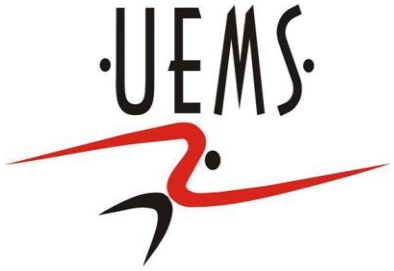


PCI- Variáveis, Atribuição, I/O

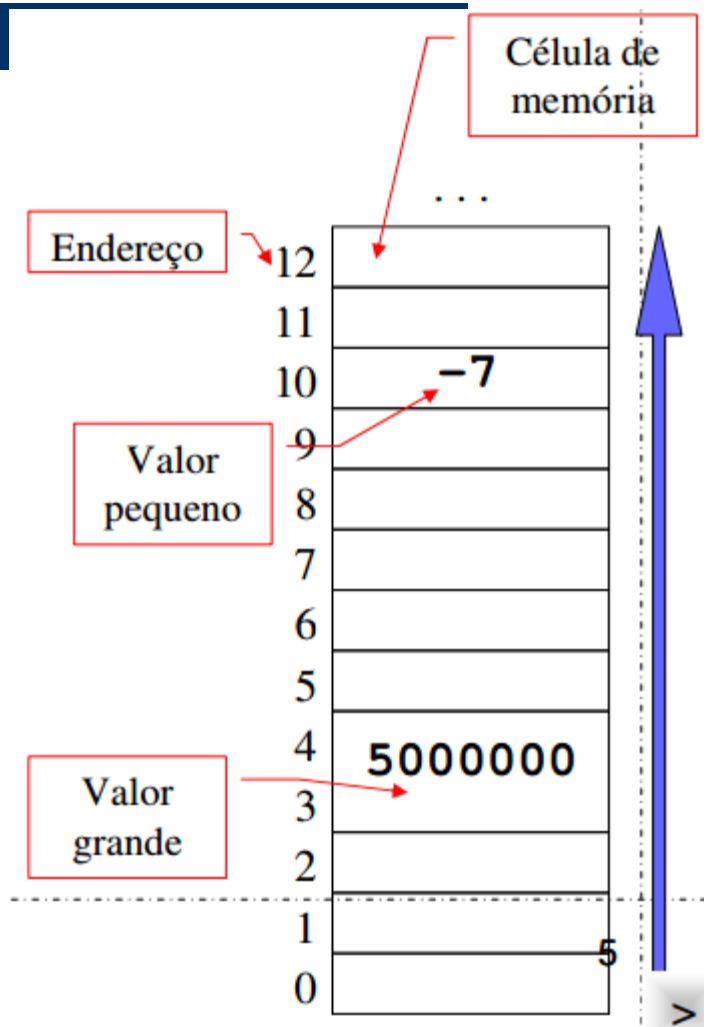
Profa. Mercedes Gonzales Márquez

Memória

- **Memória:** sequência de células
- **Endereço:** posição da célula
- **Células armazenam dados**

– **Valor pequeno:** uma célula

– **Valor grande:** duas ou mais células



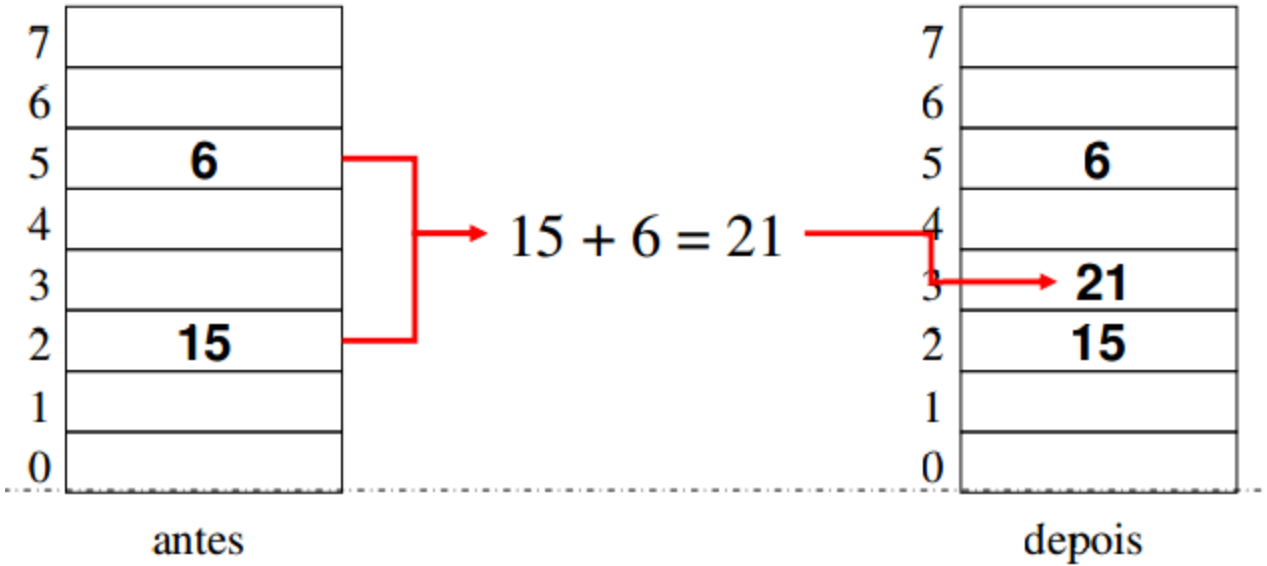
Memória

- **Operações na memória:**

1. Consulta (lê) células de memória

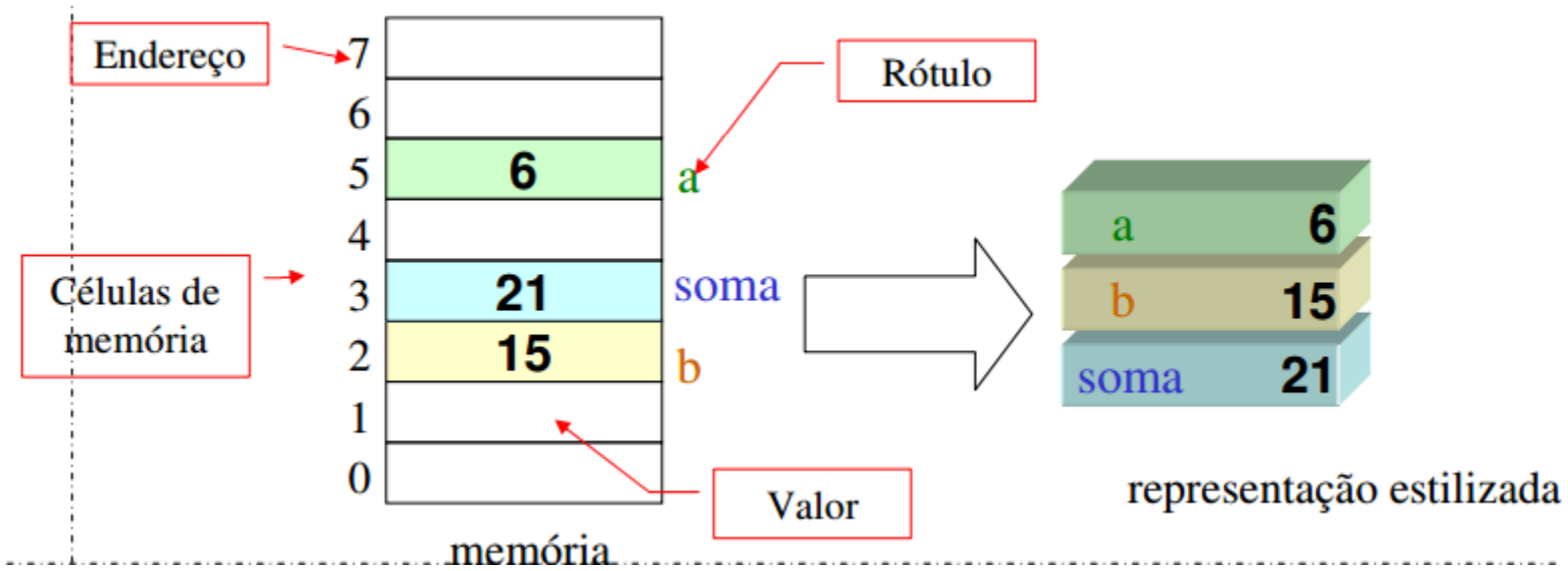
2. Programa calcula um novo valor

3. Armazena (escreve) o novo valor em uma célula



Variáveis

- Abstração dos endereços de memória (posição nomeada de memória)
- Rótulo simbólico para cada endereço que guarda um dado de um certo tipo.



Tipos primitivos de dados

- Em C, há três tipos primitivos de dados: **números inteiros**, números de **ponto flutuante** (números fracionários) e **caracteres** (como ‘a’, ‘#’, ‘Z’, ‘2’).

- Esses tipos são chamados de “primitivos” porque eles podem ser modificados e combinados de certas maneiras, de modo a criar estruturas mais complexas de dados, como veremos mais tarde.

Nota: Não deve confundir caracteres com *strings*, um caractere é colocado em aspas simples ‘’ e uma *string* é colocada em aspas duplas “”e consiste em uma sequência de caracteres

Declarando uma variável

- Todas as variáveis devem ser declaradas antes de serem utilizadas, fazendo com que o computador reserve na memória um espaço suficiente para guardar valores do tipo de dado que você pediu.
- Declara-se da seguinte forma:

  Tipo Variável Nome Variável;

  Vejamos alguns exemplos de declarações:

  int numero;

  float preco_total;

Variáveis inteiras

Os seguintes são os tipos da linguagem C que servem para armazenar inteiros:

int: Inteiro cujo comprimento depende do processador. E o inteiro mais utilizado. Em processadores Intel comum, ocupa 32 bits e pode armazenar valores de

-2.147.483.648 a 2.147.483.647.

unsigned int: Inteiro cujo comprimento depende do processador e que armazena somente valores positivos. Em processadores Intel comum, ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295.

Variáveis inteiras

long int: Inteiro que ocupa 64 bits em computadores Intel de 64bits e pode armazenar valores de aprox. -9 $x10^{18}$ a aprox. 9 $x10^{18}$.
unsigned long int: Inteiro que ocupa 64 bits e em computadores Intel de 64bits e armazena valores de 0 ate aprox. 18 $x10^{18}$.
short int: Inteiro que ocupa 16 bits e pode armazenar valores de -32.768 a 32.767.
unsigned short int: Inteiro que ocupa 16 bits e pode armazenar valores de 0 a 65.535.

Variáveis de tipo caractere

Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos. Guarda apenas um caractere.
Exemplos de declaração:
char umaLetra;

Variáveis de tipo ponto flutuante

Armazenam valores reais. Mas possuem problemas de precisão pois há uma quantidade limitada de memória para armazenar um número real.
Exemplos de números em ponto flutuante: 2.1345 ou 9098.123.

- **float:** Utiliza 32 bits, e na pratica tem precisão de aproximadamente 6 casas decimais (depois do ponto). Pode armazenar valores de $(+/-)10^{-38}$ a $(+/-)10^{38}$

double: Utiliza 64 bits, e na pratica tem precisão de aproximadamente 15 casas decimais. Pode armazenar valores de $(+/-)10^{-308}$ a $(+/-)10^{308}$

Regras para nomes de variáveis em C

- Deve começar e pode conter letra (maiúscula ou minúscula) ou underline(_).
- Não pode-se utilizar como parte do nome de uma variável: { (+ - * / \ ; . , ?
- Letras maiúsculas e minúsculas são diferentes.
- As seguintes palavras são reservadas na linguagem C e portanto não podem ser utilizadas como nome de variáveis:

auto double int struct break enum register typedef char extern return union const float short unsigned continue for signed void default goto sizeof volatile do

if static while

Constantes

- São valores previamente determinados e que não se alteram ao longo do programa.
- Assim como as variáveis, as constantes também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo string, que corresponde a uma sequência de caracteres. I Exemplos: 90, 0.23, ’c’, “Ana Ortiz".

- Constantes devem ser definidas logo no começo do programa, após as declarações das bibliotecas, de forma que elas possam ser usadas em qualquer ponto do programa.

Constantes

- Podemos definir uma constante usando o seguinte comando:

  #define CONSTANTE valor

Exemplos:

#define PI 3.14159

#define DIAS_FEVEREIRO 28

#define DIAS_DA_SEMANA 7

- Ao se definir uma constante (usando #define) não é alocado um espaço de memória para ela. O compilador apenas substitui o valor da constante em todo lugar que ela é usada no programa

Constantes e Variáveis

- Uma constante é uma expressão e, como tal, pode ser atribuída a uma variável ou ser usada em qualquer outro lugar onde uma expressão seja permitida.

Exemplo:

```
double raio, comp_circ;

raio = 1.0;
comp_circ= 2 * PI * raio;
```

Tipo Bool

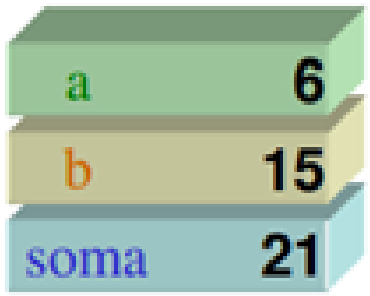
- Usando a biblioteca stdbool.h podemos usar um tipo de dados chamado bool (booleano ou lógico) que considera os valores true (=verdadeiro) e false (=falso) no programa.

```
bool b1, b2;
b1 = true; b2 = false;
if (b1) printf("%d\n", b2);  // Imprime 0
```

Comando de atribuição

- Uma das coisas mais simples que podemos fazer é guardar um valor numa variável: usamos um sinal de igual, escrevendo à sua esquerda o nome da variável, e à direita o valor que queremos guardar:

Sintaxe:
variável=valor; ou
variável=expressão




```
int main{
    int a,b,soma;
    a = 6; //valor
    b=15; //valor
    soma=a+b;
//expressão
}
```

Comando de saída: printf

Comando de saída: printf

Este comando apresenta números e caracteres na tela.

Sintaxe:

printf (“*string de controle*” , *argumento1, argumento2, ..., argumento n*)

Uma string de controle pode ter :

1. sequência de conversão (especifica um formato, inicia com %)
2. Sequência de escape (inicia com barra invertida \)

Comando de saída: printf

Exemplo de impressão de uma string de controle sem sequência de conversão.

```
int main(){
    printf (“Imprimindo Notas”);
}
```

Comando de saída: printf

1. A sequência de conversão faz parte da string de controle, inicia com % e é um “guardador de lugar” para as variáveis que se seguem, especificadas nos argumentos.

printf (“*string de controle*”, *argumento1, argumento2, ..., argumento n*)

%c um único caractere

%o, %d, %x um número inteiro em octal, decimal ou hexadecimal

%u um número inteiro em base decimal sem sinal

%ld um número inteiro longo em base decimal

%f, %lf um número real de precisão simples ou dupla

%s uma cadeia de caracteres (string)

Comando de saída: printf

- Exemplo de uso de um único argumento com sequência de conversão:

Colocamos o código %d onde queremos que a variável apareça; a variável em si é especificada à direita da string como o argumento correspondente

Se x é uma variável inteira com valor 12.
printf("Valor de x = %d", x);
imprime na tela a frase Valor de x = 12.

Comando de saída: printf

Exemplo do uso de 2 argumentos também com sequência de conversão:
Se y é uma variável do tipo char com valor 'A', então a execução de
printf("x = %d e y = %c", x, y);
imprime na tela a frase x = 12 e y = A

Comando de saída: printf

2. A sequência de escape inicia em barra invertida seguida por constantes que representam caracteres especiais. Os mais utilizados são:

Sequência	Significado
\n	Quebra de linha (*line feed* ou LF)
\t	Tabulação horizontal
\b	Retrocede o cursor em um caractere (backspace)
\r	Retorno de carro (*carriage return* ou CR): volta o cursor para o começo da linha sem mudar de linha
\a	Emite um sinal sonoro
\"	Aspa dupla
\'	Aspa simples
\\	Barra invertida
\0	Caractere nulo (caractere de valor zero, usado como terminador de strings)

Comando de saída: printf

Exemplo 1 usando sequências de conversão e de escape

```
#include <stdio.h>
// Exemplos de uso de string de controle com sequências de
conversão e de escape
int main(){
    int x=12;
    char y='A';
    float z=1.567;
    printf("Imprimindo Valores ...\n");
    printf ("Para imprimir uma aspa dupla faco assim \"\n");
    printf ("Para imprimir uma barra invertida faco assim \\\n");
    printf("Valor de x = %d\t Valor de y = %c\t Valor de z = %f",x,y,z);
}
```

Comando de saída: printf

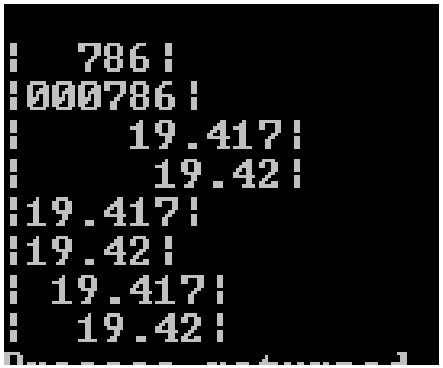
Exemplo 2 usando sequências de conversão e de escape

```
#include<stdio.h>
#define PRECO 1.99
int main(){
   int pera = 3;
   char qualidade = 'A';
   float peso = 2.5, total=peso*PRECO;
   printf("Existem %d peras de qualidade %c", pera, qualidade);
   printf("pesando %f quilos.\n", peso);
   printf("O preco por quilo e %f, total e %f\n", PRECO, total);
}
```

Comando de saída: printf

- A função printf() permite que os campos de exibição sejam formatados. As formatações mais usadas são o preenchimento com zeros à esquerda, para inteiros, e a especificação do número de casas decimais, para reais.

```
int a = 786;
float b = 19.4165;
printf("\n|%5d|",a);
printf("\n|%06d|",a);
printf("\n|%10.3f|",b);
printf("\n|%10.2f|",b);
printf("\n|%1.3f|",b);
printf("\n|%1.2f|",b);
printf("\n|%7.3f|",b);
printf("\n|%7.2f|",b);
```



Comando de entrada: scanf

Comando de entrada: scanf

A função scanf() permite que um valor seja lido do teclado e armazenado numa variável.

Sintaxe:

scanf("*cadeia de formatação*", arg1, arg2, ..., argn);

A cadeia de formatação é composta por n símbolos de conversão que indicam os tipos dos dados que serão lidos pela função. E depois uma lista de argumentos, cada um deles sendo o endereço de uma variável.

Exemplo:

```
int idade;
char sexo;
scanf("%d %c", &idade, &sexo);
```

Exemplos de programas

Exemplo1: Faça um programa que após informar o ano atual e o ano de nascimento de uma pessoa, imprima a idade inteira da pessoa.

```
#include<stdio.h>
int main(){
    int ano1,ano2;
    printf(“Digite o seu ano de nascimento:");
    scanf(“%d",&ano1);
    printf(“Digite o ano atual:");
    scanf(“%d",&ano2);
    printf(“Voce tem ou tera neste ano %d anos \n",ano2-
    ano1);
}
```

Comandos de entrada/saída

Exemplo 2: Faça um programa que, após informar a base e a altura de um retângulo, imprima a área do mesmo.

```
#include<stdio.h>
int main(){
    float base, altura, area;
    printf`(“Digite a base do retangulo:");
    scanf(“%f",&base);
    printf(“Digite a altura do retangulo:");
    scanf(“%f",&altura);
    area = base* altura;
    printf(“Valor da area e: %f\n",area);
}
```

Comandos de entrada/saída

Exercício: Faça um programa que leia seu nome e ano de nascimento, calcule sua idade baseada no ano atual e o seu ano de nascimento e imprima a seguinte mensagem:

_____, seja bem-vindo ao curso de Sistemas de Informação, espera-se que você conclua o curso no ano _____ , ou seja, quando tiver _____ anos.

Comandos de entrada/saída

Para ler e escrever um nome que consiste em um conjunto de dados char, devemos declarar um vetor de char (conjunto de char ou também chamado de string). Exemplo: char nome[6], neste caso será aceito um nome de no máximo 6 caracteres.

A sequência de conversão para variáveis string é %s.

Comandos de entrada/saída

```
#include <stdio.h>
#include <locale.h>
int main(){
    setlocale(LC_ALL,"Portuguese");
    int ano_nasc, ano_atual, idade;
    char nome[8]; /* usamos uma string que eh um vetor de char */
    printf ("Informe o seu nome:");
    scanf ("%s", nome); /*o simbolo de conversao de uma string e %s */
    printf ("Informe o ano atual:");
    scanf ("%d", &ano_atual);
    printf ("Infome o ano de seu nascimento:");
    scanf ("%d", &ano_nasc);
    idade=ano_atual-ano_nasc;
    printf ("%s, seja bem-vindo ao curso de Sistemas de
Informação.\nEspera-se que você conclua o curso no ano %d, ou seja,
quando tiver %d anos", nome, ano_atual+4, idade+4);
}
```