Programação de Computadores II

Ponteiros e Alocação Dinâmica

Gil Eduardo de Andrade

Ponteiros

Introdução

- O ponteiro é uma variável que contém um endereço de memória, normalmente relativo a localização de uma outra variável em memória;
- Sendo assim chamamos uma variável de ponteiro porque esta aponta para o endereço de memória onde encontra-se outra variável;

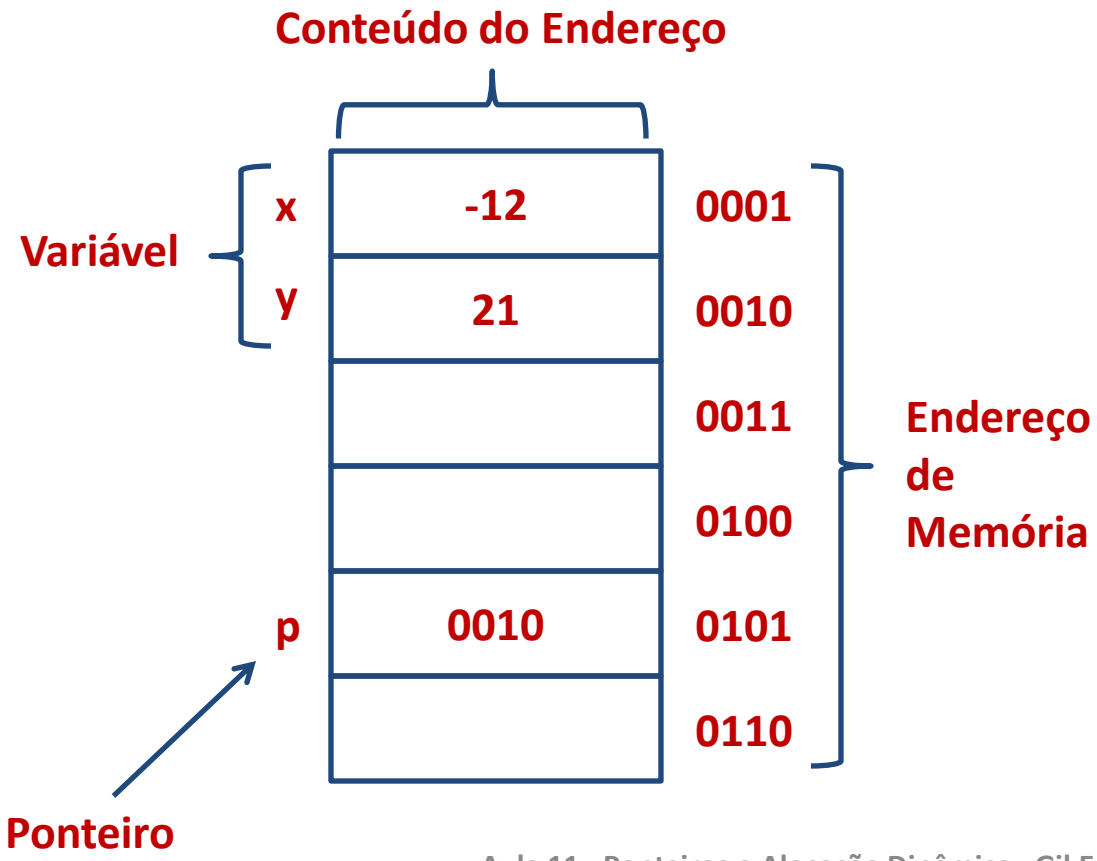
Ponteiros

Variável: Padrão x Ponteiro

- Uma variável padrão faz referência direta para um valor armazenado em memória;
- Uma variável ponteiro contém o endereço de memória de outra variável que faz referência direta ou não para um valor armazenado em memória;

Ponteiros

Variável: Padrão x Ponteiro



O *ponteiro p* armazenada o *endereço de memória* da variável *y (0010)* que possui como conteúdo o valor 21. Observe que o *ponteiro p não armazena o valor 21* mas sim o endereço de memória da variável y.

Ponteiros

Utilização de Ponteiro

- Permite a utilização de alocação dinâmica, onde é posição determinar o tamanho de um vetor ou matriz durante a execução do programa;
- Possibilita que funções modifiquem o conteúdo de seus argumentos;
- Aumenta a eficiência de determinadas rotinas – ficam mais rápidas;

Ponteiros

Codificação: declaração

- A declaração de ponteiros em C é efetuada da seguinte maneira:
  - ***tipo *nome_ponteiro;***
    - ***tipo:*** indica qual tipo de variável será apontada pelo ponteiro declarado – ex.: ***int***, ***float***, ***double***;
    - *: indica que a variável declarada se trata de um ponteiro;
    - ***nome_ponteiro:*** indica o nome pelo qual o ponteiro é referenciado;
    - ***Exemplo:***
      - ***int *p_int;***
      - ***char *p_char;***

Ponteiros

Operadores

- Existem dois operadores especiais para ponteiros:
  - ***"&"***: retorna o endereço de memória;
    - ***Ex.:*** int *p = &a; *(**p** recebe o endereço de memória de **a**)*
  - ***"*"***: retorna o conteúdo (valor) da variável;
    - ***Ex.:*** int b = *p; *(**b** recebe o conteúdo do endereço de memória apontado por **p**)*

Ponteiros

Exemplos de Codificação (declaração e operadores):

```c
#include <stdio.h>

int main() {

        // declara duas variáveis inteiras
        int a, b;
        // declara um ponteiro para inteiro
        int *p;

        // faz a variável "a" receber o valor '12'
        a = 12;
        // faz o ponteiro "p" receber o endereço de memória
        // da variável "a" - "p" aponta para "a"
        p = &a;
        // faz a variável "b" receber o conteúdo do endereço
        // de memória apontado pelo ponteiro "p"
        b = *p;
        // Aprenseta os valores armazeados em "b" e "a"
        printf("\na=%i / b=%i", a, b);

        printf("\n");
        return 0;
}
```

Ponteiros

Exemplos de Codificação (declaração e operadores):

```
g1l3du4rd0@asus-ultrabook-g1l:/run/media/g1l3du4rd0/GEA/VirtualBox/Compartilhada/IFPR/2015/PC/aula11/exemplos/pc_...
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ gcc ponteiro_operadores.c
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ ./a.out

a=12 / b=12
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$
```

Ponteiros

Exemplos de Codificação (impressão e operadores):

```c
#include <stdio.h>

int main() {

        // declara uma variável inteira
        int a;
        // declara um ponteiro para inteiro
        int *p;

        // faz a variável "a" receber o valor '12'
        a = 12;
        // faz o ponteiro "p" receber o endereço de memória
        // da variável "a" - "p" aponta para "a"
        p = &a;
        // Apresenta o conteúdo de "p" (endereço apontado por "p")
        printf("\nConteúdo de p = %p", p);
        // Apresenta o conteúdo do endereço apontado por "p"
        printf("\nConteúdo do endereço apontado por p = %i", *p);

        printf("\n");
        return 0;
}
```

Ponteiros

Exemplos de Codificação (impressão e operadores):

```
g1l3du4rd0@asus-ultrabook-g1l:/run/media/g1l3du4rd0/GEA/VirtualBox/Compartilhada/IFPR/2015/PC/aula11/exemplos/pc_...
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ gcc ponteiro_impressao.c
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ ./a.out

Conteúdo de p = 0xbfc461d8
Conteúdo do endereço apontado por p = 12
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$
```

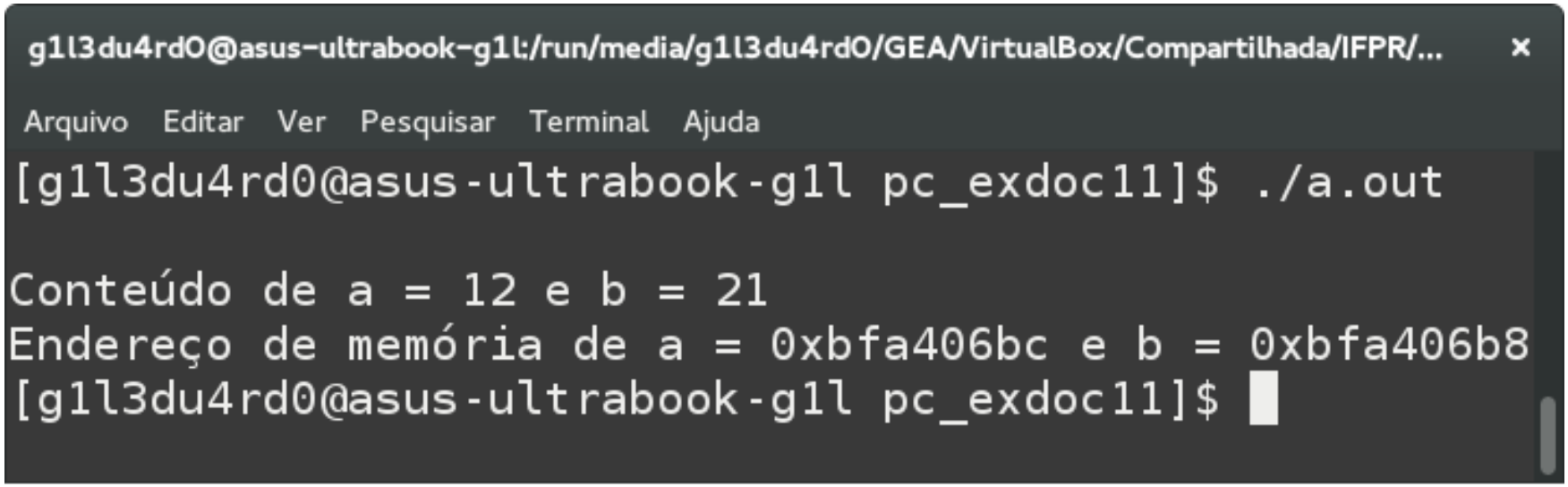
Ponteiros

Exemplos de Codificação (endereço de memória):

```c
#include <stdio.h>

int main() {

        // declara duas variáveis inteiras
        int a, b;

        // faz a variável "a" receber o valor '12'
        a = 12;
        // faz a variável "b" receber o valor '21'
        b = 21;
        // Apresenta o conteúdo de "a" e "b"
        printf("\nConteúdo de a = %i e b = %i", a, b);
        // Apresenta o endereço de memória de "a" e "b"
        printf("\nEndereço de memória de a = %p e b = %p", &a, &b);

        printf("\n");
        return 0;
}
```

Ponteiros

Exemplos de Codificação (endereço de memória):



Alocação Dinâmica

Introdução

- Alocação dinâmica é o meio pelo qual um programa pode obter memória durante a sua execução;
- Variáveis locais e globais não podem obter mais memória durante o tempo de execução;
- Portanto para obter maior quantidade de memória em tempo de execução utilizamos ponteiros e alocação dinâmica de memória;

Alocação Dinâmica

Principais Funções

- A linguagem C possui diversas funções de alocação dinâmica, contudo suas principais são a função ***malloc()*** e a função ***free()***;
  - ***malloc()*:** aloca a quantidade de memória necessária;
  - ***free()*:** libera a quantidade de memória alocada;

Alocação Dinâmica

Função *malloc()*

- ***void**** malloc(***size_t** * **numero_bytes***);
  - ***void****: retorna um ponteiro para qualquer tipo, ou seja, pode ser utilizada para qualquer tipo de ponteiro (int *; char *);
  - ***size_t***: definido na biblioteca ***<stdlib.h>***, uma espécie de ponteiro sem sinal;
  - ***mumero_bytes***: total de bytes que se deseja alocar;

Alocação Dinâmica

Exemplo: função *malloc()*

- ***int* p*** = malloc(***10*** * ***sizeof(int)***);
- Aloca espaço de memória para 10 valores inteiros, o ***sizeof(int)*** garante que seja alocado 10 vezes o tamanho de um inteiro na memória;
- Observe que cada tipo possui um tamanho específico, por exemplo, um ***double*** utiliza um espaço de memória maior que um ***int***;

Alocação Dinâmica

Exemplo de Codificação (alocação dinâmica - *int*)

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

int main() {

        // declara duas variáveis inteiras
        int x;
        int tamanho = 5;
        // declara um ponteiro para inteiro
        int *p;

        // aloca dinamicamente o ponteiro "p" com tamanho 5
        p = malloc(tamanho*sizeof(int));
        // faz ponteiro "p" receber valores nas posições alocadas
        // apresenta os valores armazenados nas posições
        for(x=0; x<tamanho; x++) {
                p[x] = x;
                printf("\np[%i] = %i", x, p[x]);
        }

        // libera o espaço de memória alocado
        free(p);

        printf("\n");
        return 0;
}
```

Observe que a biblioteca *<stdlib.h>* foi incluída para que seja possível utilizar a função e *sizeof()* e trabalhar com alocação dinâmica

Alocação Dinâmica

Exemplo de Codificação (alocação dinâmica - *int*)

```
g1l3du4rd0@asus-ultrabook-g1l:/run/media/g1l3du4rd0/GEA/VirtualBox/Compartilhada/IFPR/2015/PC/aula11/exemplos/pc_...
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ gcc alocacao_malloc.c
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ ./a.out

p[0] = 0
p[1] = 1
p[2] = 2
p[3] = 3
p[4] = 4
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$
```

Alocação Dinâmica

Exemplo de Codificação (alocação dinâmica - *char*)

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

int main() {

        // declara duas variáveis inteiras
        int x;
        int tamanho = 5;
        // declara um ponteiro para caracteres
        char *p;

        // aloca dinamicamente o ponteiro "p" com tamanho 5
        p = malloc(tamanho*sizeof(char));
        // faz ponteiro "p" receber caraceteres nas posições alocadas
        // apresenta os caracteres armazenados nas posições
        for(x=0; x<tamanho; x++) {
                p[x] = 65 + x;
                printf("\np[%i] = %c", x, p[x]);
        }

        // libera o espaço de memória alocado
        free(p);

        printf("\n");
        return 0;
}
```

Alocação Dinâmica

Exemplo de Codificação (alocação dinâmica - *char*)

```
g1l3du4rd0@asus-ultrabook-g1l:/run/media/g1l3du4rd0/GEA/VirtualBox/Compartilhada/IFPR/2015/PC/aula11/exemplos/pc_ex...
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ gcc alocacao_malloc_char.c
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ ./a.out

p[0] = A
p[1] = B
p[2] = C
p[3] = D
p[4] = E
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$
```

Realocação Dinâmica

Introdução

- Realocação dinâmica é o meio pelo qual um programa pode obter memória, mais de uma vez para um mesmo ponteiro, durante a sua execução;
- Na linguagem C a função de realocação dinâmica, que permite efetuar tal processo é a ***realloc()***;
  - ***realloc()*:** realoca a quantidade de memória necessária;

Realocação Dinâmica

Exemplo: função *realloc()*

- ***char* p*** = realloc(***void****, ***12*** * ***sizeof(char)***);
- Realoca o tamanho do ponteiro "p" para 12, o ***sizeof(char)*** garante que seja alocado 12 vezes o tamanho de um caractere em memória;
- Observe que diferentemente da função ***malloc()*** *a função* ***realloc()*** possui um parâmetro a mais (***void****) que indica se o conteúdo atual de ***"p"*** deve ser mantido após a realocação ou não;

Realocação Dinâmica

Exemplo: função *realloc()*

- ***Ex1.: char* p*** = realloc(***NULL***, ***12 * sizeof(char)***);
  - Realoca o ponteiro “p” com tamanho ’12’ sem manter o seu conteúdo;

- ***Ex2.: char* p*** = realloc(***p***, ***12 * sizeof(char)***);
  - Realoca o ponteiro “p” com tamanho ’12’ mantendo o seu conteúdo;

Realocação Dinâmica

Exemplo de Codificação (realocação dinâmica - *NULL*)

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

int main() {

        // declara um ponteiro para caracteres
        char *nome;
        // aloca o ponteiro 'nome' com tamanho 12 e atribui a string "gil eduardo"
        nome = malloc(12*sizeof(char));
        nome = "gil eduardo";
        printf("\nNome: %s", nome);
        // realoca o ponteiro 'nome' com tamanho 22, sem manter o seu conteúdo
        nome = realloc(NULL, 22*sizeof(char));
        printf("\nNome: %s", nome);

        printf("\n");
        return 0;
}
```

Realocação Dinâmica

Exemplo de Codificação (realocação dinâmica - *NULL*)

```
g1l3du4rd0@asus-ultrabook-g1l:/run/media/g1l3du4rd0/GEA/VirtualBox/Compartilhada/IFPR/2015/PC/aula11/exemplos/pc_exdoc...
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ gcc realocacao_realloc_null.c
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ ./a.out

Nome: gil eduardo
Nome:
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$
```

Realocação Dinâmica

Exemplo de Codificação (realocação dinâmica – *NOT NULL*)

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

int main() {

        // declara um ponteiro para inteiro
        int *vetor;
        // aloca o ponteiro 'vetor' com tamanho 3 e atribui os números '1' '2' '3'
        vetor = malloc(3*sizeof(int));
        vetor[0] = 0; vetor[1] = 1; vetor[2] = 2;
        printf("%i %i %i\n", vetor[0], vetor[1], vetor[2]);
        // realoca o ponteiro 'vetor' com tamanho 4 sem manter o seu conteúdo
        vetor = realloc(NULL, 4*sizeof(char));
        printf("%i %i %i\n", vetor[0], vetor[1], vetor[2]);
        // atribui os números '10' '15' '20' '25'
        vetor[0] = 10; vetor[1] = 15; vetor[2] = 20; vetor[3] = 25;
        printf("%i %i %i %i\n", vetor[0], vetor[1], vetor[2], vetor[3]);
        // realoca o ponteiro 'vetor' com tamanho 6 mantendo o seu conteúdo
        vetor = realloc(vetor, 5*sizeof(char));
        // atribui o número '30' a nova posição alocada - exibe o conteúdo do ponteiro
        vetor[4] = 30;
        printf("%i %i %i %i %i\n", vetor[0], vetor[1], vetor[2], vetor[3], vetor[4]);

        printf("\n");
        return 0;
}
```

Realocação Dinâmica

Exemplo de Codificação (realocação dinâmica – *NOT NULL*)

```
g1l3du4rd0@asus-ultrabook-g1l:/run/media/g1l3du4rd0/GEA/VirtualBox/Compartilhada/IFPR/2015/PC/aula11/exemplos/pc_exdoc...
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ gcc realocacao_realloc.c
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$ ./a.out
0 1 2
0 0 0
10 15 20 25
10 15 20 25 30

[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc11]$
```

Ponteiros e Alocação Dinâmica

Exemplos Utilizados no Documento

http://www.gileduardo.com.br/ifpr/pcii/downloads/pc_exdoc11.zip

Mais Exemplos sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pcii/downloads/pc_ex11.zip

Exercícios sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pcii/downloads/pc_pratica11.pdf