Programação de Computadores I

Vetores e Strings

int vetor[] / rand() / #define / char string[] / gets()

Gil Eduardo de Andrade

Vetores

Introdução

- Um vetor pode ser definido como é um conjunto de elementos (variáveis) que apresentam um mesmo tipo e são referenciados por um nome comum;
- Essas variáveis que compõem um vetor são acessadas através de um índice inteiro, onde o índice de menor valor (índice '0') corresponde ao primeiro elemento do vetor, enquanto o índice de maior valor corresponde ao último elemento;

Vetores

Declaração de Vetores

- A declaração de vetores é efetuada, de forma geral, da seguinte maneira:
  - ***tipo nome_vetor[tamanho];***
  - ***tipo:*** indica qual é o tipo das variáveis que compõem o vetor – ex.: ***int***, ***float***, ***double***;
  - ***nome_vetor:*** indica o nome pelo qual o vetor é referenciado;
  - ***tamanho***: indica o tamanho do vetor, o número de elementos total que o copõe;

Vetores

Vetores Inteiros:

- Considerando o slide anterior, poderíamos então declarar um vetor do tipo ***int*** de tamanho ***10***. Teríamos:
  - ***int numeros***[***10***];
- Ao declararmos o vetor ***numeros*** garantimos que o espaço de memória necessário para armazenar todos os seus elementos seja reservado;

Vetores

Vetores Inteiros:

- Contudo é importante observar que apenas declaramos o vetor, ou seja, ainda não especificamos os elementos (valores) que ele deve armazenar;
- Antes desta especificação, dizemos que o vetor possui armazenado ***"lixo de memória",*** valores quaisquer gerados por outros programas que utilizaram o mesmo espaço de memória num momento anterior;

Vetores

Armazenando Valores num Vetor:

- Como mencionado, o tamanho do vetor indica quantos elementos podemos armazenar num vetor;
- Para que seja possível especificar qual elemento do vetor receberá um dado que queremos armazenar, utilizamos um índice (valor inteiro) que indica a posição do vetor que receberá o dado;

Vetores

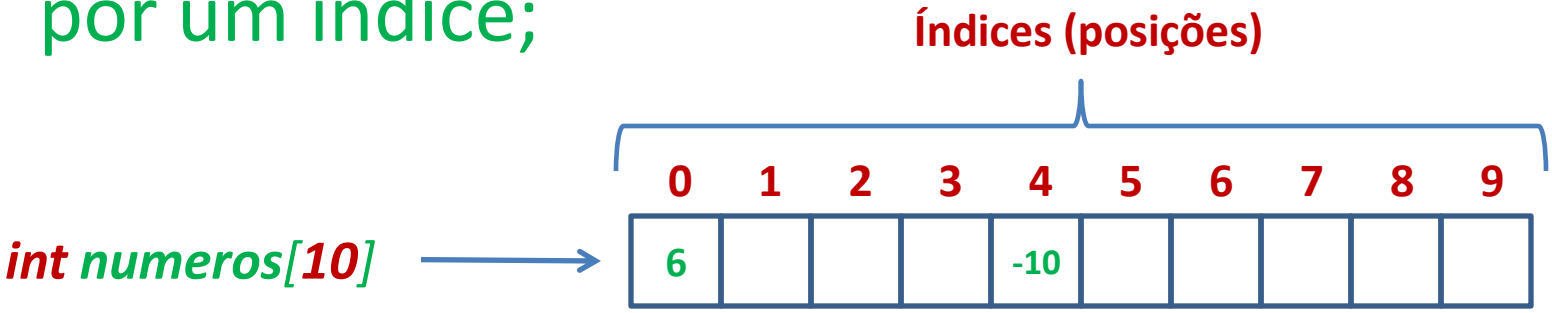
Armazenando Valores num Vetor:

- Os índices do vetor variam de ‘0’ até seu ‘tamanho-1’, ou seja, o vetor ***int numeros[10]*** possui os índices: ***1, 2, 3, 4, 5, 6, 7, 8 e 9***:
- Para armazenar dados neste vetor utilizaríamos a seguinte sintaxe:
  - ***numeros[0] = 6;*** ***numeros[4] = -10;***

Vetores

Representação Gráfica:

- Um vetor, em programação de computadores, pode ser representado por um retângulo dividido em um número de partes que é igual ao seu tamanho. Cada parte é referenciada por um índice;



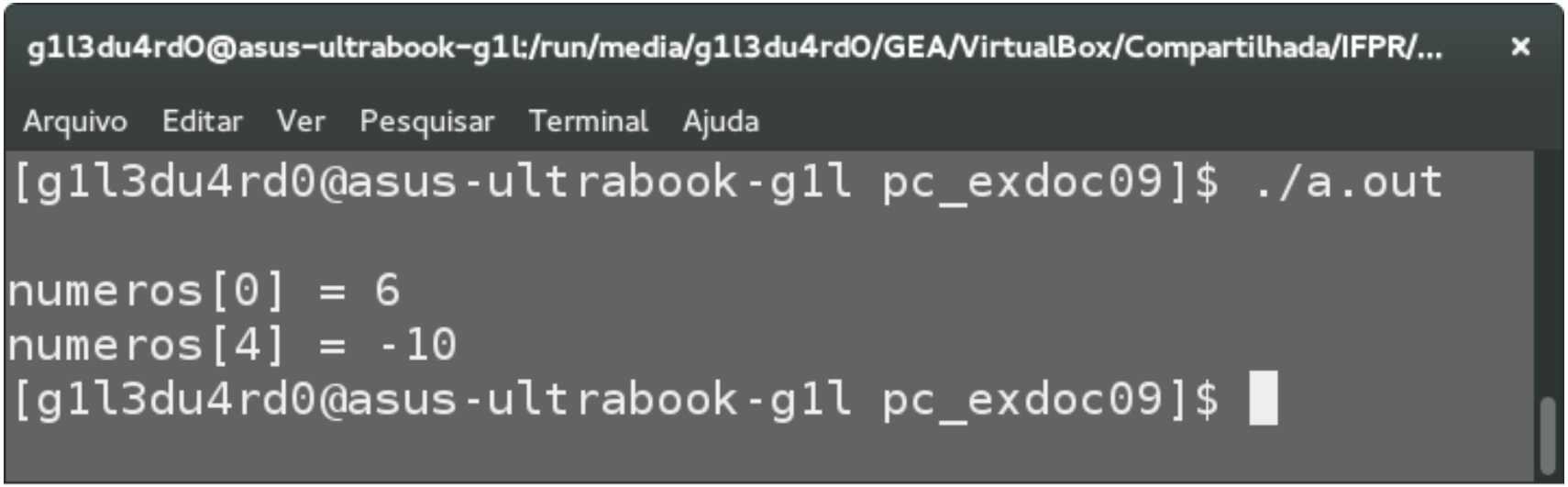
Vetores

Exemplos de Codificação (Atribuindo valores):

```c
#include <stdio.h>

int main() {

        // declara o vetor numeros do tipo 'int' com tamanho '10'
        int numeros[10];

        // atribui o valor '6' a posição '0' do vetor
        numeros[0] = 6;
        // atribui o valor '6' a posição '0' do vetor
        numeros[4] = -10;

        printf("\nnumeros[0] = %i", numeros[0]);
        printf("\nnumeros[4] = %i", numeros[4]);

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

Vetores

Exemplos de Codificação (Atribuindo valores):



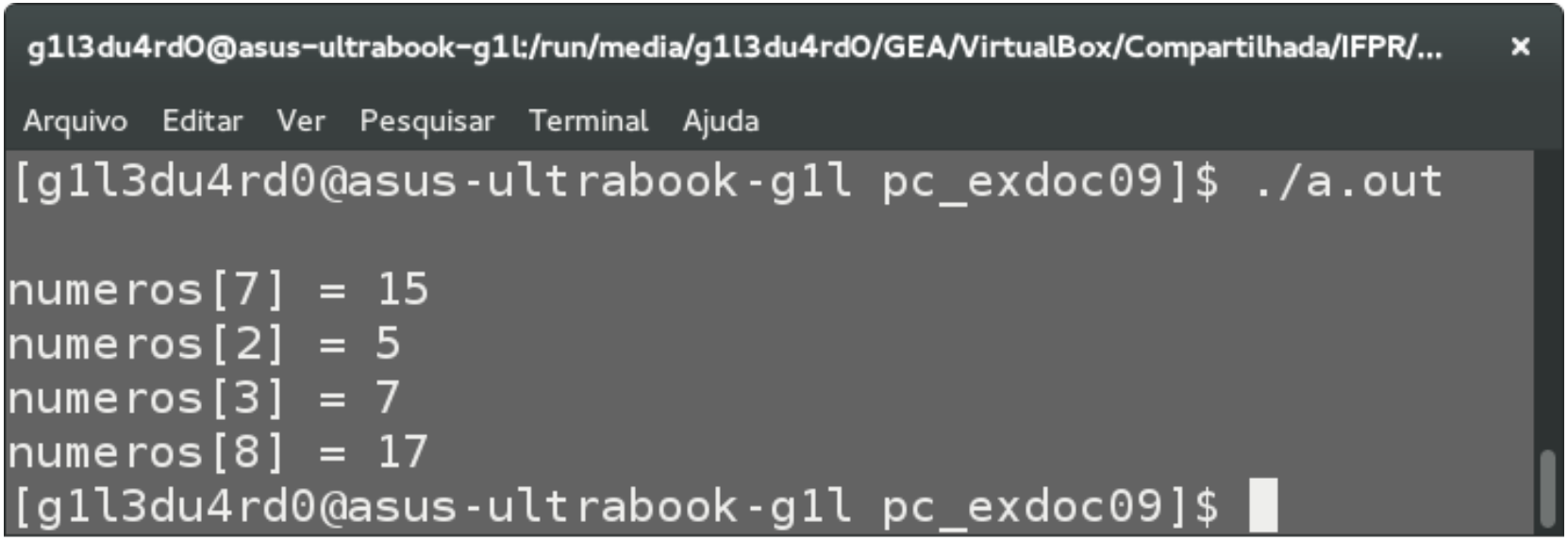
Vetores

Exemplos de Codificação (Declaração e Inicialização):

```c
#include <stdio.h>

int main() {

        // declara o vetor numeros do tipo 'int' com tamanho '10'
        // e já o iniciliza com valores pré-definidos
        int numeros[10] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};

        printf("\nnumeros[7] = %i", numeros[7]);
        printf("\nnumeros[2] = %i", numeros[2]);
        printf("\nnumeros[3] = %i", numeros[3]);
        printf("\nnumeros[8] = %i", numeros[8]);

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

Vetores

Exemplos de Codificação (Declaração e Inicialização):



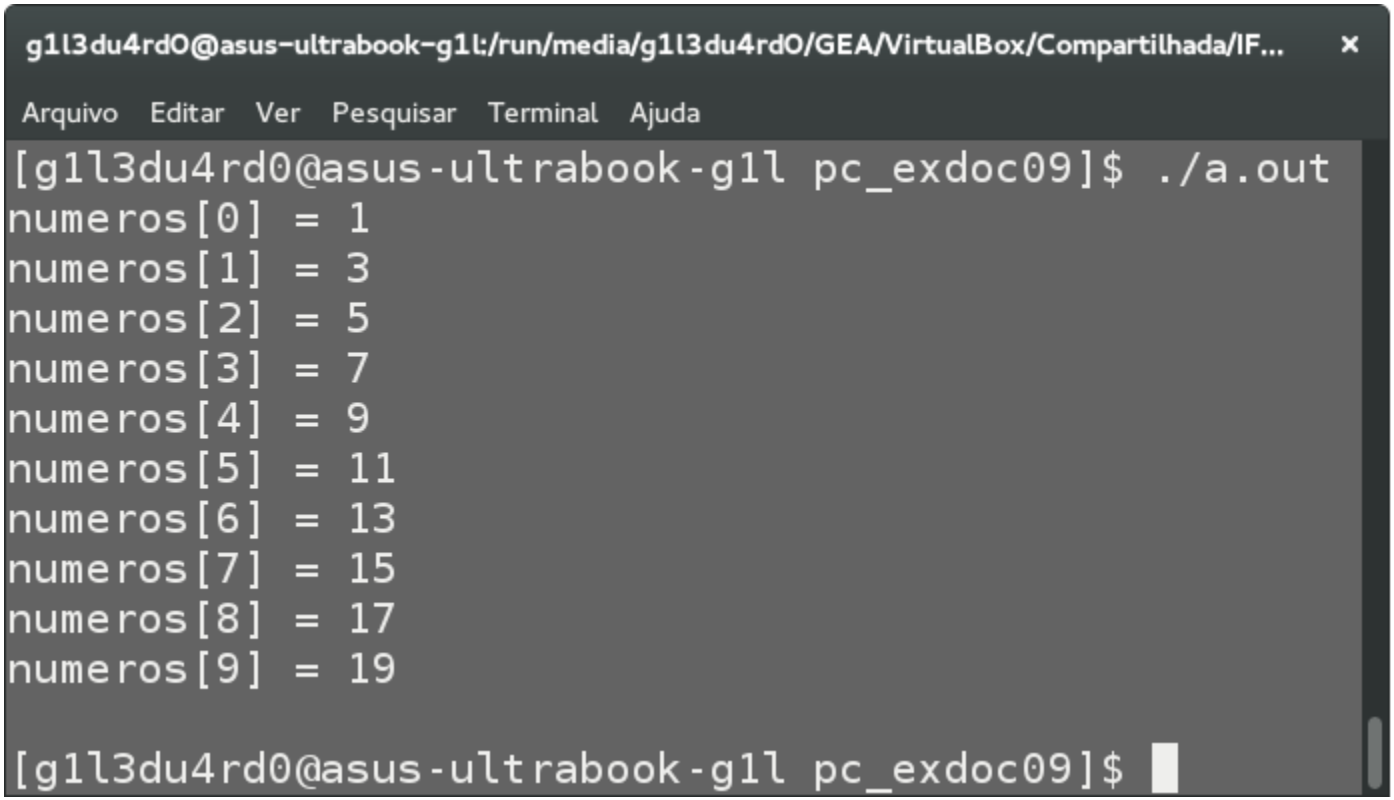
Vetores

Exemplos de Codificação (Percorrendo um Vetor):

```c
#include <stdio.h>

int main() {

        // declara o vetor numeros do tipo 'int' com tamanho '10'
        // e já o iniciliza com valores pré-definidos
        int numeros[10] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19};
        int indice;

        for(indice=0; indice<10; indice++) {
                printf("numeros[%i] = %i\n", indice, numeros[indice]);
        }

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

Vetores

Exemplos de Codificação (Percorrendo um Vetor):



Gerando Valores Aleatórios

Utilizando srand() e rand():

- Quando trabalhamos com vetores é comum necessitarmos gerar valores aleatórios para serem armazenados em suas posições;
- Para tal, em linguagem C, trabalhamos com as funções ***srand()*** e ***rand()*** disponíveis na biblioteca ***<time.h>***;

Gerando Valores Aleatórios

Utilizando srand() e rand():

- Primeiramente devemos incluir a biblioteca ***<time.h>*** no topo do código fonte C, juntamente com outras bibliotecas, por exemplo, a ***<stdio.h>***;
- O segundo passo é gerar uma semente que nos permite obter números aleatórios (pseudo). Essa semente é gerada pela função ***srand(time(***null***))*** que deve ser colocada logo após a declaração das variáveis;

Gerando Valores Aleatórios

Utilizando srand() e rand():

- Efetuados os dois passos anteriores já podemos utilizar o método ***rand()*** que permite obter valores inteiros aleatórios dentro de um limite pré-definido;
- A função ***rand()*** deve ser utilizada precedida do operador matemático ***%*** e do valor inteiro que indica o limite superior dos valores que podem ser gerados;

Gerando Valores Aleatórios

Utilizando srand() e rand():

- Por exemplo, se utilizarmos a seguinte linha de código:
  - ***rand()%10***: estaremos gerando valores inteiros aleatórios entre ***0 e 9***. Isso possibilita perceber que o valor máximo sempre será um valor menor que o limite inteiro especificado, ou seja, no exemplo com ***10***, teremos um valor aleatório máximo de ***9***;

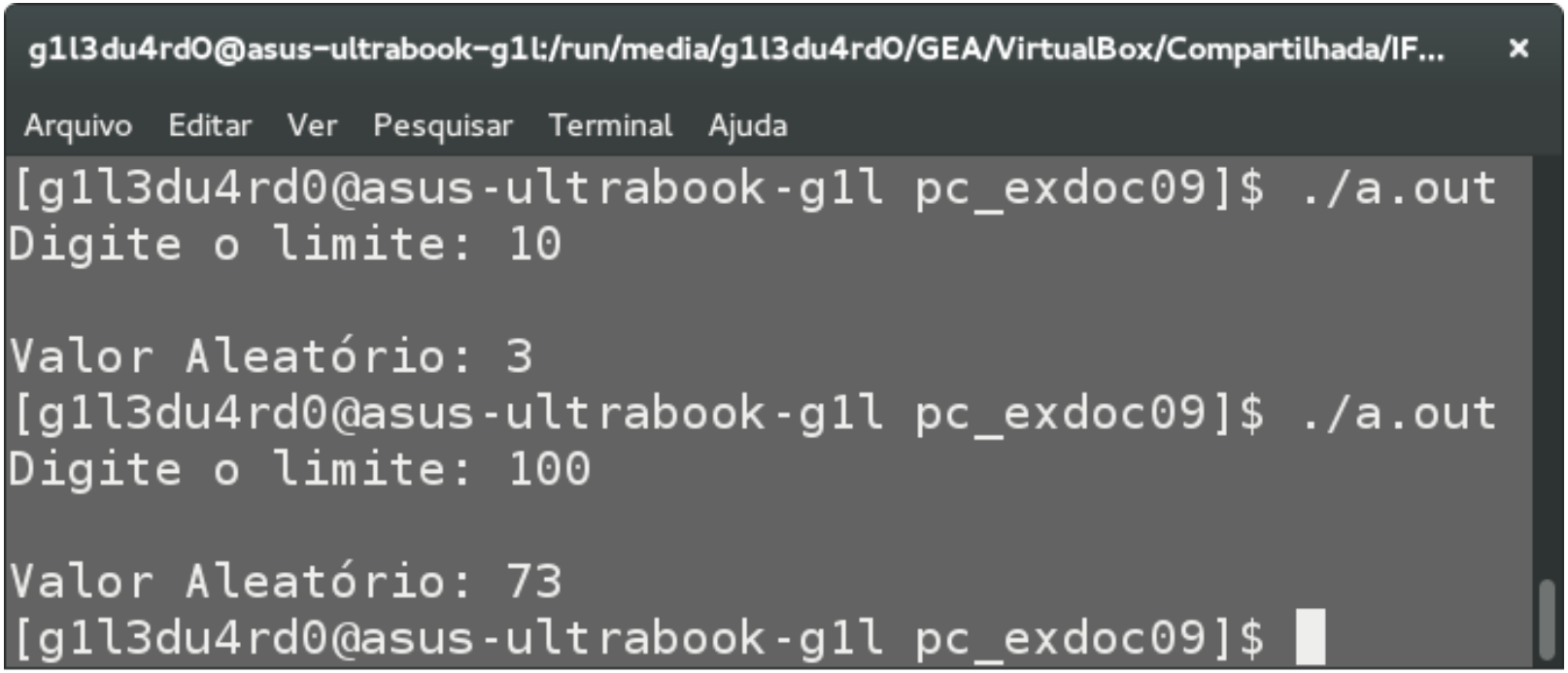
Gerando Valores Aleatórios

Exemplos de Codificação (Valores aleatórios entre 0 - ?):

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

int main() {

        int valor, limite;

        // Gera uma semente, necessária para obter
        // números pseudo-aleatórios
        srand(time(NULL));

        printf("Digite o limite: ");
        scanf("%i", &limite);

        // Gera um número pseudo-aleatório entre "0"
        // e o "limite-1" escolhido pelo usuário
        valor = rand()%limite;

        printf("\nValor Aleatório: %i", valor);

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

Gerando Valores Aleatórios

Exemplos de Codificação (Valores aleatórios entre 0 - ?):



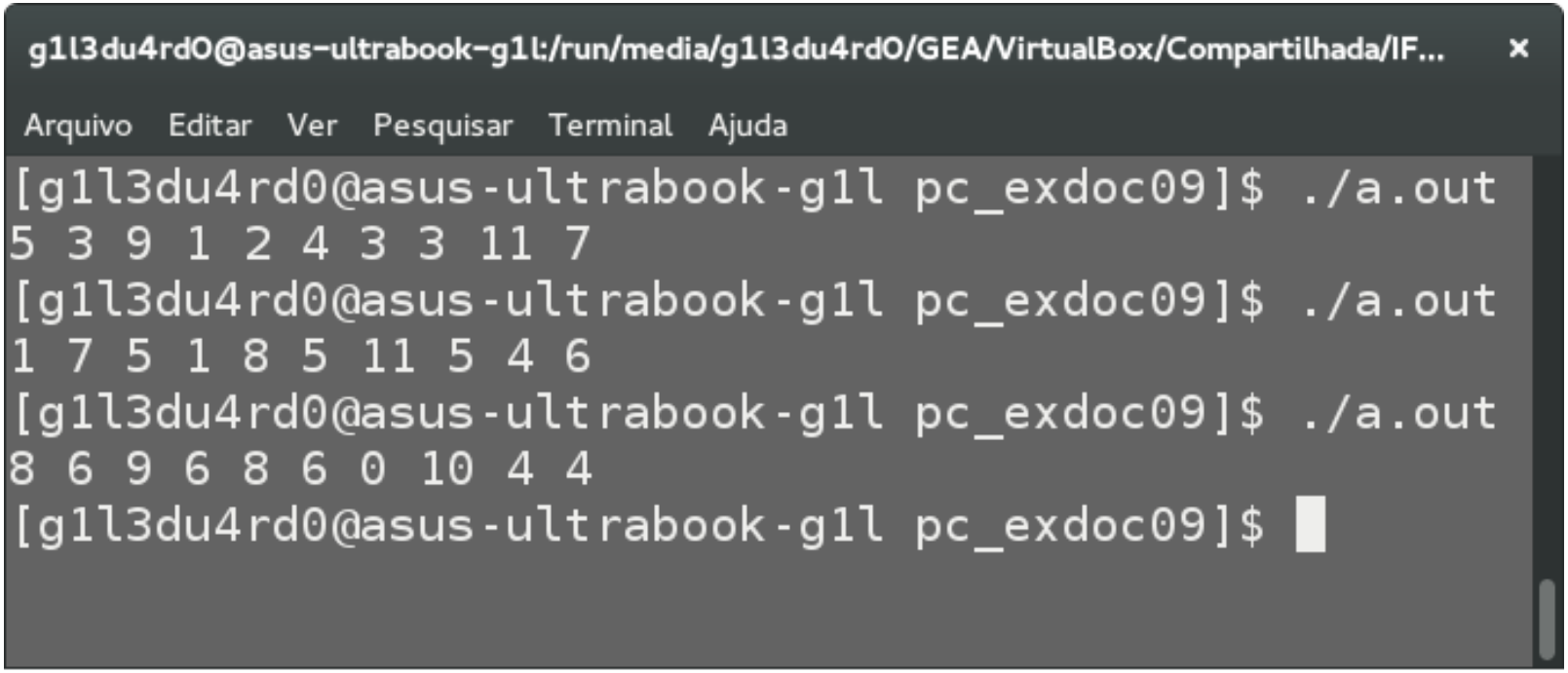
Gerando Valores Aleatórios

Exemplos de Codificação (Vetor com valores aleatórios):

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

int main() {

        int vetor[10], indice;

        // Gera uma semente, necessária para obter
        // números pseudo-aleatórios
        srand(time(NULL));

        // Gera e imprime um vetor com valores aleatorios
        for(indice=0; indice<10; indice++) {
                vetor[indice] = rand()%13;
                printf("%i ", vetor[indice]);
        }

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

Gerando Valores Aleatórios

Exemplos de Codificação (Vetor com valores aleatórios):



Constantes: *#define*

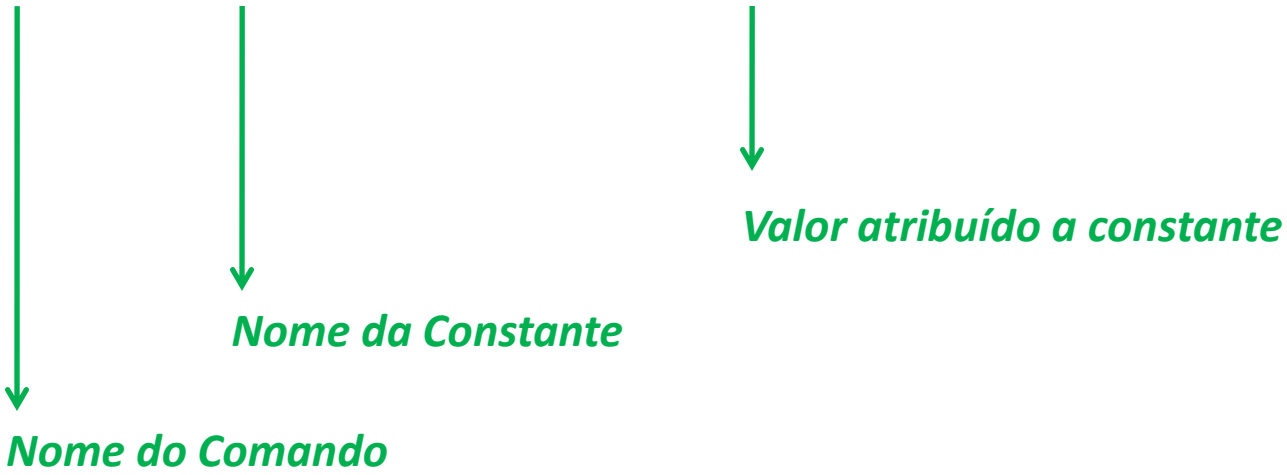
Utilizando *#define*:

- O comando ***#define*** permite especificar valores que serão constantes ao longo do código-fonte C que será implementado;
- O ***#define*** deve ser utilizado entre o trecho de código onde efetuamos a ***inclusão das bibliotecas*** e o código relativo a função ***main()***;

Constantes: *#define*

Sintaxe *#define*:

- O ***#define*** segue a seguinte sintaxe:
  - ***#define*** *nome_constante* ***valor***;



Constantes: *#define*

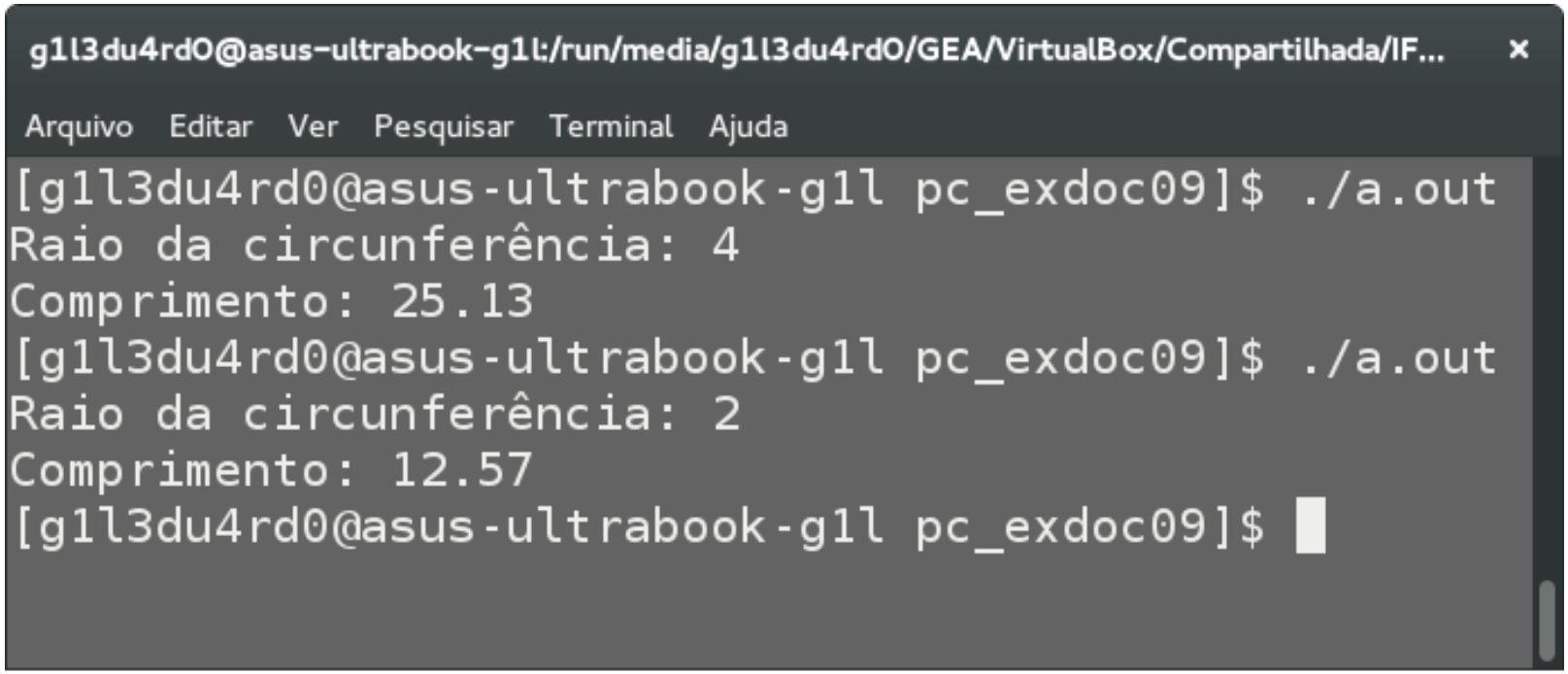
Exemplos de Codificação (#define básico):

```c
#include <stdio.h>

// define a constante PI = 3.1415
#define PI 3.1415

int main() {

        double raio, comp;

        printf("Raio da circunferência: ");
        scanf("%lf", &raio);

        comp = 2*PI*raio;
        printf("Comprimento: %.2lf", comp);

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

Observe que a constante PI foi escrita em caixa alta (letras maiúsculas). Este procedimento é recomendado como boas práticas de programação, já que ao longo do código-fonte fica fácil reconhecer o que é variável e o que constante, tanto para o programador quanto para quem está visualizando o código pela primeira vez.

Constantes: *#define*

Exemplos de Codificação (#define básico):



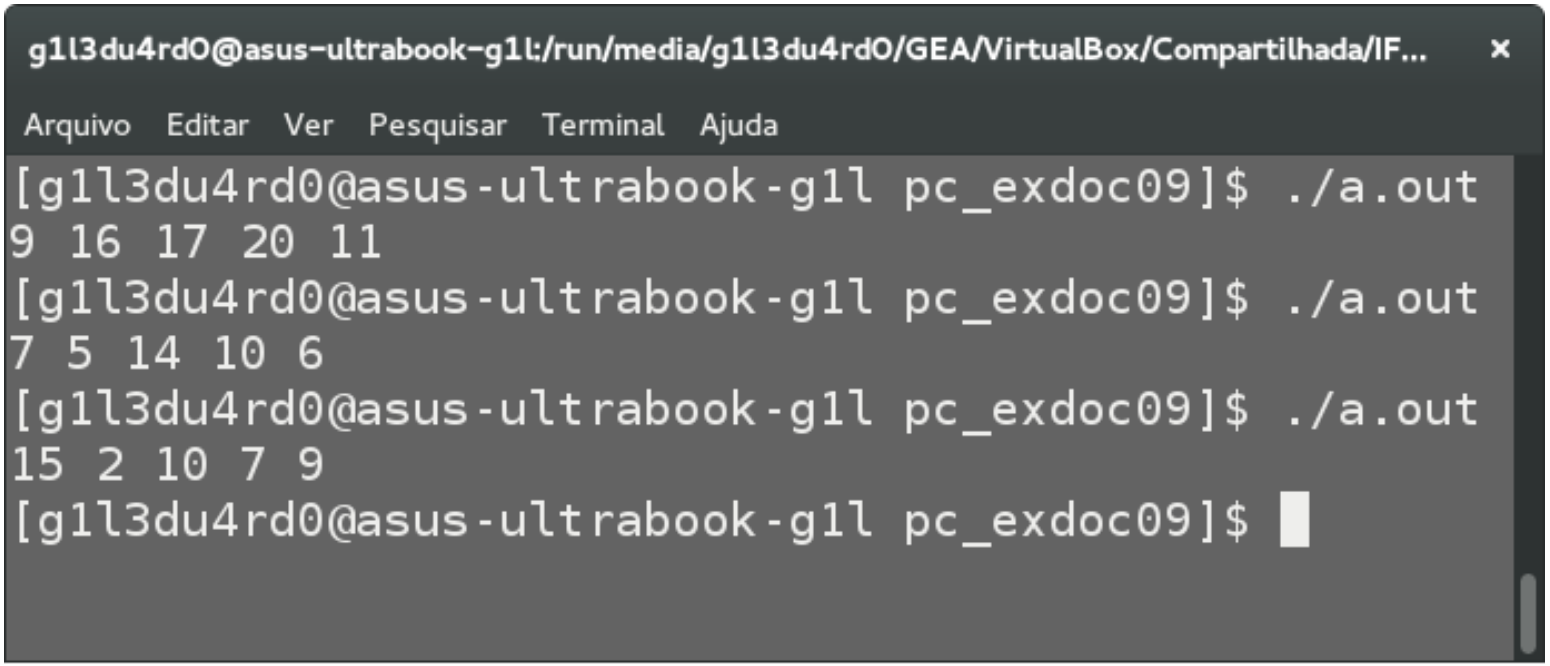
Constantes: *#define*

Exemplos de Codificação (#define com vetor):

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

// Define as constante TAM = 5 e LIM = 21
#define TAM 5
#define LIM 21


int main() {

        int vetor[TAM], a;

        srand(time(NULL));

        for(a=0; a<TAM; a++) {
                vetor[a] = rand()%LIM;
                printf("%i ", vetor[a]);
        }

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

Constantes: *#define*

Exemplos de Codificação (#define com vetor):



Strings: *vetores do tipo 'char'*

Definição *String*:

- Quando trabalhamos com vetores de caracteres (***char***), dizemos que estamos trabalhando com Strings;
- Dentro desse contexto, podemos definir Strings como um conjunto de caracteres, ou ainda, um conjunto de elementos do tipo ***char***;

Strings: *vetores do tipo 'char'*

Declarando *Strings*:

- A declaração de Strings (vetores ***char***), acontece da mesma forma como vimos para vetores inteiros (***int***);
  - ***Ex.: char rua[20];***

- Onde ***char*** é o tipo dos elementos que compõem a string, ***rua*** é nome da String e ***10*** é o tamanho da string;

Strings: *vetores do tipo 'char'*

Capturando *Strings*:

- Para que seja possível efetuar a leitura de uma String, considerando que a mesma é composta por vários elementos do tipo ***char*** utilizamos a função ***gets()*** ao invés da função ***scanf()***;
  - ***Ex.: gets(rua);***

Strings: *vetores do tipo 'char'*

Imprimindo *Strings*:

- Quando precisamos imprimir o conteúdo de uma String, considerando que o mesmo é composto, normalmente, por vários caracteres, utilizamos o comando ***printf()*** com o modificador ***%s*** ***(s – string)***;
  - ***Ex.: printf(%s, rua);***

Strings: *vetores do tipo 'char'*

Final de *Strings – caractere '\0'*:

- Quando precisamos percorrer uma String utilizando, normalmente, um laço de repetição, é preciso identificar o seu término, como condição de parada para do laço;
- Para tal as Strings possuem o caractere especial ***'\0'*** em sua última posição, indicando o seu término;

Strings: *vetores do tipo 'char'*

Final de *Strings – caractere '\0'*:

- Portanto ao efetuarmos a leitura de uma String digitada pelo usuário, da qual não sabemos o seu tamanho inicialmente, podemos utilizar um laço de repetição e o caractere ***'\0'*** para percorrer toda essa String;
  - ***Ex.: for(a=0; rua[a] != '\0'; a++) { };***

Strings: *vetores do tipo 'char'*

Exemplos de Codificação (declaração e inicialização String):

```c
#include <stdio.h>

int main() {

        // declarando e inicializando uma String
        char rua[20] = {'P', 'e', 'd', 'r', 'o', ' ', 'I', 'v', 'o'};

        printf("String 'rua': %s", rua);

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

```
g1l3du4rdO@asus-ultrabook-g1l:/run/media/g1l3du4rdO/GEA/VirtualBox/Compartilhada/IF...
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda

[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc09]$ ./a.out
String 'rua': Pedro Ivo
[g1l3du4rd0@asus-ultrabook-g1l pc_exdoc09]$
```

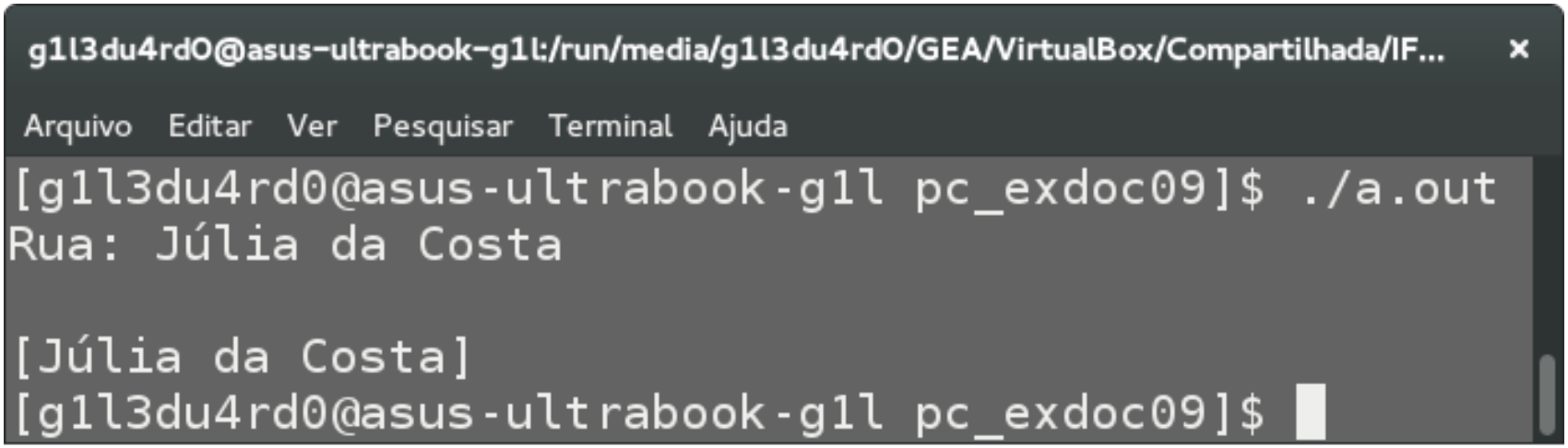
Strings: *vetores do tipo 'char'*

Exemplos de Codificação (Capturando e imprimindo String):

```c
#include <stdio.h>

int main() {

        // declara uma String com nome 'rua'
        char rua[20];

        // caputura e armazena na String 'rua'
        printf("Rua: ");
        __fpurge(stdin); //fflush(stdin);
        gets(rua);

        // imprime a String 'rua'
        printf("\n[%s]", rua);

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

Constantes: *#define*

Exemplos de Codificação (Capturando e imprimindo String):



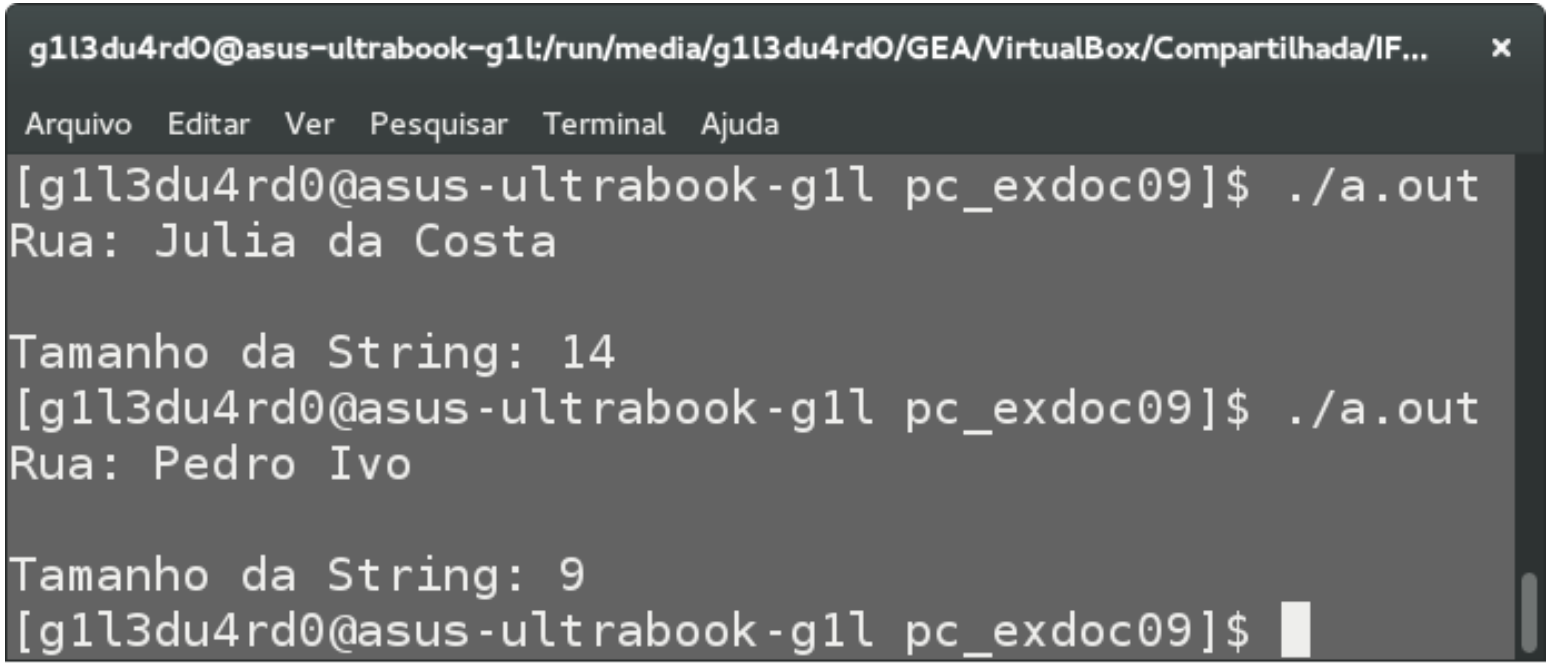
Constantes: *#define*

Exemplos de Codificação (Percorrendo uma String):

```c
#include <stdio.h>

int main() {

        // declara uma String com nome 'rua'
        char rua[20];
        int a;

        // caputura e armazena na String 'rua'
        printf("Rua: ");
        __fpurge(stdin); //fflush(stdin);
        gets(rua);

        // percorre e encontra o tamanho
        for(a=0; rua[a] != '\0'; a++){ }

        printf("\nTamanho da String: %i", a);

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

Constantes: *#define*

Exemplos de Codificação (Percorrendo uma String):



Operadores de Condição

Exemplos Utilizados no Documento

http://www.gileduardo.com.br/ifpr/pci/downloads/pc_exdoc09.zip

Mais Exemplos sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pci/downloads/pc_ex09.zip

Exercícios sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pci/downloads/pc_pratica09.pdf