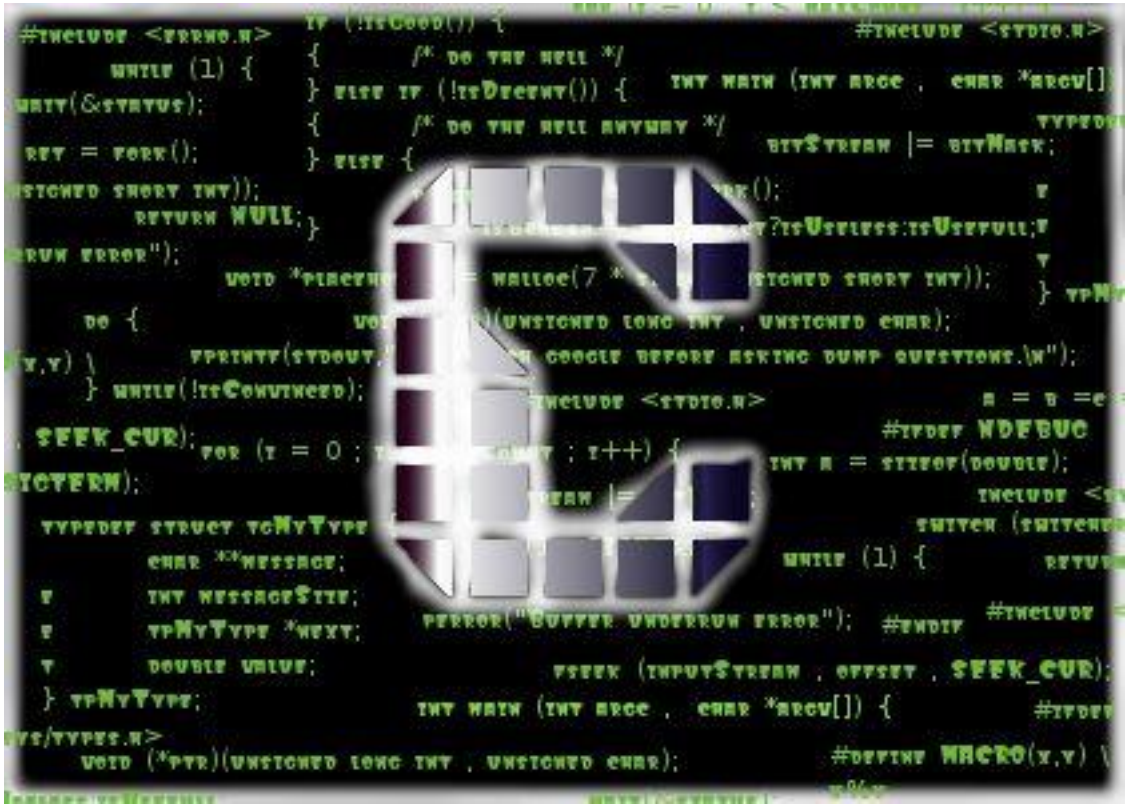
Introdução à linguagem C



Ciclo de desenvolvimento de um programa

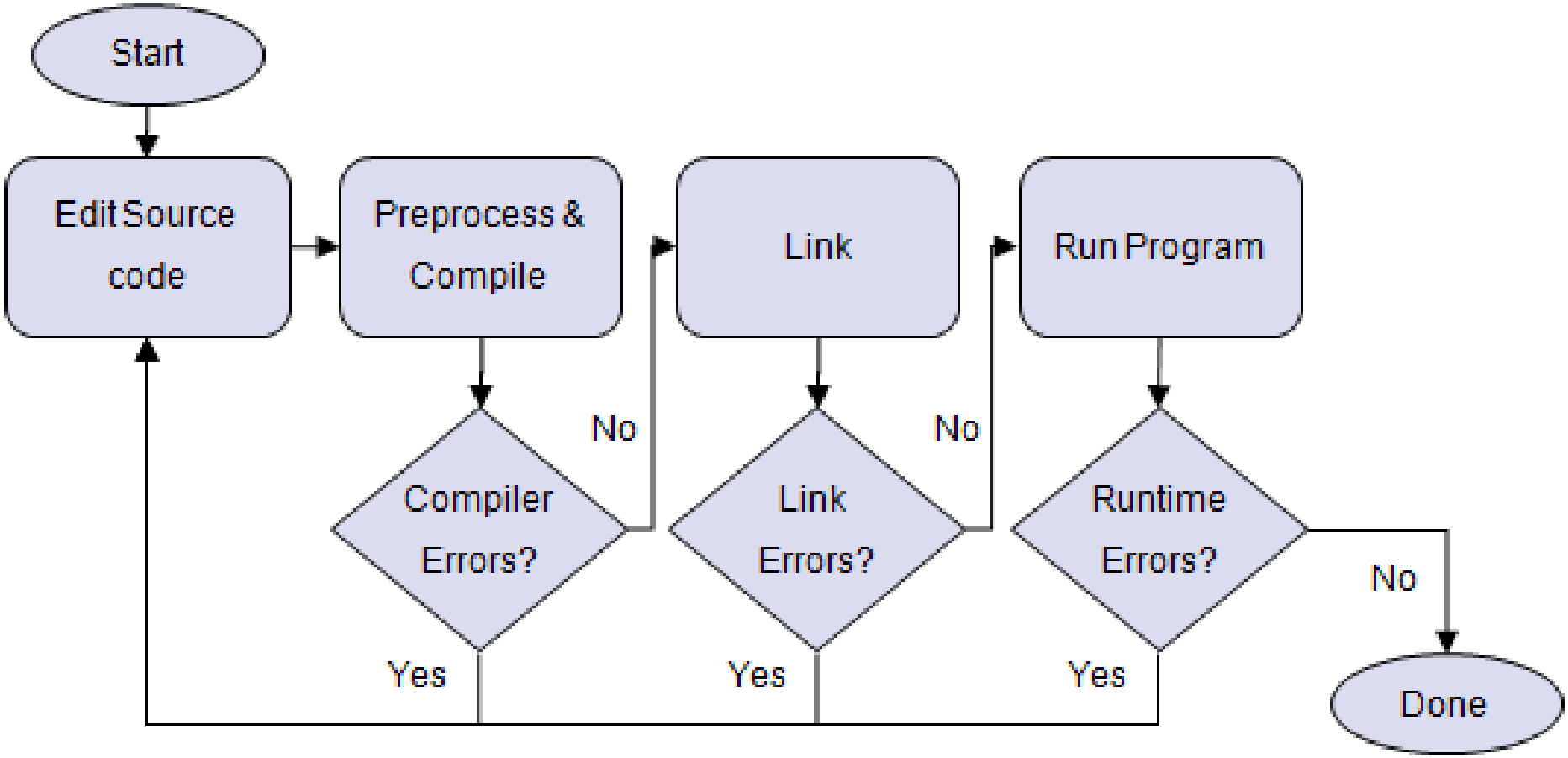


Imagem: http://w3processing.com/index.php?subMenuItemId=222

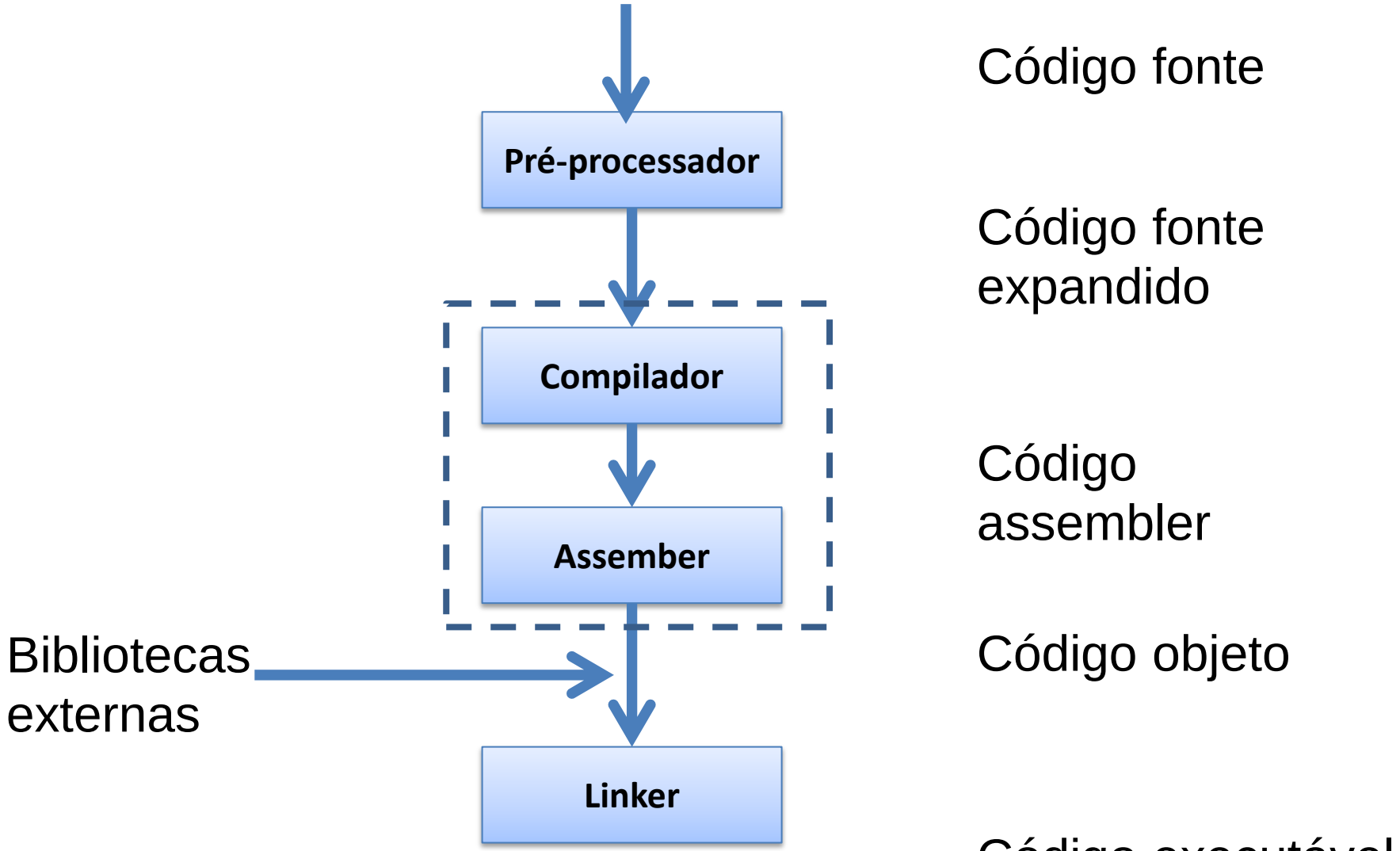
Como funciona um programa?

- MONTADOR (*assembler*)
  - Tradutor para linguagens de 2ª geração.
- COMPILADOR:
  - Traduz todo o programa de uma vez.
- INTERPRETADOR:
  - Traduz o programa instrução por instrução.

Compilação em C

- Um programa em C é composto por um ou mais ficheiros de código fonte (.c e os respetivos .h)
- Cada ficheiro é processado primeiro pelo pré-processador originando um ficheiro de texto
- O ficheiro de texto é processado pelo compilador (i.e., é compilado) originando um ficheiro de código objeto (assembler)
- Todos os ficheiros objetos são agregados pelo “linker”, criando-se o ficheiro executável

Modelo de compilação da linguagem C



História da Linguagem C

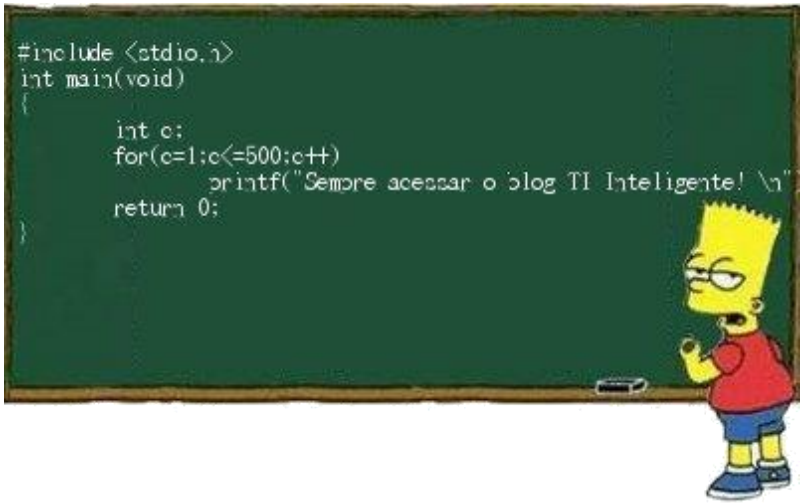
- Surgiu no início dos anos 70
- Criada inicialmente para o UNIX
- Criadores:
  - Dennis Ritchie (direita)
  - Kenneth Thompson (esquerda)
- Baseada na Linguagem B
- Versão inicial bastante simples

História da Linguagem C

- Ampla popularização nos anos 80
- Muitas arquiteturas e compiladores
- Problemas com a incompatibilidade
- Padronização de 82 a 89 (C ANSI)
- Até hoje existem problemas entre os diversos compiladores e sistemas operacionais

Caraterísticas

- Paradigma Procedural (Procedimentos)
- Flexível
- Alta performance
- Poucas restrições
- Ótima iteração com:
  - Sistemas Operacionais
  - Dispositivos de Hardware
  - Outras Linguagens



Palavras Reservadas

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

Obs.: C é case sensitive

Estrutura básica de um programa C

```
diretivas para o pré-processador
declaração de variáveis globais
main ()
{
   declaração de variáveis locais da função main
   comandos da função main
}
```

Diretivas para o processador - Bibliotecas

- Diretiva `#include` permite incluir uma biblioteca
- Bibliotecas contêm funções pré-definidas, utilizadas nos programas

Exemplos:

`#include <stdio.h>`	Funções de entrada e saída
`#include <stdlib.h>`	Funções padrão
`#include <math.h>`	Funções matemáticas
`#include <string.h>`	Funções de texto

O ambiente Dev-C++

- O Dev-C++ é um ambiente de desenvolvimento de programas em C e C++ (editor, compilador, bibliotecas...)

- Pode ser obtido (download) em http://www.bloodshed.net/devcpp.html

Usando o Dev-C++

- Iniciar o Dev-C++
- Criar um novo arquivo, com o comando:
  - ***File > New Source File***
- Editar o programa seguinte:

```
#include <stdio.h>
main()
{
  printf ("Hello world!");
}
```

- Guardar o programa com o nome **1programa.c**
- Compilar o programa com o comando
  - **Execute** > **Compile**
  - ou com a tecla **Ctrl-F9**

- Se houver algum erro de sintaxe, aparece uma ou mais mensagens no rodapé da janela. Neste caso, corrija o programa e repita.
- Se não houver erros, execute o programa com o comando
  - **Executar** > **Executar** ou com a tecla **Ctrl-F10**

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  printf ("Hello world!\n");
  system("PAUSE");
}
```

Dicas

- Terminar todas as linhas com ;
- Sempre salvar o programa antes de o compilar
- Sempre compilar o programa antes de o executar
- Quando ocorrer um erro de compilação, dê um duplo clique sobre a mensagem de erro para destacar o comando errado no programa
- Verificar também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o ;
- Usar comentários, iniciados por //

Declarações

- Declaram as variáveis e seus tipos
- Os nomes das variáveis devem conter apenas letras, dígitos e o símbolo _
- Os principais tipos são: **int**, **float**, **double** e **char**

Exemplos:

```
int n;
int quantidade_valores;
float x, y, somaValores;
char sexo;
char nome[40];
```

Declarações

Algoritmo

Var n1, n2, n3, media: real

Na Linguagem C...

```
#include <stdio.h>

main()
{
    double n1, n2, n3, media;


}
```

Tipos

- São as formas que utilizamos para representar dados
- C possui 5 tipos básicos:

  `char, int, float, double e void`

- E 4 modificadores básicos:

  `signed, unsigned, long e short`

Os 4 podem ser aplicados ao **`int`**

`long` pode ser aplicado ao **`double`**

`signed` e **`unsigned`** aplicados ao **`char`**

int

O tipo de dado **int** (inteiro) serve para armazenar valores numéricos inteiros.

- Existem vários tipos de inteiros, cada um de um tamanho diferente (dependendo do sistema operacional e/ou arquitetura do processador):

  - **int**, pode possuir 16 bits, 32 bits ou 64 bits
  - **short int**, deve possuir tamanho de no mínimo 16 bits e não pode ser maior que *int*
  - **long int**, deve possuir tamanho mínimo de 32 bits
  - **long long int**, deve possuir tamanho mínimo de 64 bits

float

O tipo de dado **float** serve para armazenar números de ponto flutuante, ou seja, com casas decimais.

double

O tipo de dado **double** serve para armazenar números de ponto flutuante de dupla precisão, normalmente tem o dobro do tamanho do *float* e portanto o dobro da capacidade.

char

O tipo **char** ocupa 1 byte, e serve para armazenar carateres ou inteiros.

- Isto significa que o programa reserva um espaço de 8 bits na memória RAM ou em registos do processador para armazenar um valor (*char* de tamanho maior que 8 bits é permitido pela linguagem C, mas os casos são raros).
- Com vetores do tipo *char* é possível criar cadeias de carateres (**strings**).

Em C as variáveis podem ser inicializadas com um valor quando são declaradas.

Exemplo:

```
int cont = 10;
```

O operador = (i.e., igual) usado para afetar valores a variáveis.

Exemplo:

```
cont = 32;
letra = 'A';
```

Em C tanto as variáveis **float** como as **double** são impressas com seis casas decimais.

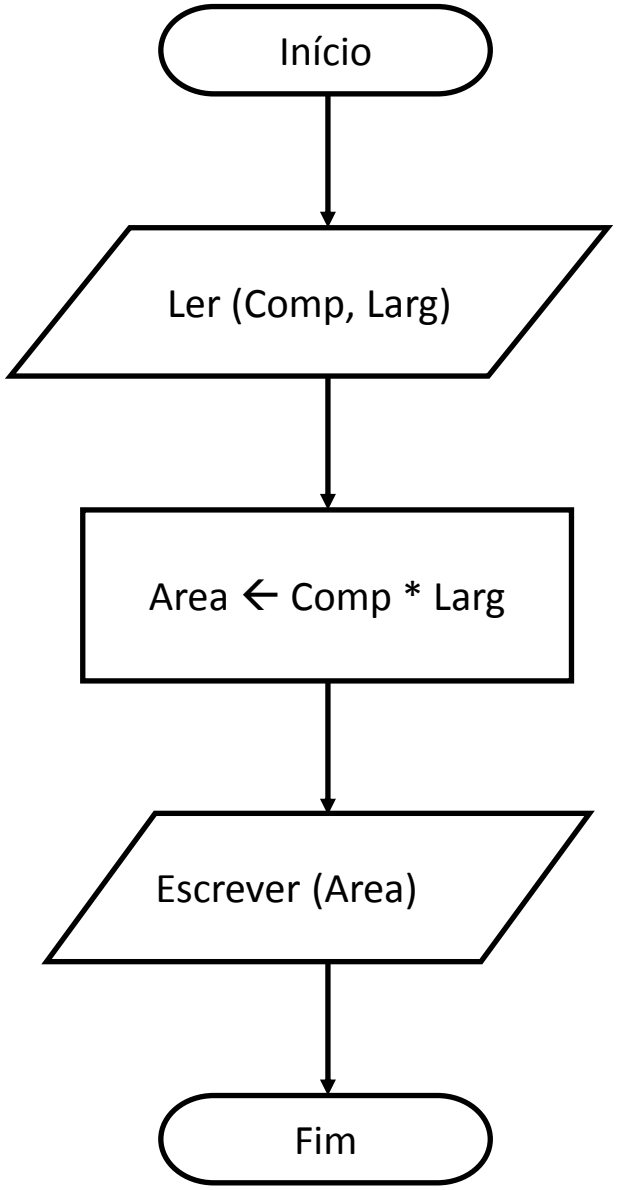
Isto NÃO está relacionado com a precisão com que o número é armazenado internamente, apenas significa quantas casas decimais o **printf()** usa para imprimir estes tipos de variáveis.

O programa seguinte ilustra como se declaram e como se imprimem os diferentes tipos de dados:

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

main()
{
      int      soma = 100;
      char     letra = 'Z';
      float    set1 = 23.567;
      double   num2 = 11e+23;

      printf("Variável inteira: %d\n", soma);
      printf("Variável caracter: %c\n", letra);
      printf("Variável float: %f\n", set1);
      printf("Variável double: %e\n", num2);

      system("pause");
}
```

ToDo:

Dado o seguinte fluxograma para o cálculo de uma área, escrever o respetivo programa em C.



Constantes

Constantes são o oposto das variáveis, apesar de trabalharem da mesma forma.

As variáveis têm esse nome exatamente porque podemos mudar o seu valor durante a execução do programa.
Já as constantes são dados gravados em memória que de forma alguma podemos alterar seu valor.

```
#define tipo 'A';
#define numero 56
```

ToDo:

Escrever um programa em C que calcule o perímetro e a área de um círculo, baseado no raio digitado pelo utilizador.

Saber que:

```
Perimetro = 2 * PI * R
Area= PI * R ^ 2)
```

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

#define PI 3.14159265;
int main (void)
{
      float raio;
      double area, perimetro;
      printf ("Digite o raio do seu circulo:\n cm\r");
      scanf ("%f", &raio);
      area = (raio*raio)*PI;
      perimetro = raio*2*PI;
      system ("cls");
      printf ("A area do seu circulo e: %.2f cm\n", area);
      printf ("O perimetro do seu circulo e: %.2f cm",
perimetro);
      printf ("\n\n");

      system ("pause");
}
```

Compila o programa seguinte e elimina os erros de sintaxe existentes. Uma vez compilado com sucesso, executa o programa e verifica o resultado.

```
#include <stdio.h. >
int main(){

  int a, b, c, d;

  printf("Digite o primeiro numero:\n");
  scanf("%d",&a);

  printf("Digite o segundo numero:\n");
  scanf("%d",&b);

  printf("Digite o terceiro numero:\n");
  scanf("%d",&c);

  d = a + b + c
  printf("Resultado: %d",d);
  system("PAUSE ");

  return 0;
```

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

main(){

  int a, b, c, d;

  printf("Digite o primeiro numero:\n");
  scanf("%d",&a);

  printf("Digite o segundo numero:\n");
  scanf("%d",&b);

  printf("Digite o terceiro numero:\n");
  scanf("%d",&c);

  d = a + b + c;
  printf("Resultado: %d\n",d);

  system("PAUSE ");
  return 0;
}
```

Tipos

- Modificadores de acesso:
  - **`const`**: a variável não pode ter o seu valor alterado
  - **`volatile`**: a variável pode ter o seu valor modificado fora do controle do programa
- Classes de Armazenamento:
  - **`auto`**: indica que uma variável é local (opcional), também é usada na declaração de *nested functions*
  - **`extern`**: variável declarada em outro arquivo
  - **`register`**: armazena, se possível, a variável em um registrador na própria CPU.
- Classes de Armazenamento (Cont.):
  - **`static`**: não permite que um módulo externo possa alterar nem ver uma variável global, também é usado para manter o valor de uma variável local em uma função de uma chamada para outra.

Tipos

- O tamanho do inteiro depende da arquitetura do sistema:
  - Sistemas de 32 bits ⇨ inteiro de 32 bits
  - Sistemas de 64 bits ⇨ inteiro de 64 bits

- Restrições:
  - `short int` e `int` devem ter pelo menos 16 bits
  - `long int` com no mínimo 32 bits
  - `short int` <= `int` <= `long int`

Variáveis

- Declaração:

  tipo nome = inicialização;

- Escopo da variáveis:
  - **globais**: podem ser usadas em qualquer lugar do programa
  - **locais**: podem ser usadas apenas na função onde foi declarada

```
int a, b = 10; // Variáveis globais

void f(char c) {
     double d = 10.0; // Variável local
     int i = a; // Variável local
     // ...
}

int main() {
     int i = b;  // Variável local
     return 0;
}
```

Variáveis

- Restrições
  - O nome das variáveis deve começar com uma letra ou um sublinhado "_"
  - Os demais caracteres podem ser letras, números ou sublinhado
  - O nome da variável não pode ser igual a uma palavra reservada e aos nomes das funções
  - Tamanho máximo para o nome de uma variável:
    - 32 caracteres

Constantes

- São valores que são mantidos fixos pelo compilador
- Também podem ser:
  - Octais - 0NUMERO_OCTAL
  - Hexadecimais - 0xNUMERO_HEXADECIMAL
- Exemplos:
  - `'\n'` (caractere), `"C++"` (string), `10` (inteiro), `15.0` (float), `0xEF` (239 em decimal), `03212` (1674 em decimal)

Constantes de Barra Invertida

Código	Significado
\b	Retrocesso (backspace)
\f	Alimentação de formulário (form feed)
\t	Tabulação Horizontal (tab)
\n	Nova Linha
\"	Aspas
\'	Apóstrofo
\0	Nulo
\\	Barra Invertida
\a	Sinal Sonoro (Beep)
\N	Constante Octal (N é o valor da constante)
\xN	Constante Hexadecimal (N é o valor da constante)

Operadores Aritméticos

Operador	Ação
+	Soma
–	Subtração ou troca de sinal
*	Multiplicação
/	Divisão
%	Resto da divisão inteira
++	Incremento
– –	Decremento

Exercícios

1) Qual o valor das variáveis x, y e z após o seguinte segmento de código:

```
int x, y, z;
x = y = 10;
z = ++x;
x = -x;
y++;
x = x + y - (z--);
```

2) Utilize o DEVCPP para compilar e executar o código do exercício anterior.

Operadores Relacionais

Operador	Relação
>	Maior que
>=	Maior que ou igual a
<	Menor que
<=	Menor que ou igual a
==	Igual a
!=	Diferente de

Os operadores relacionais do C realizam ***comparações*** entre variáveis.

Operadores Lógicos

Operador	Função
&&	AND
||	OR
!	NOT

Os operadores relacionais retornam verdadeiro (1) ou falso (0). Para fazer ***operações com valores lógicos*** (verdadeiro e falso) temos ***os operadores lógicos***:

Operadores Lógicos Bit a Bit

Operador	Ação
&	AND Lógico
|	OR Lógico
^	XOR (OR exclusivo)
~	NOT
>>	Shift Rigth
<<	Shift Left

O C permite que se façam ***operações lógicas "bit-a- bit"*** em números. Ou seja, neste caso, o número é representado na sua forma binária e as operações são feitas em cada um dos bit.

Entrada e Saída de Dados

Entrada de Dados

- Função **scanf**

```
scanf ("formatos", &var1, &var2,...)
```

Exemplos:

```
int i, j;
float x;
char c;
scanf("%d", &i);
scanf("%d %f", &j, &x);
scanf("%c", &c);
scanf("%s", nome);
```

%d	inteiro
%f	float
%lf	double
%c	char
%s	palavra

Algoritmo

```
var n1, n2, n3, media: real

  ler n1
  ler n2
  ler n3

  ler n1, n2, n3
```

Na Linguagem C...

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   double n1, n2, n3, media;
   scanf ("%lf",&n1);
   scanf ("%lf",&n2);
   scanf ("%lf",&n3);
   scanf ("%lf %lf %lf",&n1,&n2,&n3);
system("PAUSE");
}
```

Operadores Matemáticos

Operador	Exemplo	Comentário
+	`x + y`	Soma x e y
-	`x - y`	Subtrai y de x
*	`x * y`	Multiplica x e y
/	`x / y`	Divide x por y
%	`x % y`	Resto da divisão de x por y
++	`x++`	Incrementa em 1 o valor de x
--	`x--`	Decrementa em 1 o valor de x

Algoritmo

```
var n1, n2, n3, media: real

ler n1, n2, n3

media=(n1+n2+n3)/3
```

Na Linguagem C...

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   double n1, n2, n3, media;
scanf ("%lf %lf %lf",&n1,&n2,&n3);
media=(n1+n2+n3)/3;

system("PAUSE");
}
```

Saída de Dados

- Função **printf**

```
printf ("formatos", var1, var2,...)
```

Exemplos:

```
int i, j;
float x;
char c;
printf("%d", i);
printf("%d, %f", j, x);
printf("%c", c);
printf("%s", nome);
```

%d	inteiro
%f	float
%lf	double
%c	char
%s	palavra

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   int i, j;
  float x;
  char c;
  char nome [10];
  i = 1; j = 2; x = 3;
  printf("%d", i);
  printf("%d, %f", j, x);

 system("PAUSE");
}
```

```
#include <stdio.h>
#include <stdlib.h>
main()
{
      // definicao de variaveis
      double n1, n2, n3, media;
     // entrada de dados
      scanf ("%lf %lf %lf",&n1, &n2, &n3);
    // operacao
      media=(n1+n2+n3)/3;
   // saida de dados
      printf("%f", n1);
      printf("%f",  n2);
      printf("%f", n3);
      printf("%f", media);

      system("PAUSE");
}
```

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    // definicao de variaveis
        int n;
     int quantidade_valores;
     float x, y, somaValores;
     char sexo;
     char nome[40];
     x = 1; y = 2;
     //atribuicao
    somaValores = x + y;
    // mostra mensagem de texto na tela
    printf(“Olá Mundo!");
    // mostra valor na tela
    printf("somaValores=%f", somaValores);
  system("PAUSE");
}
```

1) Tendo como dados de entrada a altura e o sexo de uma pessoa, construa um programa que calcule seu peso ideal, utilizando a seguinte fórmula: **peso ideal = (72.7*h) – 58**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main()
{
  float altura, pesoIdeal;
  char sexo;
  printf ("\nDigite sua altura em metros: ");
  scanf ("%f",&altura);
  printf ("\nDigite seu sexo (F/M): ");
  sexo=getche();
  pesoIdeal=(72.7*altura)-58;
  printf ("\nO sexo digitado foi %c e o peso ideal da
    pessoa eh %.2f quilos\n", sexo, pesoIdeal);
      system("PAUSE");
}
```

Operadores de Atribuição

Operador	Exemplo	Comentário
=	x = y	Atribui o valor de y a x
+=	x += y	Equivale a x = x + y
-=	x -= y	Equivale a x = x – y
*=	x *= y	Equivale a x = x * y
/=	x /= y	Equivale a x = x / y
%=	x %= y	Equivale a x = x % y

Funções Matemáticas

```
#include <math.h>
```

Função	Exemplo	Comentário
`ceil`	`ceil(x)`	Arredonda o número real para cima; ceil(3.2) é 4
`cos`	`cos(x)`	Cosseno de x (x em radianos)
`exp`	`exp(x)`	**e** elevado à potencia x
`fabs`	`fabs(x)`	Valor absoluto de x
`floor`	`floor(x)`	Arredonda o número deal para baixo; floor(3.2) é 3
`log`	`log(x)`	Logaritmo natural de x
`log10`	`log10(x)`	Logaritmo decimal de x
`pow`	`pow(x, y)`	Calcula x elevado à potência y
`sin`	`sin(x)`	Seno de x
`sqrt`	`sqrt(x)`	Raiz quadrada de x
`tan`	`tan(x)`	Tangente de x

2) Construa um programa que tendo como entrada dois pontos quaisquer do plano P(x1,y1) e Q(x2,y2), imprima a distância entre eles. A fórmula da distância é:

$$\sqrt{(x2-x1)^2 + (y2-y1)^2}$$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{

}
```

>> ver passo seguinte

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
  float x1, y1, x2, y2;

  float distancia;

  printf ("\nDigite o valor de x1: ");
  scanf ("%f",&x1);

  printf ("\nDigite o valor de y1: ");
  scanf ("%f",&y1);
```

>> ver passo seguinte

```
printf ("\nDigite o valor de x2: ");
scanf ("%f",&x2);

printf ("\nDigite o valor de y2: ");
scanf ("%f",&y2);

distancia=sqrt(pow((x2-x1),2)+pow((y2-y1),2));


printf ("\nA distancia entre os pontos P1 e P2
  é %.2f\n", distancia);

system("PAUSE");
}
```

>> visualização completa do programa

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
 float x1, y1, x2, y2;
 float distancia;
 printf ("\nDigite o valor de x1: ");
 scanf ("%f",&x1);
 printf ("\nDigite o valor de y1: ");
 scanf ("%f",&y1);
 printf ("\nDigite o valor de x2: ");
 scanf ("%f",&x2);
 printf ("\nDigite o valor de y2: ");
 scanf ("%f",&y2);
 distancia=sqrt(pow((x2-x1),2)+pow((y2-y1),2));
 printf ("\nA distancia entre os pontos P1 e P2 é %.2f\n",
   distancia);
 system("PAUSE");
}
```

3) Construir um programa que calcule a quantidade de latas de tinta necessárias e o custo para pintar tanques cilíndricos de combustível, onde são fornecidos a altura e o raio do cilindro.

Sabendo que:

- a lata de tinta custa 20,00 €
- cada lata contém 5 litros
- cada litro de tinta pinta 3 metros quadrados.

Sabendo que:

Área do cilindro=$3.14*raio^2 + 2*3.14*raio*altura$

e que raio e altura são dados de entrada.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
   float altura,raio,areaCilindro
   float qtdadeLitros,qtdadeLatas,custo;

    printf ("\nDigite o valor da altura em
metros: ");

    scanf ("%f",&altura);
    printf ("\nDigite o valor do raio em metros:
");

    scanf ("%f",&raio);
```

>> ver passo seguinte

```
areaCilindro=3.14*raio*raio + 2*3.14*raio*altura;

printf ("\nA area do cilindro é %.2f metros
  quadrados", areaCilindro);

qtdadeLitros=areaCilindro/3;
printf ("\nA quantidade de litros necessaria é de
  %.2f ", qtdadeLitros);

qtdadeLatas=qtdadeLitros/5;
printf ("\nA quantidade de latas necessaria é de
  %.2f ", qtdadeLatas);

custo=qtdadeLatas*20;
printf ("\nO valor total das tintas é de € %.2f
  \n", custo);

system("PAUSE");
}
```

>> visualização completa do programa

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
float altura,raio,areaCilindro,qtdadeLitros,qtdadeLatas,custo;
printf ("\nDigite o valor da altura em metros: ");
scanf ("%f",&altura);
printf ("\nDigite o valor do raio em metros: ");
scanf ("%f",&raio);
areaCilindro=3.14*raio*raio + 2*3.14*raio*altura;
printf ("\nA area do cilindro é %.2f metros quadrados",
   areaCilindro);
qtdadeLitros=areaCilindro/3;
printf ("\nA quantidade de litros necessaria é de %.2f ",
   qtdadeLitros);
qtdadeLatas=qtdadeLitros/5;
printf ("\nA quantidade de latas necessaria é de %.2f ",
   qtdadeLatas);
custo=qtdadeLatas*20;
printf ("\nO valor total das tintas é de € %.2f \n", custo);
system("PAUSE");
}
```