Curso de C/C++ Avançado

Aula 2 – Ponteiros, Alocação Dinâmica e Diretivas de Compilação

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

- *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***

Vetores e Ponteiros

- *Quando declaramos uma matriz:*
  - *tipo meuArray[tam];*
  - *O compilador aloca a matriz em uma região de memória contínua*
  - *meuArray na verdade é um ponteiro para a primeira posição da memória que foi alocada.*
  - *meuArray[10] é equivalente a *(meuArray + 10)*

Vetores e Ponteiros

- *Podemos usar ponteiros como se fossem matrizes:*

```
int mat[] = { 1, 2 , 3 };
int *p = mat;
printf("%d", p[2]); // imprime 3
```

Exemplo

```
#include <stdio.h>
int main() {
    int mat[50][50];
    int *p= (int *) mat, i;
    // percorre a matriz com um único loop !!!
    for (i = 0; i < 2500; i++) {
        *p = 0;
        p++;
    }
    return 0;
}
```

É muito mais rápido e diminui o tamanho do código

Vetores e Ponteiros

- *Matrizes **não** podem ser usadas como ponteiros*
  - *Exemplo:*

```
int vetor[10];
int *ponteiro, i;
ponteiro = &i;
// as operações a seguir são inválidas
vetor = vetor + 2; // ERRADO: vetor não é variável
*vetor = 0; // ERRADO: vetor não é variável
vetor = ponteiro; // ERRADO: vetor não é variável
```

  - *tipo meuArray[]; ⇔ const tipo * meuArray;*

Strings e Ponteiros

- *Strings são variáveis do tipo *char*
- *Exemplo:*

```
#include <stdio.h>
#include <string.h>
int main() {
    char curso[15];
    char *p = "Curso de C/C++";

    // p pode ser usado como uma string
    strcpy(curso, p);
    printf("%s %s\n", curso, p);

    return 0;
}
```

Ponteiro do tipo void

- *Este tipo de ponteiro pode apontar para qualquer outro tipo*
- *Mas para se ter acesso ao conteúdo por ele endereçado precisamos fazer um cast*
- *É útil para a passagem de parâmetros genéricos*

Exemplo

```
#include <stdio.h>

int main() {
        char *s = "string";
        int a = 1, *p = &a;
        float f = 2.0;
        void *v;

        v = p; // v aponta para um inteiro
        printf("%d\n", *((int *) v));
        v = s; // v aponta para uma string
        printf("%s\n", (char *) v);
        v = &f; // v aponta para um float
        printf("%f\n", *((float *) v));

        return 0;
}
```

Ponteiros para Ponteiros

- *Guardam o endereço de outro ponteiro*
- *Sintaxe:*
  - *tipo **nomeDaVariavel;*
- *Também podem ser usados como matrizes bidimensionais*
- *Podemos criar ponteiros para ponteiros para ponteiros para ponteiros, e assim por diante*

Exemplo

```
#include <stdio.h>

int main() {
    int a = 10, *p, **pp;

    p = &a;
    pp = &p;

    printf("%d\n", **pp); // imprime 10

    return 0;
}
```

Obs.: Para acessar o valor de a através de pp aplicamos o operador * duas vezes

Alocação Dinâmica

- *C permite que o programador aloque memória em tempo de execução.*
- *Para isto existe um conjunto de funções*
- *void *malloc(unsigned int numBytes);*
  - *Usada para alocar memória*
  - *numBytes = o número de bytes a serem alocados*
  - *Retorna um ponteiro para o início do bloco de memória alocado*

Alocação Dinâmica

- *void *calloc(unsigned int numBytes,*
  unsigned int qtd);
  - *Aloca numBytes * qtd bytes*
  - *Retorna um ponteiro para o início do bloco de memória alocado*
- *void *realloc(void *ptr, unsigned int tam);*
  - *Modifica o tamanho ocupado na memória por ptr para tam*
  - *Retorna um ponteiro para o início do bloco de memória que foi realocado*

Alocação Dinâmica

- *void memset(void *p, int c,*
  unsigned int tam);
  - *Inicializa a região de memória endereçada por p com valor*
  - *p deve estar alocado*
- *void free(void *p);*
  - *Libera a memória ocupada por p*

Exemplos

- *exemploAlocacaoDinamica1.c*
- *exemploAlocacaoDinamica2.c*

Protótipos de funções

- *Os compiladores de C são muito eficientes, mas isto traz alguns problemas*
- *Funções só podem ser usadas se forem definidas anteriormente*
- *A solução é utilizar protótipos de funções*
- *Sintaxe:*
  - *tipoDeRetorno nomeDaFunção(tipo1, ... , tipoN);*

Exemplo

```
void b(int); // protótipo de b

void a(int i) {
    // ...
    b(i);
    // ...
}

void b(int i) {
    // ...
    a(i);
    // ...
}
```

Ponteiros para Funções

- *Sintaxe:*
  - *tipoDeRetorno (*nomeDoPonteiro) (tipoP1, ... , tipoPN)*
- *Exemplo:*

```
int (*p)(const char *);
p = puts; // inicializa
p (“UFPE”); // faz a chamada
(*p) (“UFPE”); // faz a chamada de outra maneiraz
```

- *Também podemos passar ponteiros para funções como parâmetro de outras funções*

Exemplo

- *exemploPonteiroFuncao.c*

A função main

- *Principal função do programa*
- *Sintaxe completa:*
  - *int main(int argc, char *argv[]);*
  - *Permite que o usuário passe argumentos para o programa*
  - *argc é quantidade de parâmetros*
  - *argv contém os parâmetros*
  - *Obs.: argv[0] contém o nome do programa por isso argc > 0*

Exemplo

```
#include <stdio.h>

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

    for (i = 0; i < argc; i++) {
     printf("%s\n", argv[i]);
    }

    printf("total: %d\n", argc);

    return 0;
}
```

Diretivas de Compilação

- *São comandos que não são compilados, mas sim interpretados pelo **pré-processador***
- *Todas as diretivas começam por # e podem ser utilizadas em qualquer parte do código*
- *Principais diretivas definidas no C ANSI:*

#if, #ifdef, #ifndef, #else, #elif, #endif,
#include, #define, #undef, #error

#include

- *Diz ao compilador para incluir um arquivo especificado no seu programa*
- *Sintaxe:*
  - *#include <nomeDoArquivo>*
    - O arquivo se encontra no path do compilador
  - *#include "nomeDoArquivo"*
    - O arquivo se encontra em outro local
- *Exemplos:*
  - *#include<string.h>*
  - *#include "pessoa.h"*

#define

- *Define uma macro no compilador*
- *Sintaxe:*
  - *#define nomeDaMacro corpoDaMacro*
  - *corpoDaMacro é opcional*
- *Exemplos:*
  - *#define TAMANHO 10*
  - *#define _WIN32*
  - *#define max(a,b) ((a>b) ? (a) : (b))*

#undef

- *Apaga uma macro do compilador*
- *Sintaxe:*
  - *#undef nomeDaMacro*
- *Exemplo:*
  - *#undef max*

#ifdef e #endif

- *Permite a compilação condicional*
- *Sintaxe:*

```
#ifdef nomeDaMacro
    corpoDaMacro // código
#endif
```

- *Se existir uma macro chamada nomeDaMacro no compilador corpoDaMacro será compilado*
- *#endf é usada para indicar o fim da macro*

A Diretiva #ifndef

- *Sintaxe:*

```
#ifndef nomeDaMacro
    corpoDaMacro // código
#endif
```

- *Se **não** existir uma macro chamada nomeDaMacro no compilador corpoDaMacro será compilado*

#if, #else e #elif

- *Sintaxe:*

```
#if expressãoConstante1
    declarações1
#elif expressãoConstante2
    declarações2
...
#elif expressãoConstanteN
    declaraçõesN
#else
    declarações
#endif
```

#error

- *Provoca um erro de compilação*
- *Sintaxe:*
  - *#error mensagem*

Exemplo

- *exemploDiretivas.c*
- *exemploError.c*

Exercícios

1)Crie as seguintes macros:

- *min(a,b)*
  - Retorna o mínimo entra a e b
- *isPar(a)*
  - Retorna 1 se a for par e 0 caso contrário

Exercícios

2)Implemente as seguintes funções:

- *int mystrlen(char *c)*
  - retorna o tamanho da string *c*
- *void mystrcpy(char *dest, char *orig)*
  - copia *orig* para *dest*
- *void mystrcat(char *dest, char *orig)*
  - concatena orig em dest
- *int mystrchar(char *str, char c)*
  - Retorna o índice onde *c* ocorre em *str*

Exercícios

3)Declare um ponteiro para uma das funções que você fez no exercício anterior e chame-a utilizando o mesmo

Referências

- *Matos, P. A. & Carvalho, G. H. P. - A Linguagem de Programação C*
- *Curso de C da UFMG*
  - *http://ead1.eee.ufmg.br/cursos/C/*
- *Algumas Notas sobre Programação em C*
  - *http://paginas.fe.up.pt/~apm/C_tut/Cap_7.htm*
- *Standart C*
  - *http://www.ccs.ucsd.edu/c/*
- *Slides de Gustavo (ghcp@cin.ufpe.br) do Curso C/C++*