cada construtor deve ter um destrutor (responsável por desalocar a memória)
- Os nomes podem ser adaptados:
  - Matriz_CriaMatrizVazia
  - Matriz_LiberaMatriz
- Descrição no cabeçalho da função → informar se é construtor ou destrutor

Biblioteca de Erros

- Erros recorrentes:
  - Sucesso
  - Ponteiro nulo
  - Espaço de memória insuficiente
- Criar um .h geral
  - Facilita o entendimento do código
  - Disponibilizando o .h para o usuário → parte da documentação

Biblioteca de Erros

```
#ifndef BIBLIOERRO_H_INCLUDED
#define BIBLIOERRO_H_INCLUDED

/*Biblioteca que define os tipos de erros possíveis*/
/*Disponível ao usuário - parte da documentação*/

#define ERRO_SUCESSO 0
#define ERRO_PONTEIRO_NULO 1
#define ERRO_ENDERECO_INVALIDO 2
#define ERRO_MEMORIA_INSUFICIENTE 3


#endif // BIBLIOERRO_H_INCLUDED
```

Ponteiros

- Todo ponteiro deve ser verificado antes de ser acessado
  - Exceto no construtor → verificar sucesso na alocação

Identação

- Identar com espaços (não TAB)
- Quatro espaços em branco
- Evitar muitos níveis de identação
  - No máximo 5, contando o nível inicial da função
  - Procurar quebrar em rotinas

Referências

- Manual de Codificação – elaborado por Carolina Scarton (São Carlos, 2011)

Exercício

- Escreva um programa que receba da linha de comando o dia, mês e ano correntes, e imprima a data em formato apropriado. Utilizar os argumentos da função *main* e seguir o manual de codificação.
- Entrada: Date 29 08 2011
- Saída: 29 de Agosto de 2011

Argumentos na linha de comando

- Passar informação ao programa desde a linha de comando.
- Exemplo:

gcc -o myprog myprog.c

- No exercício:

Date 29 08 2011

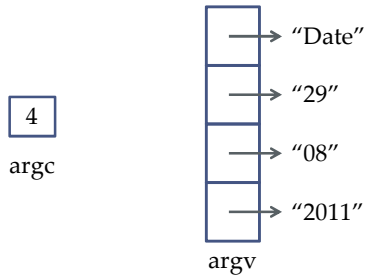
Argumentos do *main*

int main(int argc, char *argv[]) { ... }

Número de argumentos

Valores dos argumentos

- No exercício:




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

int main(int argc, char *argv[]){

    int iMes;
    char * arrNomesMeses [] = {"Janeiro", "Fevereiro", "Março", "Abril", "Maio",
                        "Junho", "Julho", "Agosto", "Setembro", "Outubro",
                        "Novembro", "Dezembro"};

    /* Testa se o numero de parametros fornecidos estah correto*/
    if(argc == 4) {
        iMes = atoi(argv[2]);  /* argv contem strings. A string referente ao mes deve ser
                                  transformada em um numero inteiro.*/

        if (iMes<1 || iMes>12)  /* Testa se o mes eh valido */
            printf("Erro!\nUso: data dia mes ano, todos inteiros");
        else
          printf("\n%s de %s de %s", argv[1], arrNomesMeses[iMes-1], argv[3]);
    }

    else printf("Erro!\nUso: data dia mes ano, todos inteiros");

    return 0;
}
```