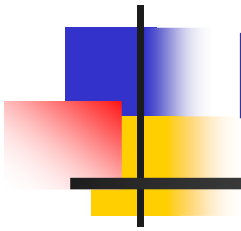
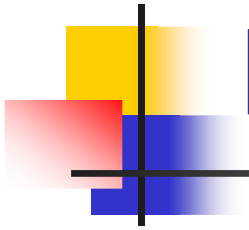


Linguagem C: Introdução



Linguagem C

- É uma Linguagem de programação genérica que é utilizada para a criação de programas diversos como:
  - Processadores de texto
  - Planilhas eletrônicas
  - Sistemas operacionais
  - Programas de comunicação
  - Etc.

Estrutura de um Programa C

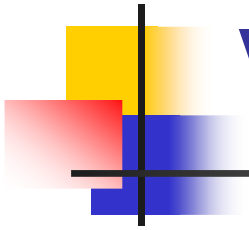
```
#include <stdio.h>
int main() {
printf("Bem-vindo ao mundo da programação!\n");
return 0;
}
```

- O programa acima exibe na tela a frase "Bem-vindo ao mundo da programação!"
- Todo programa em C deve possuir a função main() que representa o corpo principal do programa
- A função printf é responsável por imprimir (exibir) uma cadeia de caracteres (string) na tela
- Todos os comandos (com exceção do abre e fecha chaves devem ser finalizados com ponto e vírgula ";"
- A linguagem C é *case sensitive* ou seja, letras maiúsculas ou minúsculas fazem a diferença no código

A Diretiva #include

- Informa ao compilador para incluir na compilação do programa outros arquivos.
- Geralmente estes arquivos contém funções de bibliotecas ou rotinas do usuário.

Arquivo	Descrição
stdio.h	Funções de entrada e saída (I/O)
stdlib.h	Funções de uso genérico
string.h	Funções de tratamento de strings
math.h	Funções matemáticas
ctype.h	Funções de teste e tratamento de caracteres

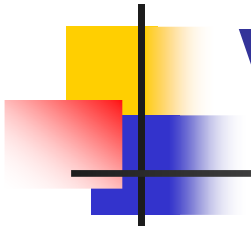


Variáveis

- Locais onde armazenamos valores na memória
- Toda variável é caracterizada por um **NOME**, que a identifica em um programa, e por um **TIPO**, que determina o que pode ser armazenado naquela variável
- Declaração de uma variável

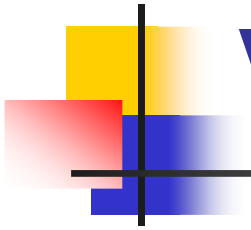
  (tipo) nome da variável;
- Exemplo:

  int soma;



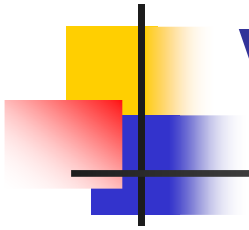
Variáveis Inteiras

- Variáveis utilizadas para armazenar valores inteiros, em formato binário
  - Formato: int <nome da variável>
  - Exemplo: int soma;
- O número de bytes para armazenar um valor inteiro depende do computador. Se forem associados 2 bytes podemos armazenar números na faixa de -32768 a 32767.
- Existem outras formas de se declarar tipos inteiros:
  - **unsigned int**: Inteiro cujo comprimento depende do computador e que armazena somente valores positivos. Ocupa 16 bits e pode armazenar valores de 0 a 65.535
  - **long int**: Inteiro que ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647, independente do computador
  - **unsigned long int**: Inteiro que ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295, independente do computador



Variáveis Caracteres

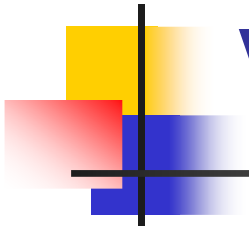
- Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos
  - São, na verdade, variáveis inteiras que armazenam um número associado ao símbolo
  - A principal tabela de símbolos utilizada pelos computadores é a tabela ASCII(*American Standard Code for Information Interchang*), mas existem outras tabelas, como o UNICODE
- Formato: char <nome da variável>.
  - Armazena um símbolo (no caso, o inteiro correspondente). Seu valor pode ir de -128 a 127
- Existe uma outra forma de se declarar caracteres.
  - Formato :<unsigned char> <nome da variável>
  - Armazena um símbolo (no caso, o inteiro correspondente). Seu valor pode ir de 0 a 255



Variáveis Reais

- Formato : float <nome da variável>
  - Armazena valores reais
  - Utiliza 32 bits, sendo 1 para o sinal, 8 para o expoente e 23 para a mantissa
  - Pode armazenar valores de (+/-)10^{-38} a (+/-)10^{38}
- Existe uma outra forma de se declarar reais:
  - Formato: double <nome da variável>
  - Utiliza 64 bits, sendo 1 para o sinal,11 para o expoente e 52 para a mantissa
  - Pode armazenar valores de (+/-)10^{-308} a (+/-)10^{308}

Regras para Nomes de Variáveis

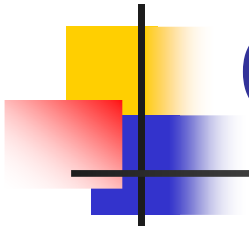


- Variáveis devem começar com uma letra (maiúscula ou minúscula) ou subscrito(_)
- Nunca podem começar com um número
- Podem conter letras maiúsculas, minúsculas, números e subscrito
- Não é possível utilizar { (+ - * / \ ; . , ? como parte do nome de uma variável

Regras para Nomes de Variáveis (cont.)

- As seguintes palavras já têm um significado na linguagem C e por esse motivo não podem ser utilizadas como nome de variáveis:

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

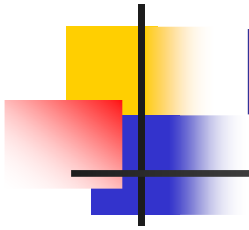


Constantes

- Valores previamente determinados e que, por algum motivo, devem aparecer dentro de um programa
- Definidas através do comando

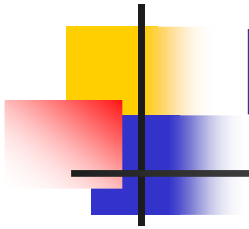
  #define

  - Exemplo: #define IDADE 34
- Exemplos de constantes: 85, 0.10, ’c’, ”Meu primeiro programa”



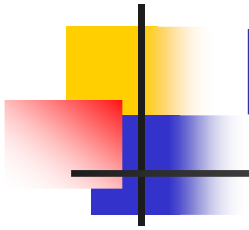
Escrevendo Dados

- Escrevendo o conteúdo de uma variável na tela
  - Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando printf
  - Para isso, utilizamos um símbolo no texto para representar que aquele trecho deve ser substituído por uma variável e, no final, passamos uma lista de variáveis ou constantes, separadas por vírgula
- Exemplo: printf (”A variavel %s contem o valor %d”,”a”, a);
  - O código acima imprime *A variável a contem o valor 10*
  - Nesse caso, %s deve ser substituído por uma variável ou constante do tipo string enquanto %d deve ser substituído por uma variável ou constante do tipo inteiro



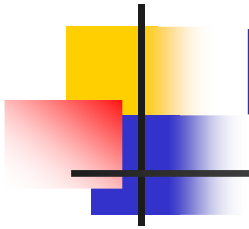
Escrevendo Inteiros

- %d: Escreve um inteiro na tela sem formatação
  - Ex: printf ("%d", 10) imprime 10
  - %< numero >d: Escreve um inteiro na tela, preenchendo com espaços à esquerda para que ele ocupe pelo menos < numero > casas na tela
  - Ex: printf ("%4d", 10) imprime < espaço >< espaço >10
- %0< numero >d: Escreve um inteiro na tela, preenchendo com zeros àesquerda para que ele ocupe pelo menos comprimento < numero >
  - Exemplo: printf ("%04d", 10) imprime 0010
- %< numero1 >.0< numero2 >d: Escreve um inteiro na tela, preenchendo com espaços à esquerda para que ele ocupe pelo menos < numero1 > casas na tela e com zeros para que ele possua pelo menos comprimento <numero2 >.
  - Exemplo: printf ("%6.04d", 10) imprime < espaço >< espaço >0010
- A letra d pode ser substituída pelas letras u e l, ou as duas, quando desejamos escrever variáveis do tipo unsigned ou long, respectivamente.
  - Exemplo: printf ("%d", 4000000000) escreve -294967296 na tela, enquanto que printf("%u", 4000000000) escreve 4000000000



Escrevendo Outros Números

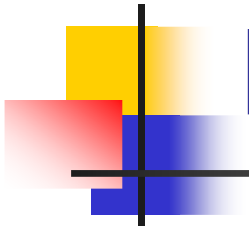
- %f : Escreve um ponto flutuante na tela, sem formatação
  - Exemplo: printf ("%f", 10.0) imprime 10.000000
- %e : Escreve um ponto flutuante na tela, em notação científica
  - Exemplo: printf ("%e", 10.02545) imprime 1.002545e+01
- %< tamanho >.< decimais >f : Escreve um ponto flutuante na tela, com tamanho < tamanho > e < decimais > casas decimais
  - O ponto, utilizado para separar a parte inteira da decimal, também conta no tamanho
  - Exemplo: printf ("%6.2f", 10.0) imprime < espaço >10.00
- A letra f pode ser substituída pelas letras lf, para escrever um double ao invés de um float
  - Exemplo: printf ("%6.2lf", 10.0) imprime < espaço >10.00



Máscaras

Código	Função
%c	Lê um único caractere
%s	Lê uma série de caracteres
%d	Lê um número decimal
%u	Lê um decimal sem sinal
%l	Lê um inteiro longo
%f	Lê um número em ponto flutuante
%lf	Lê um double

Escrevendo Palavras

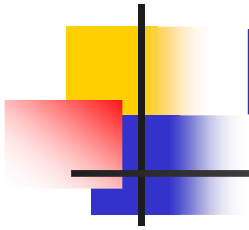


- %c — Escreve uma letra
  - Exemplo: printf ("%c", 'A') imprime "a"
- Note que printf ("%c", 65) também imprime a letra A
- %s — Escreve uma string
  - Exemplo: printf ("%s", "Meu primeiro programa")
  - imprime Meu primeiro programa

Lendo Dados

- scanf
  - Realiza a leitura de um texto a partir do teclado
  - Parâmetros
    - Uma string, indicando os tipos das variáveis que serão lidas e o formato dessa leitura
    - Uma lista de variáveis
- O comando aguarda que o usuário digite um valor e atribui o valor digitado à variável. Exemplo:

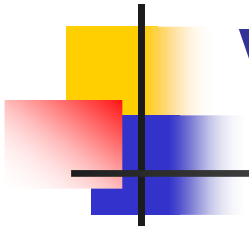
```
#include <stdio.h>
main(){
int n;
printf("Digite um numero: ");
scanf("%d",&n);
printf("O valor digitado foi %d\n",n);
}
```



Lendo Dados (cont.)

- O programa apresentado anteriormente é composto de quatro passos:

1. Cria uma variável n;
2. Escreve na tela Digite um numero;
3. Lê o valor do numero digitado;
4. Imprime o valor do numero digitado.



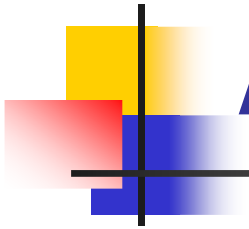
Variáveis na Memória

- Toda variável tem um endereço de memória associado a ela
- Esse endereço é o local onde essa variável é armazenada no sistema
- Normalmente, o endereço das variáveis não são conhecidos quando o programa é escrito
- O operador & retorna o endereço de uma determinada variável
  - Ex: printf ("%d", &valor); imprime o endereço da variável valor
- É necessário usar o operador & no comando scanf, pois esse operador indica que o valor digitado deve ser colocado no endereço referente a uma variável
- Esquecer de colocar o & comercial é um erro muito comum que pode ocasionar erros de execução

Lendo Dados (cont.)

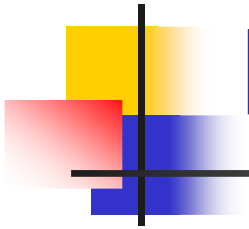
- Leitura de várias variáveis. Exemplo:

```
#include <stdio.h>
void main(){
int m, n, o;
printf("Digite tres numeros: ");
scanf("%d %d %d",&m, &n, &o);
printf("O valores digitados foram\%d %d %d\n",
m, n, o);
}
```



Atribuição

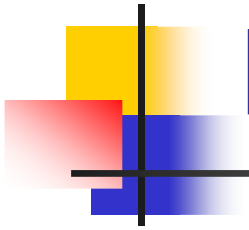
- Atribuir um valor de uma expressão a uma variável significa calcular o valor daquela expressão e copiar aquele valor para uma determinada variável
  - Ex.: soma = a + b;
- No exemplo, a variável soma recebe o valor calculado da expressão a + b
- O operador de atribuição é o sinal de igual (=)
  - A esquerda do operador de atribuição deve existir somente o nome de uma variável
  - À direita, deve haver uma expressão cujo valor será calculado e armazenado na variável



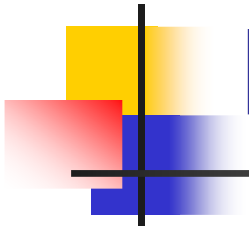
Expressões

- Uma expressão é um conjunto de operações aritméticas, lógicas ou relacionais utilizados para fazer "cálculos" sobre os valores das variáveis
  - Exemplo: a + b Calcula a soma de a e b
- Uma constante é uma expressão e como tal, pode ser atribuída a uma variável
  - Exemplo: a = 10;
- Uma variável também é uma expressão
  - Exemplo: a = b;

Expressões: exemplos



- < expressão > + < expressão >: Calcula a soma de duas expressões
  - Exemplo: a = a + b;
- < expressão > - < expressão >: Calcula a subtração de duas expressões
  - Exemplo: a = a - b;
- < expressão > * < expressão >: Calcula o produto de duas expressões
  - Exemplo: a = a * b;
- < expressão > / < expressão >: Calcula o quociente de duas expressões
  - Exemplo: a = a / b;
- < expressão > % < expressão >: Calcula o resto da divisão (inteira) de duas expressões
  - Exemplo: a = a % b;
- - < expressão >: Calcula o oposto da expressão
  - Exemplo: a = -b;



Precedência de Operações

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado
- Em C, os operadores são calculados na seguinte ordem
  - * e /, na ordem em que aparecerem na expressão
  - %
  - + e -, na ordem em que aparecerem na expressão
- OBS.: As precedências não são obedecidas caso se use parênteses para agrupar uma expressão

Incremento e Decremento

- Possuem duas funções: servem como uma expressão e incrementam ou decrementam o valor da variável ao qual estão associados em uma unidade
  - Exemplo: c++ incrementa o valor da variável c em uma unidade
- Dependendo da posição do operador de incremento e decremento, uma função é executada antes da outra
  - Operador à direita da variável: Primeiro a variável é incrementada, depois a expressão retorna o valor da expressão

```
#include <stdio.h>
void main () {
int a = 10;
printf ("%d", ++a);
}
```

  - O programa acima Imprime 11

Incremento e Decremento

- Operador à esquerda da variável: Primeiro a variável é retornada e depois incrementada

```
#include <stdio.h>
void main () {
int a = 10;
printf ("%d", a++);
}
```

- O programa acima imprime 10

Atribuições Simplificadas

- Uma expressão da forma a = a + b onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada como a += b

Comando	Exemplo	Corresponde a:
+=	a += b	a = a + b;
-=	a -= b	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;