Curso de C/C++ Avançado

Aula 1 – História da Linguagem C, Conceitos básicos de C, Strings, Matrizes e Ponteiros

Allan Lima – http://allanlima.wordpress.com

creative commons
COMMONS DEED

SOME RIGHTS RESERVED

- Você pode:
  - copiar, distribuir, exibir e executar a obra
  - criar obras derivadas
  - fazer uso comercial da obra
- Sob as seguintes condições:
  - **Atribuição**. Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.
  - **Compartilhamento pela mesma Licença**. Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.
  - Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra.
  - Qualquer uma destas condições podem ser renunciadas, desde que Você obtenha permissão do autor.
- **Veja aqui a licença completa**

Apresentação do Curso

- Módulo C
  - 4 aulas
- Módulo C++
  - 4 aulas
- Extensões
  - 2 aulas:
    - *Sockets*
    - *Threads*

Metodologia das aulas

- Exercícios no final de cada aula
- Análise de exemplos de código
- Uma prova ao final da décima aula

História de C

- Surgiu no início dos anos 70
- Criada inicialmente para o UNIX
- Criadores:
  - Dennis Ritchie (dir)
  - Kenneth Thompson (esq)
- Baseada na Linguagem B
- Versão inicial bastante simples

www.viphostsystem.com

História da Linguagem C

- Ampla popularização nos anos 80
- Muitas arquiteturas e compiladores
- Problemas com a incompatibilidade
- Padronização de 82 a 89 (C ANSI)
- Até hoje existem problemas entre os diversos compiladores e sistemas operacionais

Características

- Paradigma Procedural
- Flexível
- Alta performance
- Poucas restrições
- Ótima iteração com:
  - Sistemas Operacionais
  - Dispositivos de Hardware
  - Outras Linguagens

Tipos

- C possui 5 tipos básicos:
  - char, int, float, double e void
- E 4 modificadores básicos:
  - signed, unsigned, long e short
  - Os 4 podem ser aplicados ao int
  - long pode ser aplicado ao double
  - signed e unsigned aplicados ao char

Tipos

- Modificadores de acesso:
  - const: a variável não pode ter o seu valor alterado
  - volatile: a variável pode ter o seu valor modificado fora do controle do programa
- Classes de Armazenamento:
  - auto: indica que uma variável é local (opcional), também é usada na declaração de *nested functions*
  - extern: variável declarada em outro arquivo
  - register: armazena, se possível, a variável em um registrador na própria CPU.

Tipos

- Classes de Armazenamento (Cont.):
  - static: não permite que um módulo externo possa alterar nem ver uma variável global, também é usado para manter o valor de uma variável local em uma função de uma chamada para outra.

- Exemplos:
  - *exemploRegister.c*
  - *exemploStatic.c*
  - *exemploAuto.c*

Tipos

- O tamanho do inteiro depende da arquitetura do sistema:
  - Sistemas de 32 bits -> inteiro de 32 bits
  - Sistemas de 64 bits -> inteiro de 64 bits
- Restrições:
  - short int e int devem ter pelo menos 16 bits
  - long int com no mínimo 32 bits
  - short int <= int <= long int

Variáveis

- Restrições
  - O nome das variáveis devem começar com uma letra ou um sublinhado “_”
  - O demais caracteres podem ser letras, números ou sublinhado
  - O nome da variável não pode ser igual a uma palavra reservada e aos nomes das funções
  - Tamanho máximo para o nome de uma variável:
    - 32 caracteres

Constantes

- São valores que são mantidos fixos pelo compilador
- Também podem ser:
  - Octais - 0NUMERO_OCTAL
  - Hexadecimais - 0xNUMERO_HEXADECIMAL
- Exemplos:
  - ‘\n’, “C++”, 10, 15.0, 0xEF (239 em decimal), 03212 (1674 em decimal)

Constantes de Barra Invertida

Código	Significado
\b	Retrocesso (backspace)
\f	Alimentação de Formulário (form feed)
\t	Tabulação Horizontal (tab)
\n	Nova Linha
\”	Aspas
\’	Apostrofo
\0	Nulo
\\	Barra Invertida
\a	Sinal Sonoro (Beep)
\N	Constante Octal (N é o valor da constante)
\xN	Constante Hexadecimal (N é o valor da constante)

Operadores Aritméticos

Operador	Ação
+	Soma
-	Subtração ou troca de sinal
*	Multiplicação
/	Divisão
%	Resto da divisão inteira
++	Incremento
--	Decremento

Operadores Relacionais

Operador	Relação
>	Maior que
>=	Maior que ou igual a
<	Menor que
<=	Menor que ou igual a
==	Igual a
!=	Diferente de

Operadores Lógicos

Operador	Função
&&	AND
||	OR
!	NOT

Operadores Bit a Bit

Operador	Ação
&	AND Lógico
|	OR Lógico
^	XOR (OR exclusivo)
~	NOT
>>	Shift Rigth
<<	Shift Left

Execício

1)Qual o valor das variáveis após a execução do seguinte trecho de código:

```
int x = 2, y = 4;
int a, b, c, d, e, f;
a = x & y;
b = x | y;
c = x ^ y;
d = ~x;
e = x << 3;
f  = x >> 1;
```

Resultado

a = 0

b = 6

c = 6

d = -3

e = 16

f = 1

Controladores de Fluxo

- C possui 7 controladores de fluxo básicos:
  - if
  - ?
  - switch
  - for
  - while
  - do-while
  - goto

Vetores

- São Matrizes com uma única dimensão
- Declaração
  - *tipo nome[tamanho];*
- Exemplo:

```
int array1[10];
```

- Podem ser inicializados na sua declaração:

```
int array2[10] = { 1, 2, 3 };
int array3[] = { 1, 2, 3 }
```

Vetores

- Cuidados importantes ao utilizar vetores
  - O compilador não verifica se o índice é válido:

```
int a[10];
int b = a[32];
int b = a[-32];
int c = a['F']; // converte f para int
```

  - O código acima compila sem problemas, mas o que irá acontecer quando ele for rodado?
  - Ninguém sabe!!!

Exemplo

```
#include <stdio.h>

int main() {
    int numeros[10] = {1, 1};
    int i;
    for (i = 2; i < 10; i++) {
        numeros[i] = numeros[i-1] + numeros[i-2];
    }
    for (i = 0; i < 10; i++) {
        printf("%d\n", numeros[i]);
    }
    return 0;
}
```

Strings

- Uma string é vetor de caracteres terminado pelo caractere nulo ‘\0’
- Sintaxe:
  - *char nomeDaString[] = “conteudo”;*
- Exemplos:

```
char frase[] = “Eu adoro C”;
char centro[4] = “CIn”;
char faculdade[10] = {‘U’, ‘F’, ‘P’, ‘E’, ‘\0’};
```

Cuidados Importantes

- Lembre-se sempre do ‘\0’
  - O tamanho do vetor deve ser o número de caracteres que ela irá possuir + 1
  - Erro comum:
    - char faculdade[] = {'U', 'F', 'P', 'E'};
    - faculdade não é uma string!

A Biblioteca string.h

- Principais funções:
  - int strlen(str)
  - int strcmp(str1, str2)
  - string strcpy(destino, origem)
  - string strcat(destino, origem)

strlen

- int strlen(str);
- Retorna o tamanho da string passada como parâmetro
- Exemplo:

```
char string[] = “Linguagem”;
printf(“%d”, strlen(string)); // imprime 9
```

Obs.: O caractere ‘\0’ não é contado.

strcmp

- int strcmp(str1, str2)
- Compara duas strings

Condição	Retorno
<0	Se str1 é menor que str2
0	Se str1 é igual à str2
>0	Se str1 é maior que str2

- A ordem lexicográfica é utilizada para a comparação

Exemplo

```
#include <stdio.h>

int main() {
    char pergunta[] = “qual é a sua linguagem favorita?”;
    char resposta[15];

    do {
        printf(“%s”, pergunta);
        scanf(“%s”, resposta);
    } while (strcmp(resposta, “C”));
    return 0;
}
```

strcpy

- string strcpy(destino, origem)
- Copia o segundo parâmetro no primeiro
- Exemplo:

```
char ori[] = “ABC”;
char dest[12] = “”; // Não esqueça de inicializar

strcpy(dest, ori); // dest = “ABC”
```

strcat

- string strcat(destino, origem)
- Concatena o primeiro parâmetro no segundo
- Exemplo:

```
char ori[] = “DEF”;
char dest[12] = “ABC”;

strcat(dest, ori); // dest = “ABCDEF”
```

Obtendo mais detalhes

- www.cplusplus.com

Outras Funções

- string gets(string)
  - Lê uma string do dispositivo de entrada padrão
  - Também lê espaços
  - Não verifica o tamanho máximo da string
- int puts(string)
  - Imprime uma string no dispositivo de saída padrão
- Ambas estão definidas no header stdio.h

Matrizes

- Sintaxe:
  - *tipo nomoDaMatriz[dim1][dim2]...[dimN];*
- Exemplos:

```
// Matriz Bidimensional
int m1[2][2] = { 1, 2, 3, 4 };
// Outra maneira
int m2[2][2] = { {1, 2 }, { 3, 4 } };
// Matriz Tridimensional
int m3[2][2][2] = { 1, 2, 3, 4, 5, 6, 7, 8 };

printf("%d\n", m1[1][1]);
printf("%d\n", m3[1][0][0]);
```

Ponteiros

- Guardam o endereço de uma variável
- Sintaxe:
  - *tipo *nomeDoPonteiro;*
- Exemplos:

```
int *a;
char  *b;
float *c;
double *d;
```

Ponteiros

- Ponteiros precisam ser inicializados
- Para isto basta usarmos o operador &
- Ele retorna o endereço de uma variável
- Exemplo:

```
int a = 10, *p;
   p = &a; // p aponta para o endereço de a
```

Ponteiros

- O operador * é usado para termos acesso ao valor do endereço de memória para o qual um ponteiro aponta
- Exemplo:

```
int a, b = 10, *p;
p = &b; // p aponta para o endereço de b
a = *p; // a = 10
*p  = 15; // b = 15, mas a ainda é 10
```

Ponteiros

- Operações aritméticas:
  - Igualdade

```
// p1 aponta para o mesmo endereço de p2
p1 = p2
*p1 = *p2 // copia o conteúdo
```

  - Incremento e decremento

```
p++; // p = p + sizeof(tipo_de_p)
p--; // p = p - sizeof(tipo_de_p)
```

Ponteiros

- Operações aritméticas:
  - incremento
    - p = p + 10;
    - *p = *p + 10;
  - Comparação
    - p1 == p2;
    - p1 > p2;
    - p1 >= p2;

Ponteiros

- Exemplo:
  - *exemploPonteiro.c*

Exercícios

2)Qual a diferença entre:

p++; (*p)++; *(p++);

Exercícios

3)Qual o valores de x e de y no final do programa?

```
int main()  {
    int y, *p, x;
    y = 0;
    p = &y;
    x = *p;
    x = 4;
    (*p)++;
    x--;
    (*p) += x;
    printf ("y = %d\n", y);
    return 0;
}
```

Exercícios

4) Escreva um programa que lê strings do teclado até que duas strings iguais sejam digitadas consecutivamente. A saída é a concatenação de todas as strings lidas e os tamanhos da maior a da menor.

Exercícios

5)Faça um programa que calcula o determinante de uma matriz 2*2. O usuário entra com a matriz e o programa imprime o determinante na tela

Exercícios

6) Implemente uma função chamada swap que recebe os dois ponteiros para inteiros como parâmetro inverte seus conteúdos.

Referências

- Matos, P. A. & Carvalho, G. H. P. - A Linguagem de Programação C
- The C Programming Language
  - http://www.engin.umd.umich.edu/CIS/course.des/cis400/c/c.html
- Curso de C da UFMG
  - http://ead1.eee.ufmg.br/cursos/C/
- Lammert Bies, ASCII character map
  - http://www.lammertbies.nl/comm/info/ascii-characters.html

Referências

- Curso de C da UFMG
  - http://ead1.eee.ufmg.br/cursos/C/
- cplusplus resources
  - http://www.cplusplus.com
- Slides de Gustavo (ghpc@cin.ufpe.br) do Curso de C/C++