Programação de Computadores I

Introdução ao C

PROFESSORA CINTIA CAETANO

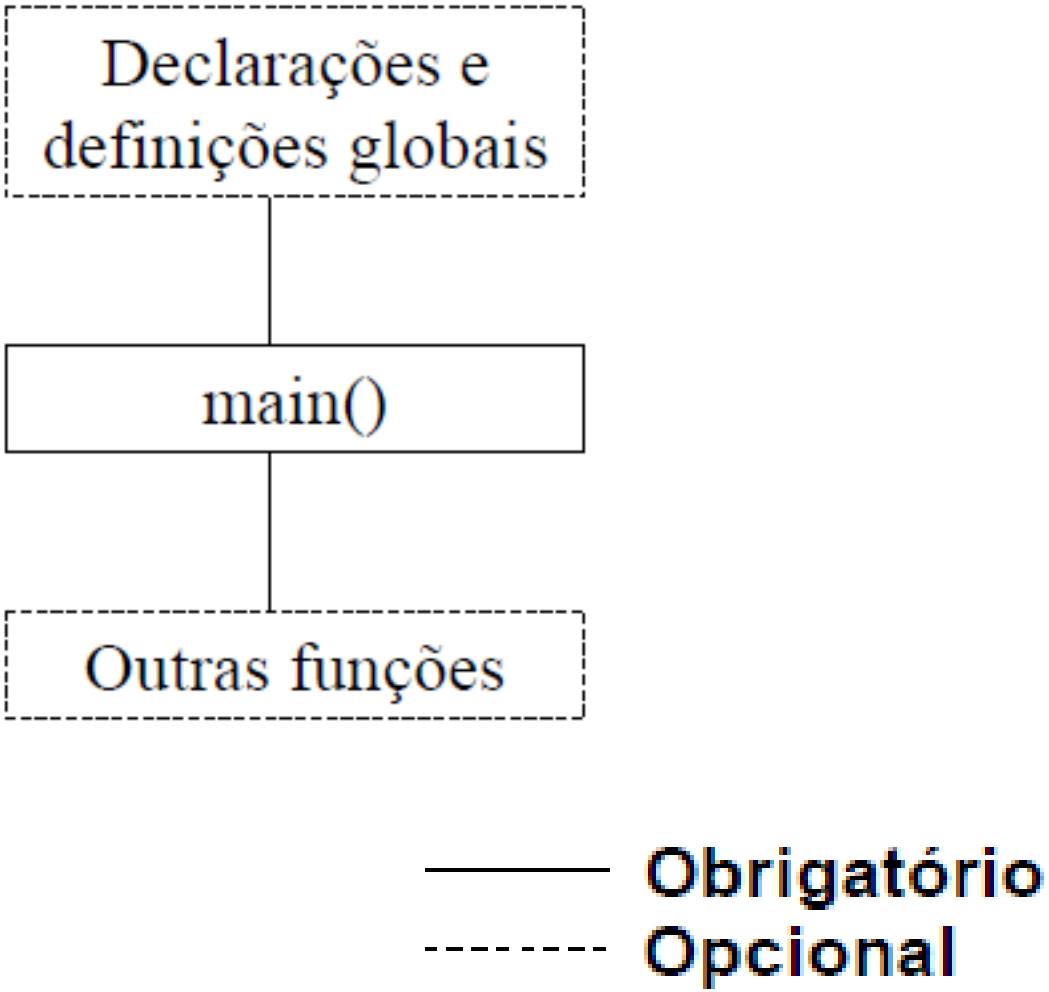
Introdução

- Criada em 1972, por Dennis Ritchie;
- Centro de Pesquisas da Bell Laboratories;
- Para utilização no S.O. UNIX;
- C é uma linguagem de propósito geral;
- Em 1989 o **Instituto Norte-Americano de Padrões** (**ANSI**) padronizou a linguagem C.

Introdução

- Case Sensitive – existe diferença entre maiúsculas e minúsculas;
- Tipos de dados primitivos: caracter, inteiro e real;
- Possui estruturas de controle de fluxo;
- Operadores aritméticos, lógicos, relacionais e condicional;
- Todo programa tem uma função principal chamada **main();**
- Todo linha de instrução em um programa é finalizada com um “**;**”

Estrutura de um programa em C



Palavras-reservadas

Palavras chaves em C (Padrão ANSI)			
auto	Double	int	Struct
break	Else	long	Switch
case	Enum	register	typedef
char	Extern	return	union
const	Float	short	unsigned
continue	For	signed	void
default	Goto	sizeof	volatile
do	If	static	while

Definição de Variáveis

- Devem ser declaradas no **início** do programa ou do sub bloco;
- Podem ser classificadas como **Locais** ou **Globais.**
  - **Locais**
    - Declaradas dentro de funções;
    - Utilizada apenas dentro do escopo da função;
    - O escopo de uma função é determinado por abre-chaves "{" e termina em fecha-chaves "}";
    - **Só existem** no momento que sua função está em execução.
  - **Globais**
    - Declaradas **fora** de todas as funções;
    - Podem ser **acessadas de qualquer parte** do programa;
    - **Existem durante toda a execução do programa**.

Nomes de Variáveis

- Deve conter um ou mais caracteres;
- O primeiro caractere **sempre** deve ser uma **letra**;
- Os caracteres **subseqüentes** podem ser **letras**, **números** ou "_";
- Não pode ser igual às **palavras-chaves**;
- Não pode ter o **mesmo nome de funções;**

Correto	Incorreto
Soma1	1soma
soma	soma!
area_triangulo	area...triangulo

- Obs: as variáveis **"soma"** e **"Soma"** são **distintas**

Declarando variáveis

Sintaxe

- **<Tipo de dados> Nome_variável**;

Ex:

char nome;

int idade;

int total;

Atribuindo valor

- **Nome_da_variavel** = expressão;

Ex:

nome = 'Joao';

idade = 18;

total = 10 + 20;

Operadores aritméticos

Operador Binário	Descrição
=	Atribuição
+	Soma
-	Subtração
/	Divisão
%	Modulo (**resto** da divisão)

Operadores aritméticos Unários e Binários

- **Unários** (**+**, **-**, **++**, **--**) agem sobre **uma variável** apenas, **modificando ou não** o seu valor, e retornam o valor final da variável.
  - **a = -b;**
  - **a++;** *(ou seja)* **a = a+1;**
  - **a--;** *(ou seja)* **a = a-1;**
  - Obs: operador "**–**" como troca de sinal é um operador unário que não altera a variável sobre a qual é aplicado, pois ele retorna o valor da variável multiplicado por **-1**.
- **Binários** (**+**, **-**, *, /, %) usam **duas variáveis** e retornam um terceiro valor, **sem modificar as variáveis originais**.

Operadores aritméticos - Precedência

Operadores	*Sentido de Execução*
( ) [ ]	Esquerda p/ Direita
++ -- - (unário) (cast)	Direita p/ Esquerda
* / %	Esquerda p/ Direita
+ - (binário)	
< > <= >=	
== !=	
!	
&&	
||	
?: (expressão condicional)	
= += *= /= -=	Direita p/ Esquerda

Operadores de Atribuição

- =, +=, -=, *=, /=, %=

Instrução normal	Instrução reduzida
var = var + expr;	Var += expr;
var = var – expr;	Var -= expr;
var = var / expr;	Var /= expr;
var = var * expr;	Var *= expr;

- Exemplos:
  - a = 5;
  - a += 5; *(ou seja)* a = (a + 5);
  - a -= 5; *(ou seja)* a = (a – 5);

Comentários

- // Meu comentário em uma linha

- /* Meu comentário através de um bloco de texto que pode estar em *n* linhas */

Arquivos de Cabeçalho (bibliotecas)

ARQUIVO	FINALIDADE
assert.h	Contém a macro "assert" que é usada para incluir diagnóstico em programas
ctype.h	Declara funções para testar caracteres
errno.h	Define o número de erro do sistema
float.h	Define os valores em ponto flutuante específicos do compilador
limits.h	Define tipos de valores-limites específicos do compilador
locale.h	Utilizada para adaptar as diferentes convenções culturais
math.h	Define as funções matemáticas e macros usadas para operações matemáticas em linguagem C
setjmp.h	Provê uma forma de evitar a seqüência normal de chamada e retorno de função profundamente aninhada
signal.h	Provê facilidades para manipular condições excepcionais que surgem durante a execução, como um sinal de interrupção de uma fonte externa ou um uso na execução
stdarg.h	Define macros para acessar argumentos quando uma função usa um número variável de argumentos
stddef.h	Define funções, constantes, macros e estruturas usadas para operações padrões de entrada e saída
stdio.h	Contém funções, tipos e macros de entrada e saída
stdlib.h	Contém funções para conversão de números, alocação de memória e tarefas similares
string.h	Define funções para manipulação de *"strings"*
time.h	Define funções e estruturas usadas na manipulação de data e hora.

Tipos Primitivos

Inteiro

- Definido pela palavra reservada **int**;
- Ocupa 16 bits (2 bytes)
- Faixa de valores: -32768 à 32767
- Exemplo:
  - int num;
  - num = -73;

Tipos Primitivos

Ponto flutuante

- Definido pela palavra reservada **float**
- Ocupa 4 bytes
- Exemplo:
  - **float** a,b,c=2.34;

Ponto flutuante de precisão dupla

- Definido pela palavra reservada **double**
- Ocupa 8 bytes
- Exemplo:
  - **double** x=2.38, y=3.1415;

Tipos Primitivos

Caractere

- Definido pela palavra reservada **char**;
- Ocupa 8 bits (1 byte)
- Faixa de valores: -128 à 127
- Exemplo:
  - **char** letra;
  - letra = 'A';

Tipos de Dados - Padrão ANSI

Tipo	Num de bits	Formato para leitura com scanf	Intervalo	
			Inicio	Fim
Char	8	%c	-128	127
unsigned char	8	%c	0	255
Signed char	8	%c	-128	127
Int	16	%i ou %d	-32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i ou %d	-32.768	32.767
Short int	16	%hi ou %hd	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi ou %hd	-32.768	32.767
Long int	32	%li ou %ld	-2.147.483.648	2.147.483.647
signed long int	32	%li ou %ld	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
Float*	32	%f, %e ou %g	3,4E-38	3.4E+38
Double*	64	%lf, %le ou %lg	1,7E-308	1,7E+308
long double*	80	%Lf	3,4E-4932	3,4E+4932

** É importante observar que os intervalos de ponto flutuante, na tabela acima, estão indicados em faixa de expoente, mas os números podem assumir valores tanto positivos quanto negativos.*

Estrutura básica de um programa em C

```
#include <stdio.h>
void main()
{
   printf("Ola Mundo!");
}
```

Estrutura básica de um programa em C

- A linha #include <stdio.h> diz ao compilador que ele deve incluir o arquivo-cabeçalho stdio.h.
- Neste arquivo existem declarações de funções úteis para entrada e saída de dados (std = standard, padrão em inglês; io = Input/Output, entrada e saída ==> stdio = Entrada e saída padronizadas).
- Toda vez que você quiser usar uma destas funções deve-se incluir este comando.

Estrutura básica de um programa em C

- A linha void main() indica que estamos definindo uma função de nome main vazia.
- Todos os programas em C têm que ter uma função main, pois é esta função que será chamada quando o programa for executado.
- O conteúdo da função é delimitado por chaves { }.
- O código que estiver dentro das chaves será executado sequencialmente quando a função for chamada.

Estrutura básica de um programa em C

- A linha printf() é uma função que passa uma string (seqüência de caracteres) como argumento.
- É por causa do uso da função printf() pelo programa que devemos incluir o arquivo-cabeçalho stdio.h]
- A função printf() irá apenas colocar a string na tela.
- É importante observar também que os comandos do C terminam com ; (ponto e vírgula)