

Introdução à Programação

AULA 3 – Escrevendo Programas

Prof^a. Glaucia M. M. Campos

glauciamelissa@uern.br

Problema 1

- Considere o seguinte problema:
 - Determinar o valor de $y = \sin(1,5)$

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      float y;
6      y = sin(1.5);
7      printf("y = %f", y);
8      printf("\n");
9      return 0;
10 }
```

Definições (1/4)

- Para resolver um problema de computação é preciso escrever um texto.
- Este texto, como qualquer outro, obedece regras de sintaxe
- Estas regras são estabelecidas por uma linguagem de programação
- Este texto é conhecido como

Programa

Definições (2/4)

- Neste curso, será utilizada a linguagem C
- A linguagem C é o subconjunto da linguagem C++ e, por isso, geralmente, os ambientes de programação da linguagem C são denominados ambientes C/C++
- Um ambiente de programação contém:
 - Editor de Programas: viabiliza a escrita do programa
 - Compilador: verifica se o texto digitado obedece à sintaxe da linguagem de programação e, caso isto ocorra, traduz o texto para uma **linguagem de máquina**

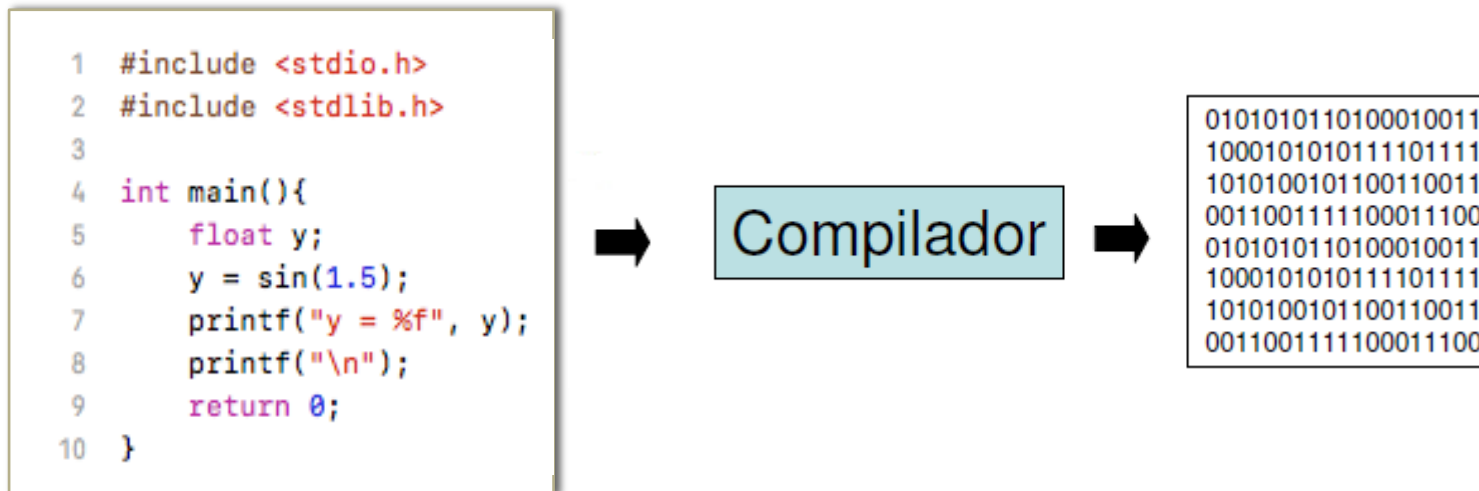
Definições (3/4)

- Que ambiente de programação iremos utilizar?
 - Existem muitos, por exemplo: Microsoft Visual C++, Borland C++ Builder, DEV-C++, Eclipse C/C++
 - Será utilizado neste curso o Visual Studio Code



Definições (4/4)

- Por que o compilador traduz o programa escrito em linguagem de programação para a linguagem de máquina?



- Os computadores atuais só conseguem executar instruções que sejam escritas na forma de códigos binários
- Programa em linguagem de máquina -> executável

Erros de Sintaxe (1/2)

- Atenção!
 - O programa executável somente será gerado se o texto do programa não contiver erros de sintaxe
 - Exemplo: considere uma string. Ah? O que é isso?
 - Uma sequência de caracteres delimitada por aspas
 - Se isso é uma string e se tivéssemos escrito:

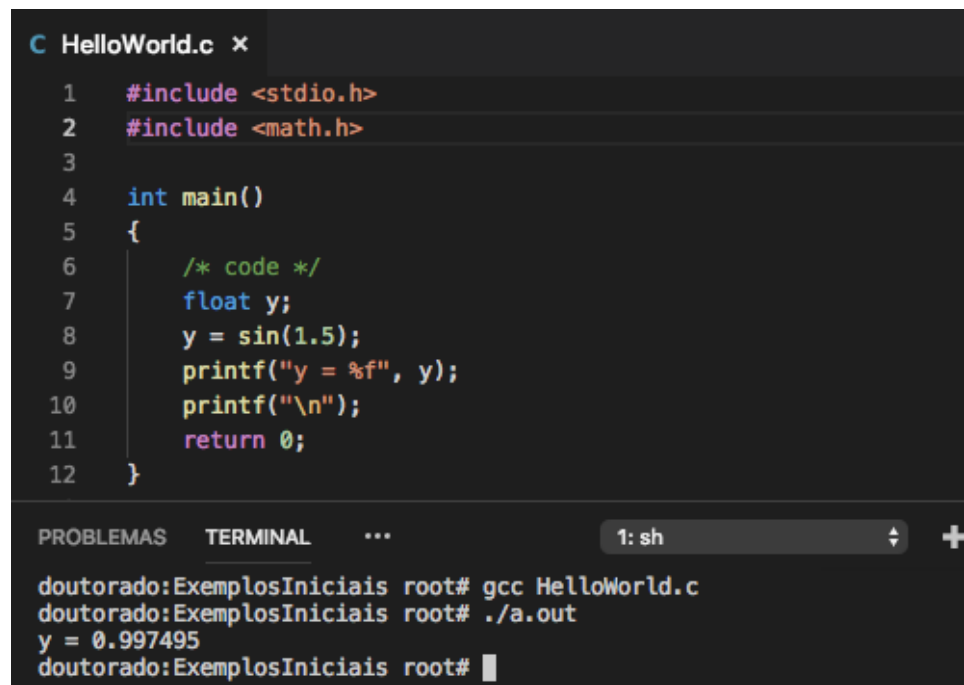
```
printf("y = %f, y);
```

- O compilador iria apontar um erro de sintaxe nesta linha do programa e exibir uma mensagem tal como:

```
undetermined string or character constant
```

Erros de Sintaxe (2/2)

- Se o nome do programa é HelloWorld.c, então após a compilação será produzido o executável a.out
- Executando-se o programa, o resultado será:



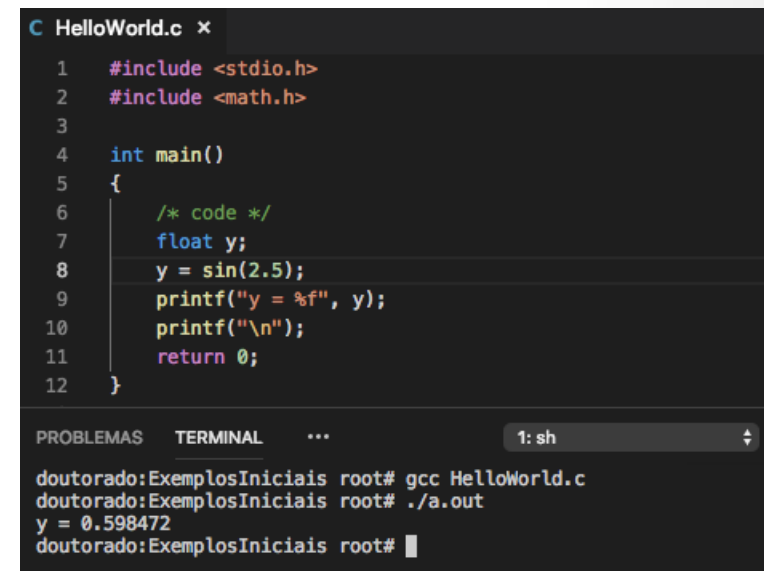
```
C HelloWorld.c x
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      /* code */
7      float y;
8      y = sin(1.5);
9      printf("y = %f", y);
10     printf("\n");
11     return 0;
12 }
```

PROBLEMAS TERMINAL ... 1: sh +

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
y = 0.997495
doutorado:ExemplosIniciais root#
```


Erros de Lógica (1/2)

- Atenção!
 - Não basta obter o programa executável!!! Será que ele está correto?
 - Se ao invés de: `Y = sin(1.5);`
 - Tivéssemos escrito: `Y = sin(2.5);`
 - O compilador também produziria o programa a.out, que executado iria produzir:



The screenshot shows a code editor window titled 'HelloWorld.c' with the following C code:

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      /* code */
7      float y;
8      y = sin(2.5);
9      printf("y = %f", y);
10     printf("\n");
11     return 0;
12 }
```

Below the code editor is a terminal window with the following output:

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
y = 0.598472
doutorado:ExemplosIniciais root#
```

Erros de Lógica (2/2)

- Embora um resultado tenha sido obtido, ele não é correto!
- O processo de identificação e correção de erros de lógica é denominado depuração (debug).
- O nome de um texto escrito em uma linguagem de programação é chamado de programa-fonte.

Arquivos de Cabeçalho (1/3)

- Note que o programa-fonte HelloWorld.c começa com as linhas:

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

- Todo programa-fonte em linguagem C começa com linhas deste tipo.
- O que elas indicam?
 - Dizem ao compilador que o programa-fonte vai utilizar arquivos de cabeçalho (extensão .h, de header).
 - E daí? O que são estes arquivos de cabeçalho?
 - Eles contêm informações que o compilador precisa para construir o programa executável.

Arquivos de Cabeçalho (2/3)

- Como assim?
 - Observe que o programa HelloWorld.c inclui algumas funções, tais como:
 - sin – função matemática seno**
 - printf – função para exibir resultados**
 - Por serem muito utilizadas, a linguagem C mantém funções como estas em bibliotecas.
 - Atenção! O conteúdo de um arquivo de cabeçalho também é um texto.

Arquivos de Cabeçalho (3/3)

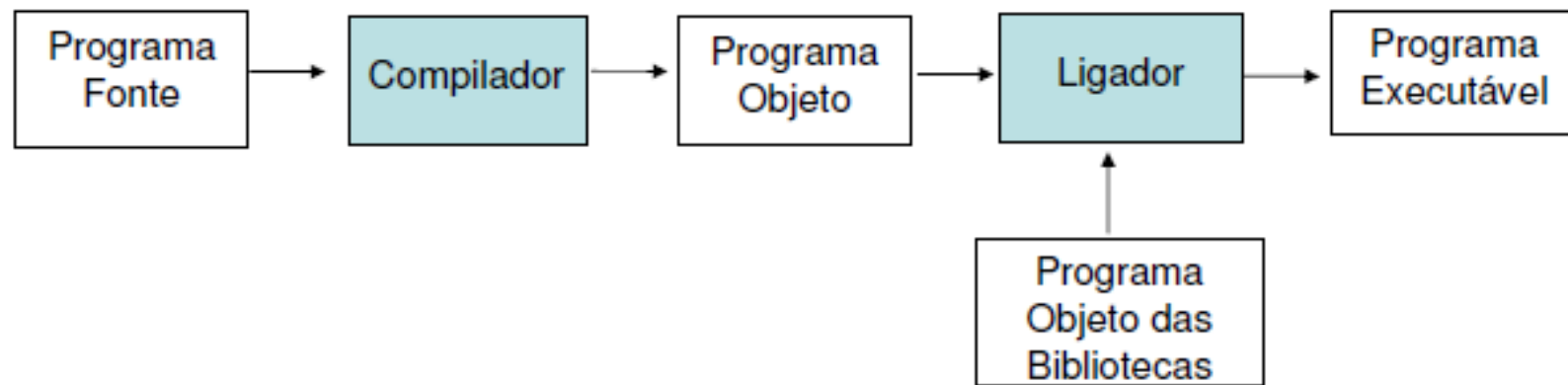
- Ao encontrar uma instrução **#include** em um programa-fonte, o compilador traduz este texto da mesma forma se o texto tivesse sido digitado no programa-fonte.
- Portanto, as linhas:

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

- Indicam ao compilador que o programa HelloWorld.c utilizará as instruções das bibliotecas stdio e math.

Processo de Compilação

- O processo de compilação, na verdade, se dá em duas etapas:
 - Fase de tradução: programa-fonte é transformado em um programa-objeto.
 - Fase de ligação: junta o programa-objeto às instruções necessárias das bibliotecas para produzir o programa executável.

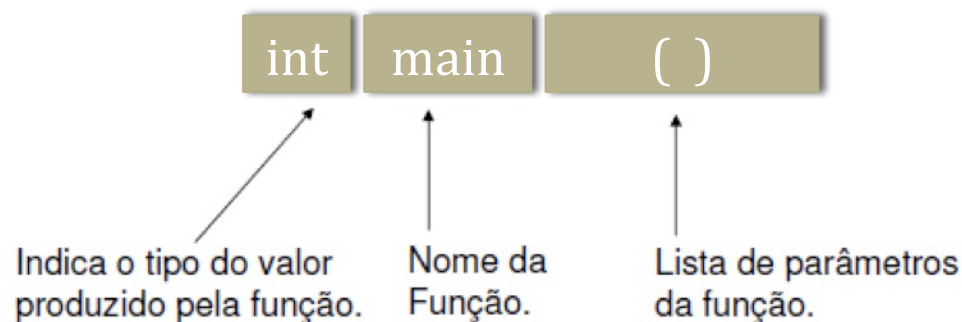


Função main (1/2)

- A próxima linha do programa é:

```
int main ( )
```

- Esta linha corresponde ao cabeçalho da função main (a função principal, daí o nome main).
- O texto de um programa em Linguagem C pode conter muitas outras funções e SEMPRE deverá conter a função main.



Função main (2/2)

- A linguagem C é **case sensitive**. Isto é, considera as letras maiúsculas e minúsculas diferentes.
- Atenção!
 - O nome da função principal deve ser escrito com letras minúsculas: **main**
 - Main ou MAIN, por exemplo, provocam erros de sintaxe
- Da mesma forma, as palavras **int** e **char** devem ser escritas com letras minúsculas

Tipos de Dados (1/3)

- Uma das principais funcionalidades de um computador é a manipulação de informações
- Por isso, é necessário que haja formas de se trabalhar com diferentes tipos de dados em um programa
- A linguagem C dispõe de quatro tipos básicos de dados:

Tipo	Tamanho (bytes)	Valor
char	1	Um caractere (ou um inteiro de 0 a 127).
int	4	Um número inteiro.
float	4	Número de ponto flutuante (SP).
double	8	Número de ponto flutuante (DP).

Tipos de Dados (2/3)

- Tipos de dados básicos acompanhados por modificadores:
 - long ou long int (4 bytes)
 - unsigned char (de 0 a 255)
 - unsigned int (de 0 a 65535)
 - unsigned long

Tipos de Dados (3/3)

- Existem também padrões internacionais para a codificação de caracteres (ASCII, ANSI, Unicode)
- A linguagem C adota o padrão ASCII (American Standard Code for Information Interchange)
 - Código para representar caracteres como números
 - Cada caractere é representado por 1 byte, ou seja, uma sequência de 8 bits
 - Por exemplo:

Caractere	Decimal	ASCII
'A'	65	01000001
'@'	64	01000000
'a'	97	01100001

Constantes

- Uma constante tem valor fixo e inalterável
- Em C, uma constante caractere é escrita entre aspas simples, uma constante cadeia de caracteres entre aspas duplas e constantes numéricas com o próprio número

```
int main( ){  
    printf ("Este é o número dois: %d", 2);  
    return 0;  
}
```

Variáveis (1/5)

- Os dados que um programa utiliza precisam ser armazenados na memória do computador
- Cada posição de memória do computador possui um endereço

8 1000	3.25 1001	'a' 1002	'g' 1003
'q' 1004	2 1005	'*' 1006	'1' 1007
1008	1009	1010	1011
1012	1013	1014	1015
1016	1017	1018	1019

← Memória

Variáveis (2/5)

- A partir dos endereços, é possível para o computador saber qual é o valor armazenado em cada uma das posições de memória.
- Como a memória pode ter bilhões de posições, é difícil controlar em qual endereço está armazenado um determinado valor!
- Para facilitar o controle sobre onde armazenar informação, os programas utilizam variáveis:
 - Uma variável corresponde a um nome simbólico de uma posição de memória.
 - Seu conteúdo pode variar durante a execução do programa.

Variáveis (3/5)

- Exemplo de variável:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     float y;
6     y = sin(1.5);
7     printf("y = %f", y);
8     printf("\n");
9     return 0;
10 }
```

A variável y armazena o valor de sin(1.5)

Variáveis (4/5)

- Cada variável pode possuir uma quantidade diferente de bytes, uma vez que os tipos de dados são representados de forma diferente.
- Portanto, a cada variável está associado um tipo específico de dados.
- Logo:

– O tipo da variável define quantos bytes de memória serão necessários para representar os dados que a variável armazena.

Variáveis (5/5)

- Dentro do programa, as variáveis são identificadas por seus nomes.
- Portanto, um programa deve declarar todas as variáveis que irá utilizar.
- Atenção!
 - A declaração de variáveis deve ser feita antes que a variável seja usada, para garantir que a quantidade correta de memória já tenha sido reservada para armazenar seu valor.

Identificadores (1/3)

- São nomes únicos definidos pelos programadores para identificar/distinguir os elementos de um algoritmo
- Algumas regras para os nomes de Identificadores:
 - 1) Devem começar por um caractere alfabético;
 - 2) Podem ser seguidos por mais caracteres alfabéticos e/ou numéricos;
 - 3) Não é permitido o uso de espaço em branco ou de caracteres especiais, como: @, #, &, *, +, ?, \$ (exceto o _);

Identificadores (2/3)

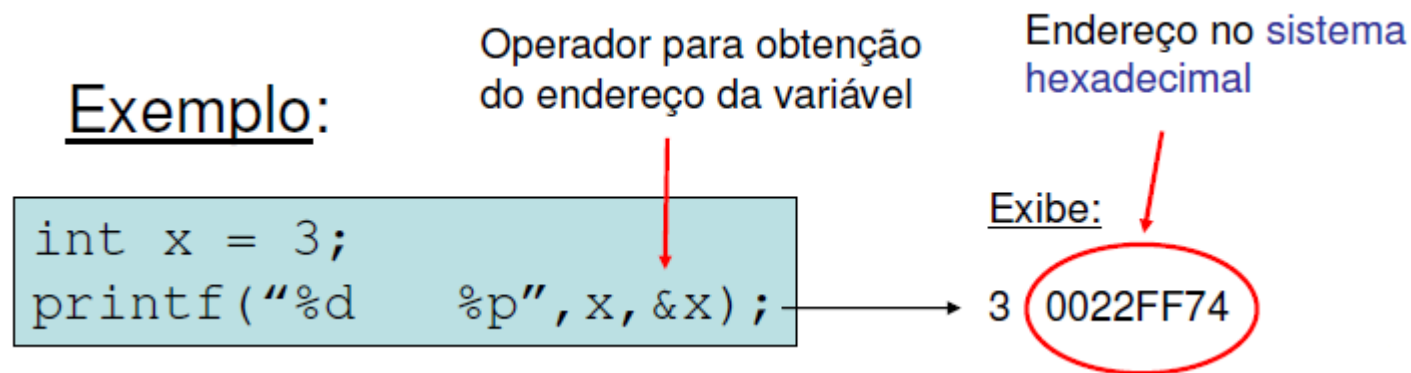
- Algumas regras para os nomes de Identificadores:
 - 4) Não poderá ser uma palavra reservada a uma instrução do algoritmo;
 - 5) Devem ser significativos.
 - 6) Não pode ser repetido dentro de um mesmo algoritmo/sub-algoritmo

Identificadores (3/3)

- Os seguintes identificadores estão errados:
 - `vm` → sem significado → `valor_médio`
 - `13salario` → não começa com caracter alfabético → `salário13`
 - `salario$` → usa caracter especial → `salario`
 - `salario_minimo` → correto
 - `salario+reajuste` → usa caracter especial → `salario_reajustado`
 - `novoSalario` → correto
 - `fumante?` → usa caracter especial → `fumante`
 - `preço médio` → tem espaço em branco → `preço_médio`
 - `%desconto` → não começa com caracter alfabético → `percentual_desconto`
 - `km/h` - usa caracter especial → `km_por_hora`

Endereços de Variáveis (1/4)

- Uma variável representa um nome simbólico para uma posição de memória.
- Cada posição de memória de um computador possui um endereço. Logo, o endereço de uma variável é o endereço da posição de memória representada pela variável.

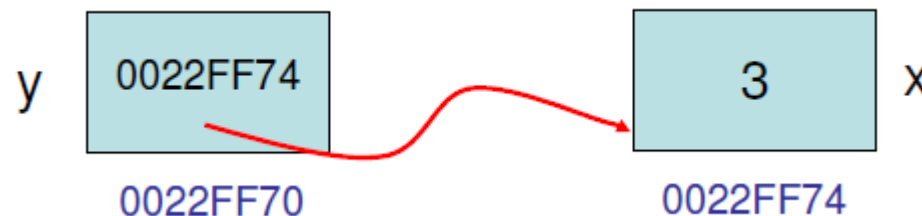


Endereços de Variáveis (2/4)

- Note que o endereço de uma variável é um valor. Logo, uma variável pode armazenar um endereço.
- Uma variável que armazena um endereço de memória é conhecida como ponteiro (*pointer*).
- Daí o porque do tag usado para exibir endereços de memória ser %p.

Endereços de Variáveis (3/4)

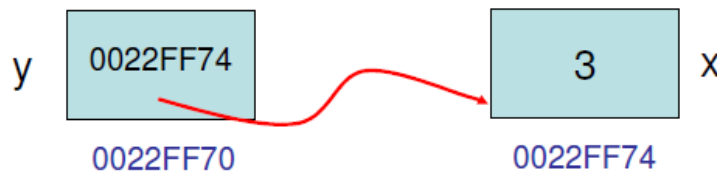
- Exemplo: suponha que y armazene o endereço 0022FF74 de uma posição de memória representada pela variável x. E ainda, que x contenha o valor inteiro 3.
- Esquematicamente, podemos representar:



- Diz-se que y é um ponteiro para x, ou que y aponta para x.

Endereços de Variáveis (4/4)

- Qual é o tipo da variável y?
 - Para declarar um ponteiro é preciso saber para qual tipo de valor este ponteiro irá apontar.
 - Exemplo do caso anterior:



- Neste caso, o ponteiro aponta para um valor inteiro. Assim, diz-se que o tipo de `y` é `int *`.
 - A declaração da variável `y` será:

```
int *y;
```

Indica que `y` é um ponteiro para `int`

Palavras reservadas

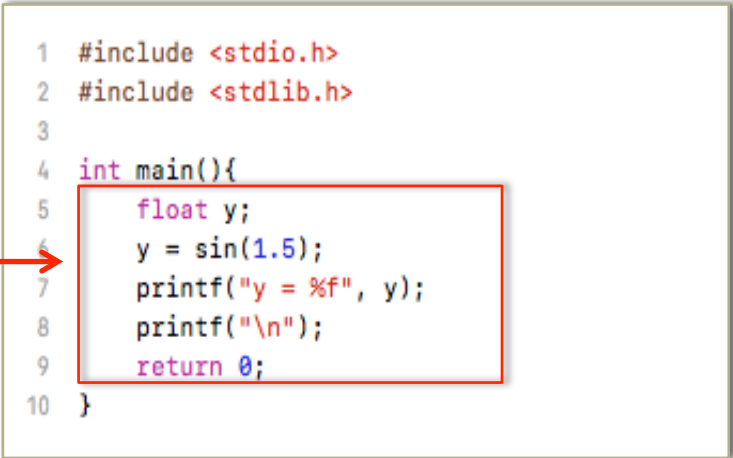
- São instruções primitivas que têm significados pré-determinados e fazem parte da estrutura de qualquer linguagem de programação
- Exemplos de palavras reservadas em C:

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

Escrevendo um Programa em C

- Escrever um programa em Linguagem C corresponde a escrever o corpo da função principal (main).
- O corpo de uma função sempre começa com abre-chave { e termina com fecha-chaves }.

Corpo da função



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     float y;
6     y = sin(1.5);
7     printf("y = %f", y);
8     printf("\n");
9     return 0;
10 }
```

Escrevendo um Programa em C

- A primeira linha do corpo da função principal do programa HelloWorld.c é:

```
float y;
```

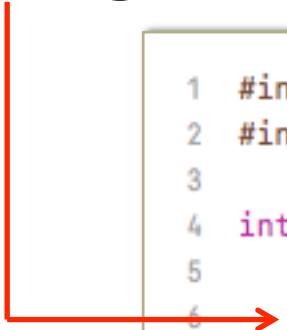
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      float y;
6      y = sin(1.5);
7      printf("y = %f", y);
8      printf("\n");
9      return 0;
10 }
```

Escrevendo um Programa em C

- Esta linha declara uma variável *y* para armazenar um número de ponto flutuante (SP).
- A declaração de uma variável não armazena valor algum na posição de memória que a variável representa.
- Ou seja, no caso anterior, vai existir uma posição de memória chamada *y*, mas ainda não vai existir valor armazenado.

Escrevendo um Programa em C

- Um valor pode ser atribuído a uma posição de memória representada por uma variável pelo operador de atribuição = .
- O operador de atribuição requer à esquerda um nome de variável e à direita, um valor.
- A linha seguinte de HelloWorld.c atribui um valor a *y*:




```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      float y;
6      y = sin(1.5);
7      printf("y = %f", y);
8      printf("\n");
9      return 0;
10 }
```

Escrevendo um Programa em C

- No lado direito do operador de atribuição existe uma referência à função seno com um parâmetro 1.5 (uma constante de ponto flutuante representando um valor em radianos.)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      float y;
6      y = sin(1.5);
7      printf("y = %f", y);
8      printf("\n");
9      return 0;
10 }
```



Escrevendo um Programa em C

- Em uma linguagem de programação, chamamos o valor entre parênteses da função, neste exemplo, o valor 1.5, de parâmetro da função.
- Da mesma forma, diz-se que $\sin(1.5)$ é o valor da função sin para o parâmetro 1.5.
- O operador de atribuição na linha $y = \sin(1.5)$ obtém o valor da função (0.997495) e o armazena na posição de memória identificada pelo nome y.
- Esta operação recebe o nome de: atribuição de valor a uma variável.

Escrevendo um Programa em C

- Atenção: O valor armazenado em uma variável por uma operação de atribuição depende do tipo da variável.
- Se o tipo da variável for int, será armazenado um valor inteiro (caso o valor possua parte fracionária, ela será desprezada).
- Se o tipo da variável for float ou double, será armazenado um valor de ponto flutuante (caso o valor não possua parte fracionária, ela será nula).

Escrevendo um Programa em C

- Exemplo:
 - Considere as seguintes declarações:

```
int a;  
float b;
```


- Neste caso, teremos:

Operação de atribuição	Valor armazenado
$a = (2 + 3) * 4$	20
$b = (1 - 4) / (2 - 5)$	1.0
$a = 2.75 + 1.12$	3
$b = a / 2.5$	1.2

Escrevendo um Programa em C

- As próximas linhas do programa HelloWorld.c são:

```
printf("y = %f", y);  
printf("\n");
```



```
1 #include <stdio.h>  
2 #include <stdlib.h>  
3  
4 int main(){  
5     float y;  
6     y = sin(1.5);  
7     printf("y = %f", y);  
8     printf("\n");  
9     return 0;  
10 }
```

* A função printf faz parte da biblioteca stdio

Escrevendo um Programa em C

- A função printf é usada para exibir resultados produzidos pelo programa e pode ter um ou mais parâmetros.
- O primeiro parâmetro da função printf é sempre uma string, correspondente à sequência de caracteres que será exibida pelo programa

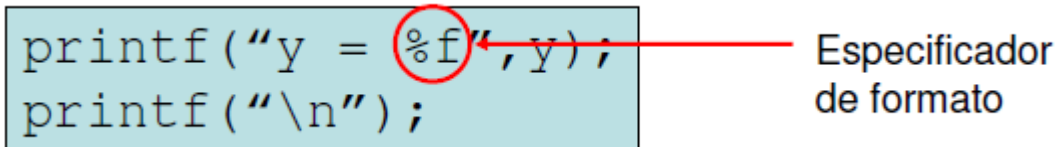
```
printf("y = %f", y);  
printf("\n");
```

Escrevendo um Programa em C

- Essa sequência de caracteres pode conter alguns tags que representam valores. Estes tags são conhecidos como especificadores de formato.

```
printf("y = %f", y);  
printf("\n");
```

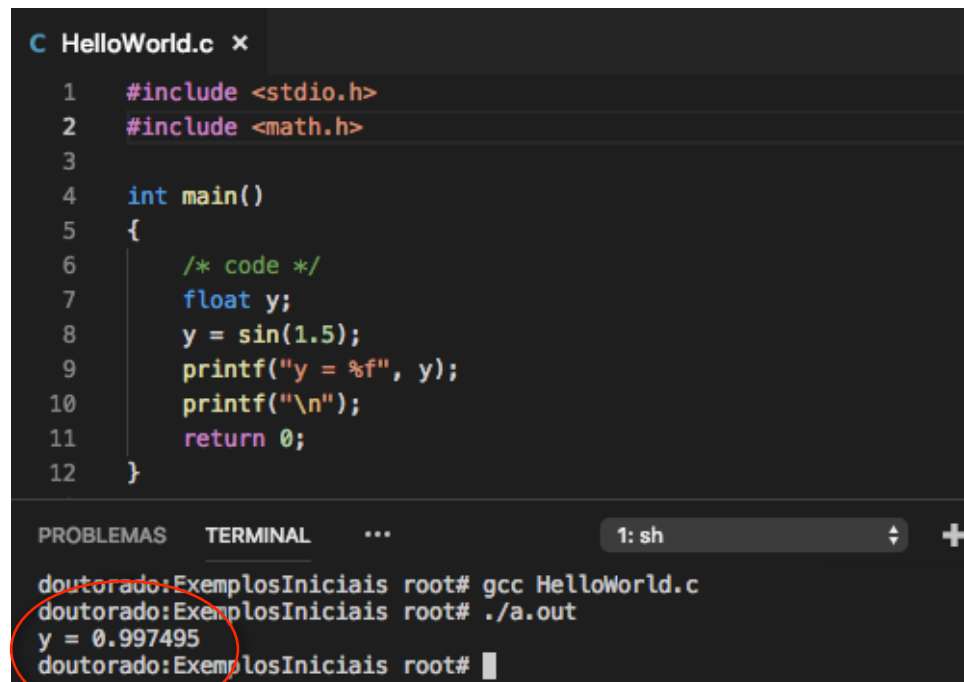
Especificador de formato



- Um especificador de formato começa sempre com o símbolo %. Em seguida, pode apresentar uma letra que indica o tipo do valor a ser exibido.
- Assim, printf("y = %f", y) irá exibir a letra y, um espaço em branco, o símbolo =, um espaço em branco, um valor de ponto flutuante.

Escrevendo um Programa em C

- Veja:



The image shows a code editor window titled 'HelloWorld.c' with the following C code:

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      /* code */
7      float y;
8      y = sin(1.5);
9      printf("y = %f", y);
10     printf("\n");
11     return 0;
12 }
```

Below the code editor is a terminal window. The terminal shows the compilation and execution of the program:

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
y = 0.997495
doutorado:ExemplosIniciais root#
```

A red circle highlights the output 'y = 0.997495' in the terminal, and a red arrow points from the word 'Resultado' to this circle.

Resultado

Escrevendo um Programa em C

- Na função printf, para cada tag existente no primeiro parâmetro, deverá haver um novo parâmetro que especifica o valor a ser exibido.

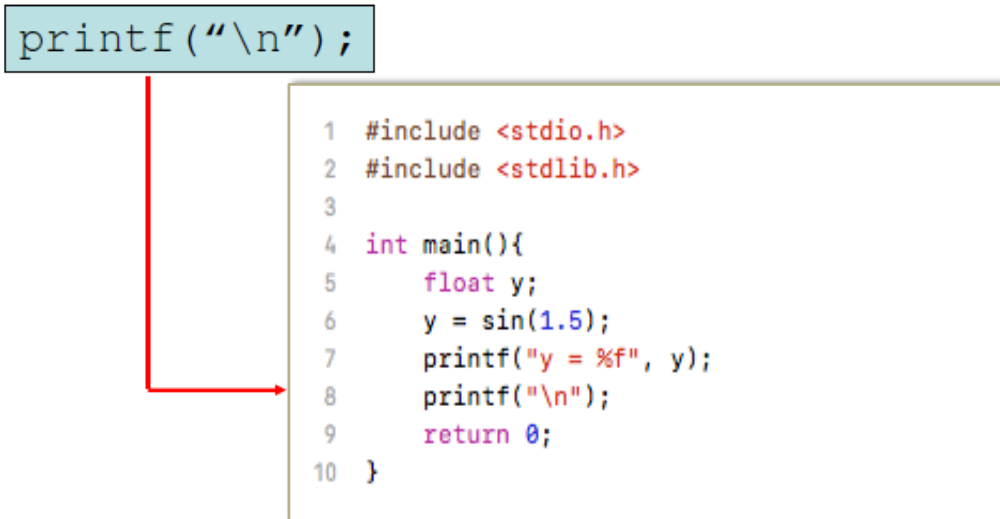
```
printf("a = %d, b = %c e c = %f", a, 'm', (a+b));
```

- A linguagem C utiliza o símbolo \ (barra invertida) para especificar alguns caracteres especiais:

Caractere	Significado
\a	Caractere (invisível) de aviso sonoro.
\n	Caractere (invisível) de nova linha.
\t	Caractere (invisível) de tabulação horizontal.
\'	Caractere de apóstrofo

Escrevendo um Programa em C

- Observe a próxima linha do programa HelloWorld.c:



A diagram illustrating a callout. A light blue box containing the text `printf("\n");` has a red line extending from its bottom center. This line turns 90 degrees to the right and then 90 degrees down, ending with an arrowhead pointing to the line number 8 in the code block below.


```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     float y;
6     y = sin(1.5);
7     printf("y = %f", y);
8     printf("\n");
9     return 0;
10 }
```

- Ela exibe “o caractere (invisível) de nova linha” . Qual o efeito disso? Provoca-se uma mudança de linha! Próxima mensagem será na próxima linha

Escrevendo um Programa em C

- A execução será retomada quando o usuário pressionar alguma tecla.
- A última linha do programa HelloWorld.c é:

```
return 0;
```




```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     float y;
6     y = sin(1.5);
7     printf("y = %f", y);
8     printf("\n");
9     return 0;
10 }
```


Escrevendo um Programa em C

- É usada apenas para satisfazer a sintaxe da linguagem C.
- O comando return indica o valor que uma função produz.
- Cada função, assim como na matemática, deve produzir um único valor.
- Este valor deve ter o mesmo tipo que o declarado para a função.

Escrevendo um Programa em C

- No caso do programa HelloWorld.c, a função principal foi declarada como sendo do tipo int. Ou seja, ela deve produzir um valor inteiro.



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     float y;
6     y = sin(1.5);
7     printf("y = %f", y);
8     printf("\n");
9     return 0;
10 }
```

- A linha return 0; indica que a função principal irá produzir o valor inteiro 0.

Escrevendo um Programa em C

- Mas e daí?!!
 - O valor produzido pela função principal não é usado em lugar algum!
 - Logo, não faz diferença se a última linha do programa for:

```
return 0;
```

```
return 1;
```

ou

```
return 1234;
```

Escrevendo um Programa em C

- Neste caso, o fato de a função produzir um valor não é relevante.
- Neste cenário, é possível declarar a função na forma de um procedimento.
- Um procedimento é uma função do tipo void, ou seja, uma função que produz o valor void (vazio, inútil, à-toa). Neste caso, ela não precisa do comando return.

Escrevendo um Programa em C

- Note que os parâmetros da função main também não foram usados neste caso.
- Portanto, podemos também indicar com void que a lista de parâmetros da função principal é vazia.
- Assim, podemos ter outras formas para HelloWorld.c:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main(void)
5  {
6      float y;
7      y = sin(1.5);
8      printf("y = %f", y);
9      printf("\n");
10     return;
11 }
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main(void)
5  {
6      float y;
7      y = sin(1.5);
8      printf("y = %f", y);
9      printf("\n");
10 }
```

Depende do compilador

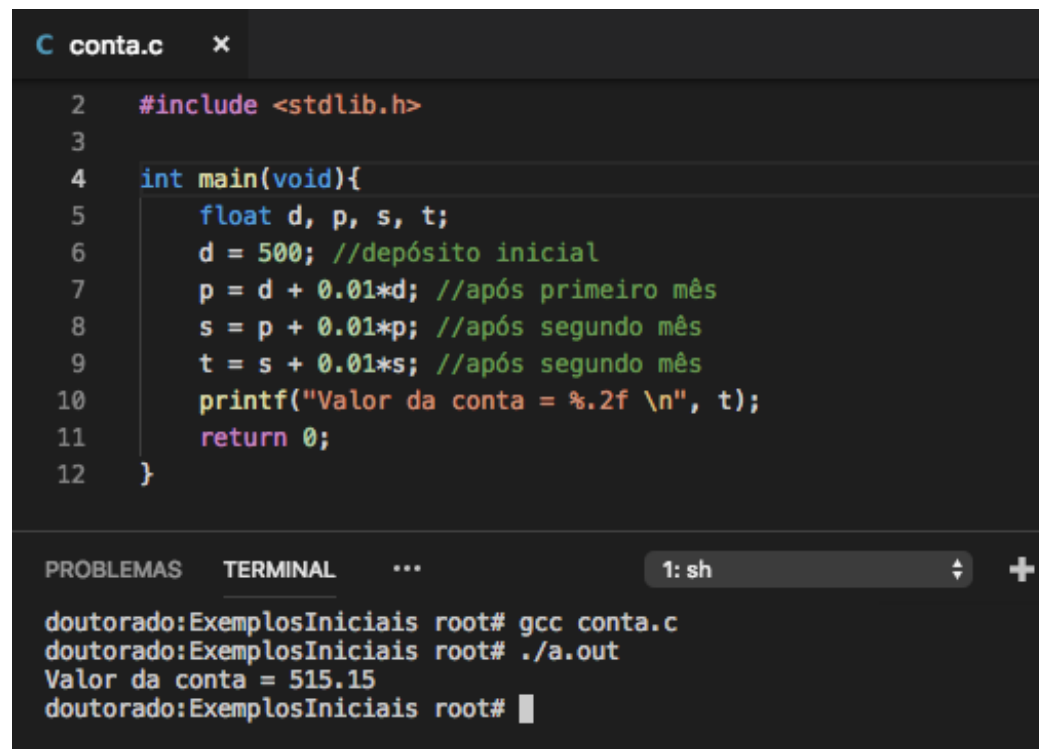
Problema (1/3)

- Uma conta poupança foi aberta com um depósito de R\$500,00. Esta conta é remunerada em 1% de juros aos mês. Qual será o valor da conta após três meses?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main(void){
5      float d, p, s, t;
6      d = 500; //depósito inicial
7      p = d + 0.01*d; //após primeiro mês
8      s = p + 0.01*p; //após segundo mês
9      t = s + 0.01*s; //após segundo mês
10     printf("Valor da conta = %.2f \n", t);
11     return 0;
12 }
```

Problema (2/3)

- Executando-se o programa, temos:



The screenshot shows a code editor with a file named `conta.c`. The code is a C program that calculates the value of a bank account after two months of 0.01% monthly interest. The code is as follows:

```
2  #include <stdlib.h>
3
4  int main(void){
5      float d, p, s, t;
6      d = 500; //depósito inicial
7      p = d + 0.01*d; //após primeiro mês
8      s = p + 0.01*p; //após segundo mês
9      t = s + 0.01*s; //após segundo mês
10     printf("Valor da conta = %.2f \n", t);
11     return 0;
12 }
```

Below the code editor, there is a terminal window showing the execution of the program. The terminal output is as follows:

```
PROBLEMAS  TERMINAL  ...  1: sh
doutorado:ExemplosIniciais root# gcc conta.c
doutorado:ExemplosIniciais root# ./a.out
Valor da conta = 515.15
doutorado:ExemplosIniciais root#
```

Problema (3/3)

- No programa conta.c, note que o tag usado na função printf é %.2f em vez de %f.
- Neste caso, o especificador de formato inclui também o número de dígitos desejados após o “ponto decimal” .
- Atenção!
 - É de extrema importância o uso de ponto-e-vírgula após cada instrução.
 - Com os pontos-e-vírgulas, o compilador sabe exatamente onde termina cada uma das instruções.

Leitura de Dados em C

- Uma outra forma de atribuir valores a variáveis é a leitura de dados. Em C, usa-se a função: `scanf`.
- Assim como `printf`, a função `scanf` pode ter vários parâmetros, sendo o primeiro uma string.
- No caso da função `scanf`, esta string deve conter apenas tags, separadas por espaços em branco.
- Os demais parâmetros da função `scanf` devem ser endereços de variáveis.

Leitura de Dados em C

- Sintaxe da função scanf():

```
scanf("expressão de controle", lista de argumentos)
```

- Onde:
 - Expressão de controle contém códigos de formatação, precedido pelo sinal %
 - Lista de argumentos consistem nos endereços das variáveis, referenciado em C pelo símbolo &

Leitura de Dados em C

- O que acontece quando o computador executa uma instrução de leitura de dados? Exemplo:

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

- A execução do programa é interrompida. O computador espera que o usuário digite algum valor e pressione a tecla Enter.
- Após pressionar Enter, o computador retoma a execução do programa e armazena o(s) valor(es) digitado(s) no(s) endereço(s) fornecido(s) a scanf.

Leitura de Dados em C

- O que difere a leitura de dados da operação de atribuição?

– Na **operação de atribuição**, o valor a ser atribuído é definido antes da execução do programa, enquanto numa **operação de leitura de dados**, o valor atribuído é definido durante a execução.


- Em programação, diz-se que coisas são estáticas quando ocorrem antes do programa executar e dinâmicas, quando ocorrem durante a execução.

<code>C = 32;</code>	→	Valor de C é estabelecido estaticamente .
<code>scanf ("%f", &C)</code>	→	Valor de C é estabelecido dinamicamente .

Leitura de Dados em C

- Exemplo:


```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      /* code */
7      int a, b, soma, subtracao, produto, divisao;
8      printf("Digite o valor de a: \n");
9      scanf("%d", &a);
10     printf("Digite o valor de b: \n");
11     scanf("%d", &b);
12     soma = a + b;
13     subtracao = a - b;
14     divisao = a / b;
15     produto = a * b;
16     printf("Soma %d: ", soma);
17     printf("Subtracao %d: ", subtracao);
18     printf("Produto %d: ", produto);
19     printf("Divisao %d: ", divisao);
20     return 0;
21 }
```



Leitura de Dados em C

- A função scanf() também pode ser usada assim:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      /* code */
7      int a, b, soma, subtracao, produto, divisao;
8      printf("Digite os valores de a e b: \n");
9      scanf("%d %d", &a, &b);
10     soma = a + b;
11     subtracao = a - b;
12     divisao = a / b;
13     produto = a * b;
14     printf("Soma %d: ", soma);
15     printf("Subtracao %d: ", subtracao);
16     printf("Produto %d: ", produto);
17     printf("Divisao %d: ", divisao);
18     return 0;
19 }
```



Leitura de Dados em C

- Código de formatação da função `scanf()`
 - `%c` – leia um único caractere
 - `%d` – leia um inteiro decimal
 - `%f` – leia um número flutuante
 - `%s` – leia uma série de caracteres
 - `%ld` – leia um inteiro longo
 - `%lf` – leia um double

Leitura de Dados em C

- Funções `getch()` e `getche()`
 - Função `scanf()` necessita que o usuário pressione [enter] depois de sua entrada para que termine a leitura

Função `getch()`

- Lê o caractere do teclado e não permite que seja impresso na tela

Função `getche()`

- Lê o caractere do teclado e permite que seja impresso na tela

Leitura de Dados em C

- Funções `getch()` e `getche()`
 - Nenhuma das duas funções aceitam argumentos e devolvem o caractere lido para a função que a chamou
 - Sintaxe:

```
char opcao;  
printf ("Digite opção:");  
opcao=getch() //ou getche()
```

Operadores Aritméticos

- Binário
 - = Atribuição
 - + Soma
 - - Subtração
 - * Multiplicação
 - / Divisão
 - % Devolve o resto da divisão inteira
- Unário
 - - Menos unário

Prioridade de Execução das Operações

- Porque as operações aritméticas devem ser feitas entre operandos do mesmo tipo?
- As representações dos números inteiros e dos números de ponto flutuante são diferentes.
- Ou seja, embora 1 e 1.0 são valores iguais, eles tem representações diferentes no computador.
- Prioridade de execução das operações:

1) expressão entre parênteses; 2) multiplicação e divisão (da esquerda para a direita); 3) operações de soma e subtração (da esquerda para a direita)

Prioridade de Execução das Operações

- Exemplo:
 - Seja $a=1.5$, $b=4$, $c=2$, $d=3$, $e=1.2$, $f=4.3$

```
v1 = (a * (c + d)) / (b * (e + f));
```

Ordem	Operação	Resultado	Conversão de tipo
1ª	$(c + d)$	$(2 + 3) = 5$	Não
2ª	$(e + f)$	$(1.2 + 4.3) = 5.5$	Não
3ª	$(a * 1ª)$	$(1.5 * 5) = 7.5$	Sim (5 para 5.0)
4ª	$(b * 2ª)$	$(4 * 5.5) = 22.0$	Sim (4 para 4.0)
5ª	$3ª / 4ª$	$7.5 / 22.0 = 0.341$	Não
6ª	$v1 = 5ª$	$V1 = 0.341$	Não

Conversão Explícita de Tipos

- É preciso muito cuidado com a divisão inteira
- O resultado da divisão inteira é sempre um número inteiro. Assim, se necessário, pode-se usar uma conversão explícita de tipo (casting)

```
int a = 10, b = 3;
```

```
int c;
```

```
float d;
```

```
c = a / b;
```

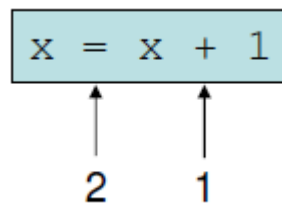
→ c = 3

```
d = (float) a / b;
```

→ d = 3.333333

Operadores de Incremento e Decremento

- Uma operação muito comum em programas de computador é incrementar o valor da variável de 1.
- Para fazer isso devemos:
 - 1.º - Somar 1 ao valor atual da variável;
 - 2.º - Armazenar o resultado na própria variável.



- Como a operação incremento de 1 em C é muito comum, tem-se um operador especial: `++`.

Operadores de Incremento e Decremento

- Ao invés de escrevermos

$x = x + 1$, podemos escrever: **$x++$**

- Da mesma forma, para a operação decremento de 1:

$x = x - 1$, podemos escrever: **$x--$**

- Os operadores $++$ e $--$ podem ser usados como prefixo ou como sufixo do nome da variável.

<pre>int a = 5, b = 3; int c;</pre>							
<pre>c = a++ + b;</pre>	→						
<pre>c = ++a + b;</pre>	→						
	<table border="0"><tr><td>a = 6</td><td>b = 3</td><td>c = 8</td></tr><tr><td>a = 7</td><td>b = 3</td><td>c = 10</td></tr></table>	a = 6	b = 3	c = 8	a = 7	b = 3	c = 10
a = 6	b = 3	c = 8					
a = 7	b = 3	c = 10					

Operadores de Incremento e Decremento

- As operações de incremento (++) e decremento (--) são exemplos de operações combinadas com a atribuição.
- Na linguagem C, sempre que for necessário escrever uma operação de atribuição da forma:

```
variavel = variavel operador expressao;
```

poderemos combinar as operações. Exemplos:

x = x + 5;		x += 5;
x = x - (a + b);	→	x -= (a + b);
x = x * (a - b);		x *= (a - b);
x = x / (x + 1);		x /= (x + 1);

Operadores Aritméticos de Atribuição

- As operações com estes operadores são mais compactas e normalmente produzem um código de máquina mais eficiente

Expressão:	Equivale a:
<code>i += 2;</code>	<code>i = i + 2;</code>
<code>x *= y + 1;</code>	<code>x = x * (y + 1);</code>
<code>t /= 2.5;</code>	<code>t = t / 2.5;</code>
<code>p %= 5;</code>	<code>p = p % 5;</code>
<code>d -= 3;</code>	<code>d = d - 3;</code>

Problemas (1/10)

```
C HelloWorld.c x
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      /* Algoritmo que imprime mensagem na tela */
7      printf("É preciso fazer todos os algoritmos para aprender!!! \n");
8      return 0;
9  }
10
11
12
13
```

PROBLEMAS TERMINAL ... 1: sh ↕ + ▢ 🗑

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
É preciso fazer todos os algoritmos para aprender!!!
doutorado:ExemplosIniciais root#
```

Problemas (2/10)

```
C HelloWorld.c x
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      /* Produto entre dois números */
7      int produto;
8      produto = 2 * 3;
9      printf("O produto entre os dois números é: %d ", produto);
10     printf("\n");
11     return 0;
12 }
```

PROBLEMAS TERMINAL ... 1: sh + □

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
O produto entre os dois números é: 6
doutorado:ExemplosIniciais root#
```

Problemas (3/10)

```
C HelloWorld.c x
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      /* Imprime a média entre 8, 9 e 7 */
7      float media;
8      media = (8 + 9 + 7) / 3;
9      printf("A média entre os números é: %f ", media);
10     printf("\n");
11     return 0;
12 }
```

PROBLEMAS TERMINAL ... 1: sh ▾ +

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
A média entre os números é: 8.000000
doutorado:ExemplosIniciais root#
```

Problemas (4/10)

```
C HelloWorld.c x
3
4  int main()
5  {
6      /* Imprime antecessor e sucessor */
7      int numero, antecessor, sucessor;
8      printf("Entre com o número: ");
9      scanf("%d", &numero);
10     antecessor = numero - 1;
11     sucessor = numero + 1;
12     printf("Antecessor do número %d: %d \n", numero, antecessor);
13     printf("Sucessor do número %d: %d \n", numero, sucessor);
14     return 0;

```

PROBLEMAS TERMINAL ... 1: sh + [icon] [icon]

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
Entre com o número: 5
Antecessor do número 5: 4
Sucessor do número 5: 6
doutorado:ExemplosIniciais root#
```

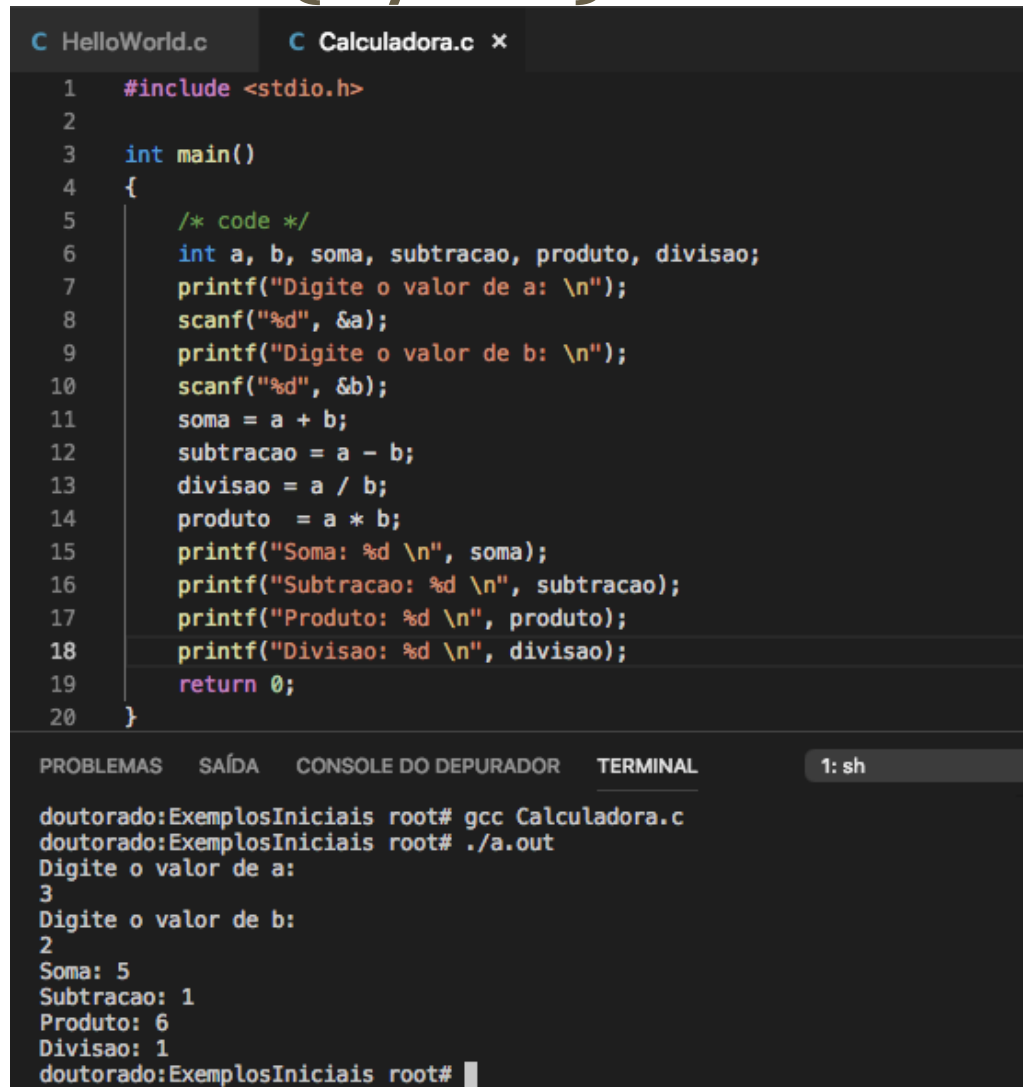
Problemas (5/10)

```
C HelloWorld.c x
2  #include <math.h>
3
4  int main()
5  {
6      /* Imprime os tipos em C */
7      int evento = 5;;
8      char corrida = 'C';
9      float tempo = 25.2627;
10     printf("Tempo vitorioso na eliminatória %c do evento %d foi: %f \n", corrida, evento, tempo);
11     return 0;
12 }
13
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: sh +

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
Tempo vitorioso na eliminatória C do evento 5 foi: 25.262699
doutorado:ExemplosIniciais root#
```

Problemas (6/10)



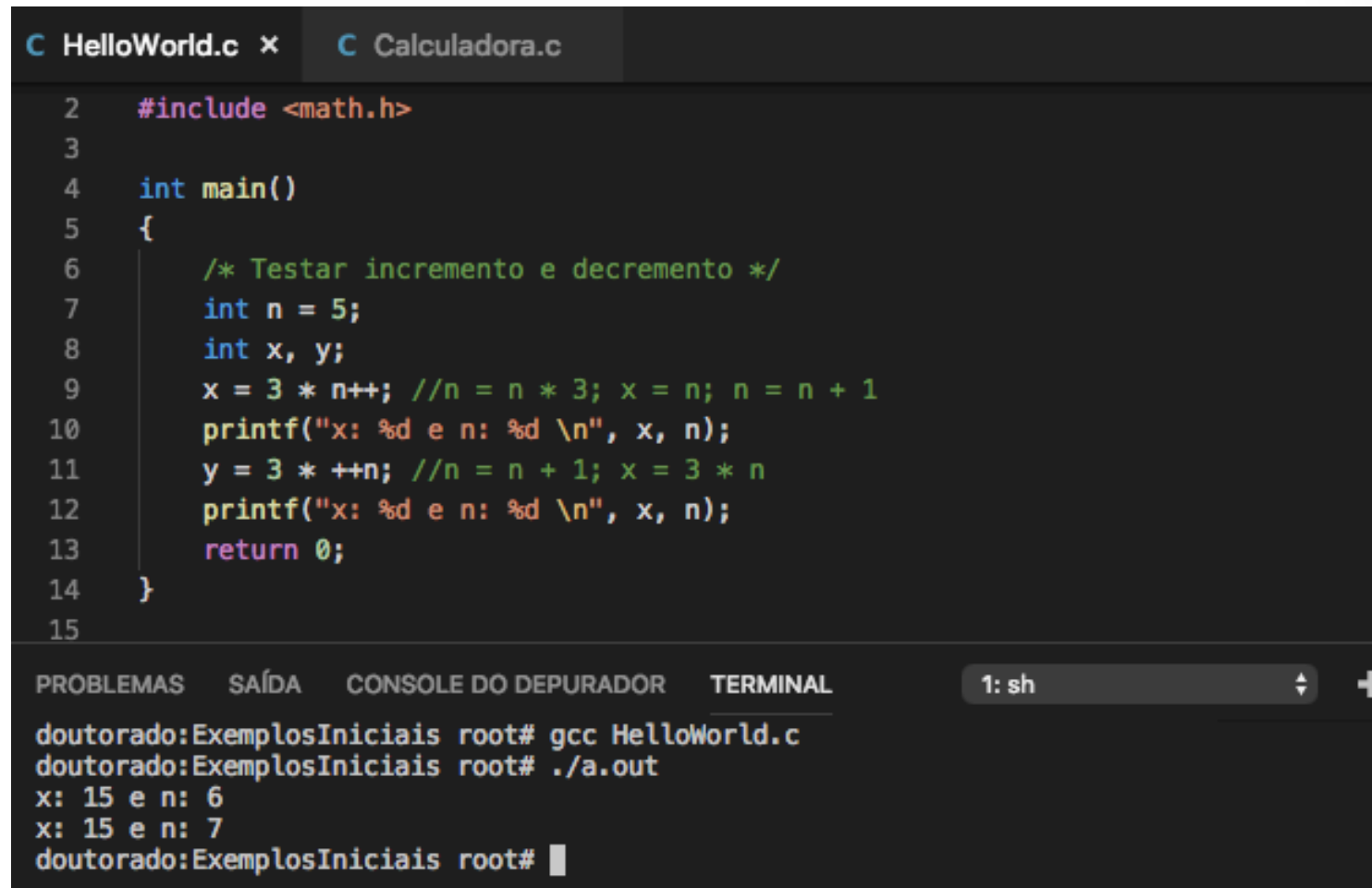
The image shows a code editor with two tabs: 'HelloWorld.c' and 'Calculadora.c'. The 'Calculadora.c' tab is active, displaying a C program that takes two integers as input and calculates their sum, difference, product, and quotient. The code is as follows:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      /* code */
6      int a, b, soma, subtracao, produto, divisao;
7      printf("Digite o valor de a: \n");
8      scanf("%d", &a);
9      printf("Digite o valor de b: \n");
10     scanf("%d", &b);
11     soma = a + b;
12     subtracao = a - b;
13     divisao = a / b;
14     produto = a * b;
15     printf("Soma: %d \n", soma);
16     printf("Subtracao: %d \n", subtracao);
17     printf("Produto: %d \n", produto);
18     printf("Divisao: %d \n", divisao);
19     return 0;
20 }
```

Below the code editor is a terminal window with the following output:

```
PROBLEMAS  SAÍDA  CONSOLE DO DEPURADOR  TERMINAL
doutorado:ExemplosIniciais root# gcc Calculadora.c
doutorado:ExemplosIniciais root# ./a.out
Digite o valor de a:
3
Digite o valor de b:
2
Soma: 5
Subtracao: 1
Produto: 6
Divisao: 1
doutorado:ExemplosIniciais root#
```

Problemas (7/10)



```
C HelloWorld.c x C Calculadora.c
2  #include <math.h>
3
4  int main()
5  {
6      /* Testar incremento e decremento */
7      int n = 5;
8      int x, y;
9      x = 3 * n++; //n = n * 3; x = n; n = n + 1
10     printf("x: %d e n: %d \n", x, n);
11     y = 3 * ++n; //n = n + 1; x = 3 * n
12     printf("x: %d e n: %d \n", x, n);
13     return 0;
14 }
15
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: sh

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
x: 15 e n: 6
x: 15 e n: 7
doutorado:ExemplosIniciais root#
```

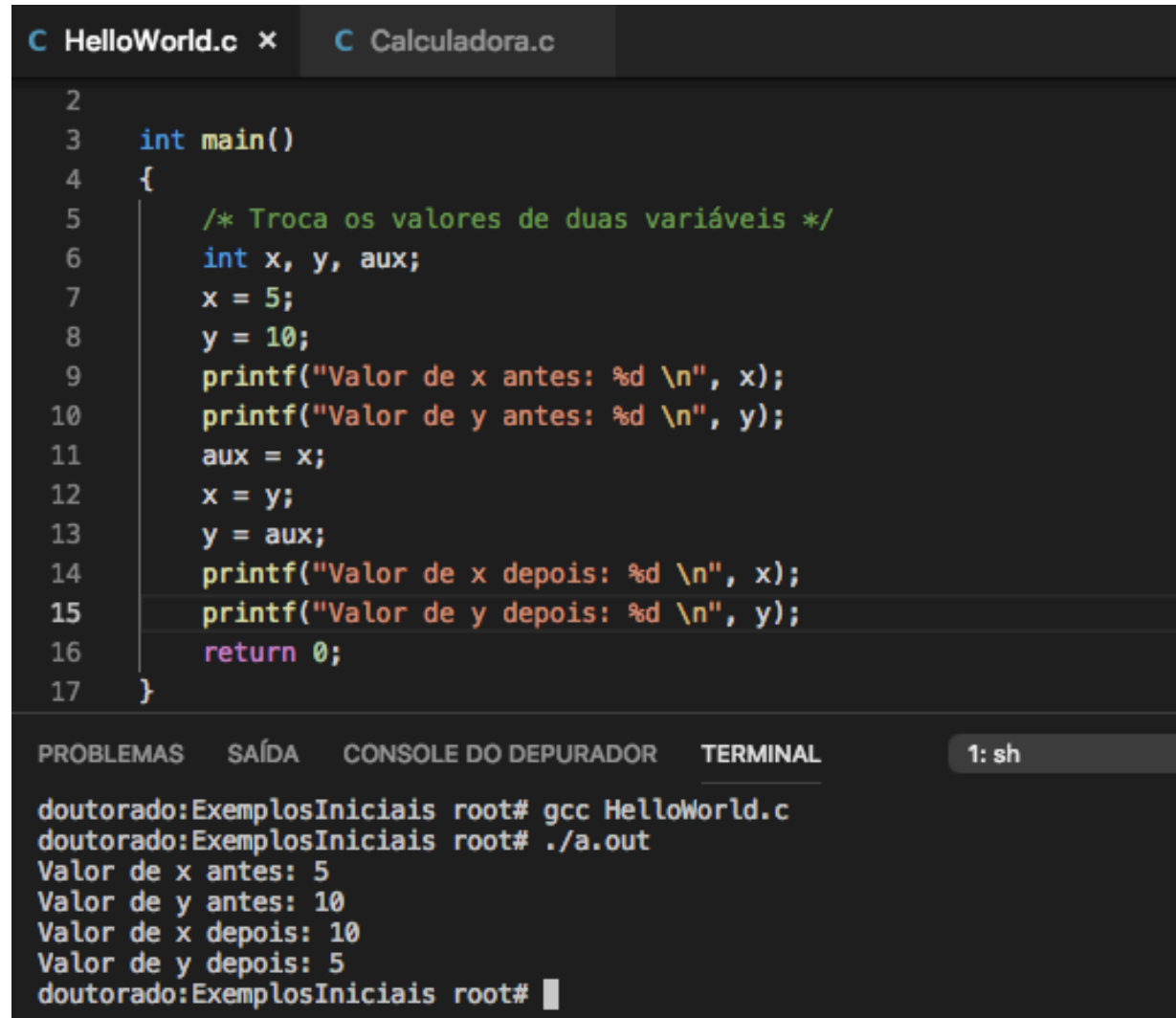

Problemas (8/10)

```
C HelloWorld.c x C Calculadora.c
1  #include <stdio.h>
2
3  int main()
4  {
5      /* Testar operadores aritméticos */
6      int x = 20;
7      x += 2; // x = x + 2
8      printf("x+=2: %d \n",x);
9      x -= 2; // x = x - 2
10     printf("x-=2: %d \n",x);
11     x *= 2; // x = x * 2
12     printf("x*=2: %d \n",x);
13     x /= 2; // x = x / 2
14     printf("x/=2: %d \n",x);
15     return 0;
16 }
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
x+=2: 22
x-=2: 20
x*=2: 40
x/=2: 20
doutorado:ExemplosIniciais root#
```

Problemas (9/10)



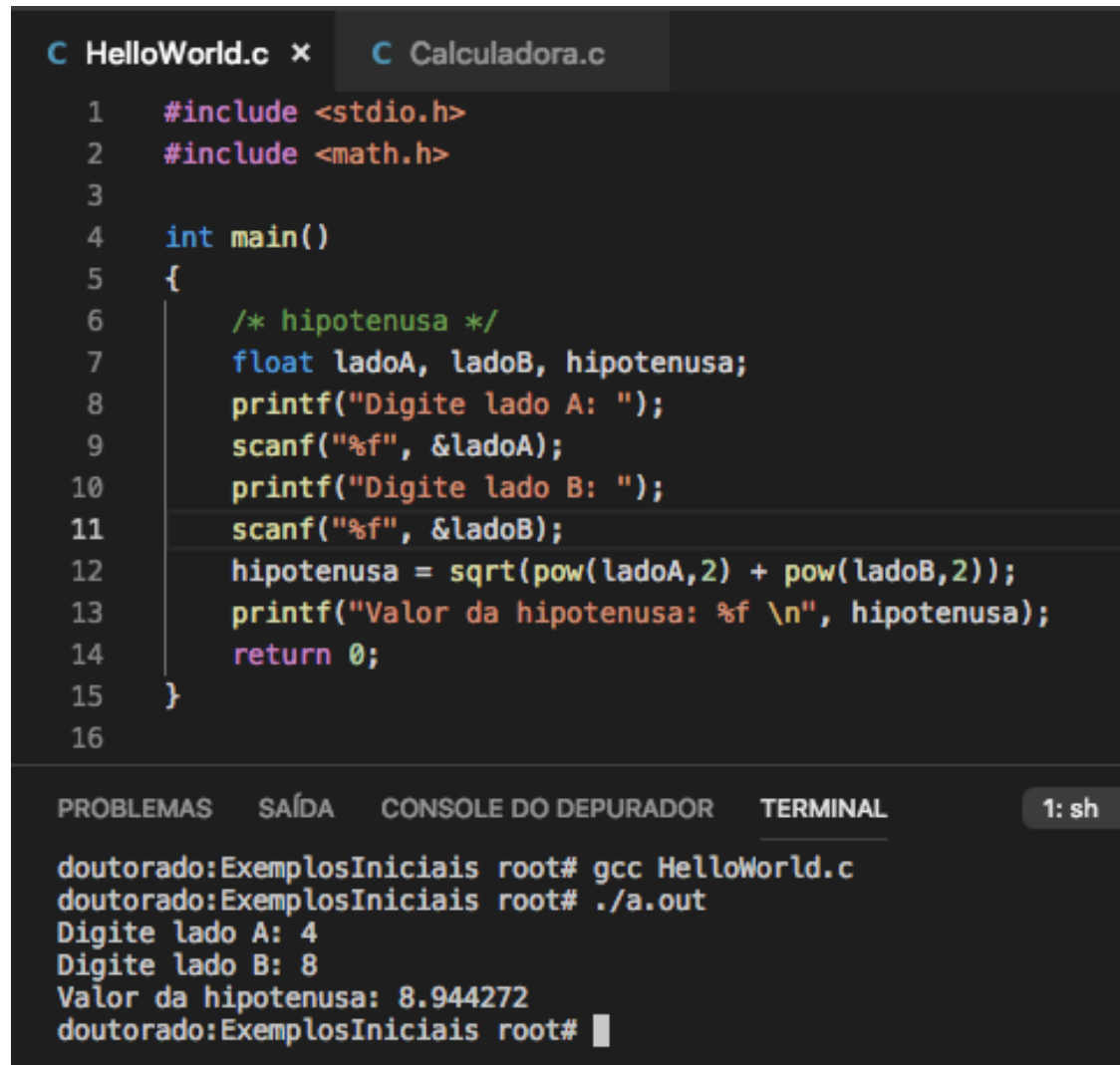
The image shows a code editor with two tabs: 'HelloWorld.c' and 'Calculadora.c'. The 'HelloWorld.c' tab is active, displaying a C program that swaps the values of two variables, x and y, using a temporary variable aux. The code is as follows:

```
2
3  int main()
4  {
5      /* Troca os valores de duas variáveis */
6      int x, y, aux;
7      x = 5;
8      y = 10;
9      printf("Valor de x antes: %d \n", x);
10     printf("Valor de y antes: %d \n", y);
11     aux = x;
12     x = y;
13     y = aux;
14     printf("Valor de x depois: %d \n", x);
15     printf("Valor de y depois: %d \n", y);
16     return 0;
17 }
```

Below the code editor is a terminal window with tabs for 'PROBLEMAS', 'SAÍDA', 'CONSOLE DO DEPURADOR', and 'TERMINAL'. The 'TERMINAL' tab is active, showing the execution of the program. The output is as follows:

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
Valor de x antes: 5
Valor de y antes: 10
Valor de x depois: 10
Valor de y depois: 5
doutorado:ExemplosIniciais root#
```

Problemas (10/10)



```
C HelloWorld.c x C Calculadora.c
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      /* hipotenusa */
7      float ladoA, ladoB, hipotenusa;
8      printf("Digite lado A: ");
9      scanf("%f", &ladoA);
10     printf("Digite lado B: ");
11     scanf("%f", &ladoB);
12     hipotenusa = sqrt(pow(ladoA,2) + pow(ladoB,2));
13     printf("Valor da hipotenusa: %f \n", hipotenusa);
14     return 0;
15 }
16
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: sh

```
doutorado:ExemplosIniciais root# gcc HelloWorld.c
doutorado:ExemplosIniciais root# ./a.out
Digite lado A: 4
Digite lado B: 8
Valor da hipotenusa: 8.944272
doutorado:ExemplosIniciais root#
```

Lista de Exercícios

1. Entrar com um número no formato CDU e imprimir invertido UDC (Exemplo: 123, sairá 321). O número deverá ser armazenado em outra variável antes de ser impresso.
2. Criar um algoritmo que, dado um número de conta corrente com três dígitos, retorne o seu dígito verificador, o qual é calculado da seguinte maneira:
 - Exemplo: numero da conta 235
 - Somar o numero da conta com o seu inverso: $235 + 532 = 767$
 - Multiplicar cada dígito pela sua ordem posicional e somar estes resultados: $767 - > 7*1 + 6*2 + 7*3 = 40$
 - O último dígito deste resultado é o dígito verificador da conta (40-> 0)

Lista de Exercícios

3. Ler dois números reais e imprimir o quadrado da diferença do primeiro valor pelo segundo valor e a diferença dos quadrados.
4. Criar um algoritmo que leia um valor de hora e informe quantos minutos e segundos se passaram desde o início do dia.
5. Efetuar o cálculo da quantidade de litros de combustível gastos em uma viagem, sabendo-se que o carro faz 12Km com um litro. Deverão ser fornecidos o tempo gasto na viagem e a velocidade média. Utilize as seguintes fórmulas:

$\text{distancia} = \text{tempo} * \text{velocidade}$ e $\text{litros_usados} = \text{distancia} / 12$.

O algoritmo deverá apresentar os valores da velocidade média, tempo gasto na viagem, distância percorrida e a quantidade de litros utilizados na viagem.

Lista de Exercícios

6. Calcular e apresentar o valor do volume de uma lata de óleo, utilizando a fórmula: $\text{volume} = 3.14159 * R^2 * \text{altura}$. Usar constantes de C.
7. Entrar com a razão de uma PA e o valor do 1º termo. Calcular e imprimir o 10º termo da série.
8. Entrar com o ângulo em graus e imprimir: seno, co-seno e tangente deste ângulo. Utilizar as funções matemáticas de C. Caso o resultado não seja o esperado, explicar o motivo e como poderia resolver este problema.
9. Entrar com o número e a base em que se deseja calcular o logaritmo deste número e imprimi-lo.
10. Faça um algoritmo que receba um caractere e mostre o seu código numérico ASCII

Lista de Exercícios

11. Entrar com os valores para xnum1, xnum2 e xnum3 e imprimir o valor de x, sabendo-se que:
$$X = \text{xnum1} + ((\text{xnum2})/(\text{xnum3}+\text{xnum1})) + 2(\text{xnum1}-\text{xnum2}) + \log_2 64.$$
12. Criar um algoritmo que efetue o cálculo do salário líquido de um professor. Os dados fornecidos são: valor da hora aula, número de aulas dadas no mês e percentual de desconto do INSS.
13. O custo ao consumidor de um carro novo é a soma do custo de fábrica com a percentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que a percentagem do distribuidor seja de 28% e os impostos de 45%, escrever um algoritmo que leia o custo de fábrica de um carro e escreva o custo ao consumidor.
14. Faça um algoritmo que receba um número inteiro e mostre o resto da divisão desse número por 5.
15. Faça um algoritmo que leia a idade de uma pessoa expressa em dias e mostre-a expressa em anos, meses e dias.

Referências

- Medina, Marco; Fertig, Cristina. Algoritmos e Programação: teoria e prática. São Paulo: Novatec Editora, 2006.
- Lopes, Anita; Garcia, Guto. Introdução à Programação: 500 algoritmos resolvidos. Rio de Janeiro: Editora Campus, 2002.
- Mizrani, Victorine Viviane. Treinamento em Linguagem C, Módulo 1. Editora Makron Books.
- Transparências modificadas do professor Dr. Flavio Luiz Cardeal Pádua, do Centro Federal de Educação Tecnológica de Minas Gerais