Introdução à Programação

Linguagem de Programação I

Professora Drª. Alana Morais

2019

Computadores na resolução de problemas

- Problemas que são custosos para o humano executar, por diversas razões.

- Uso de programas específicos de desenvolvimento
  - Algoritmos
  - Linguagens de Programação

Algoritmo

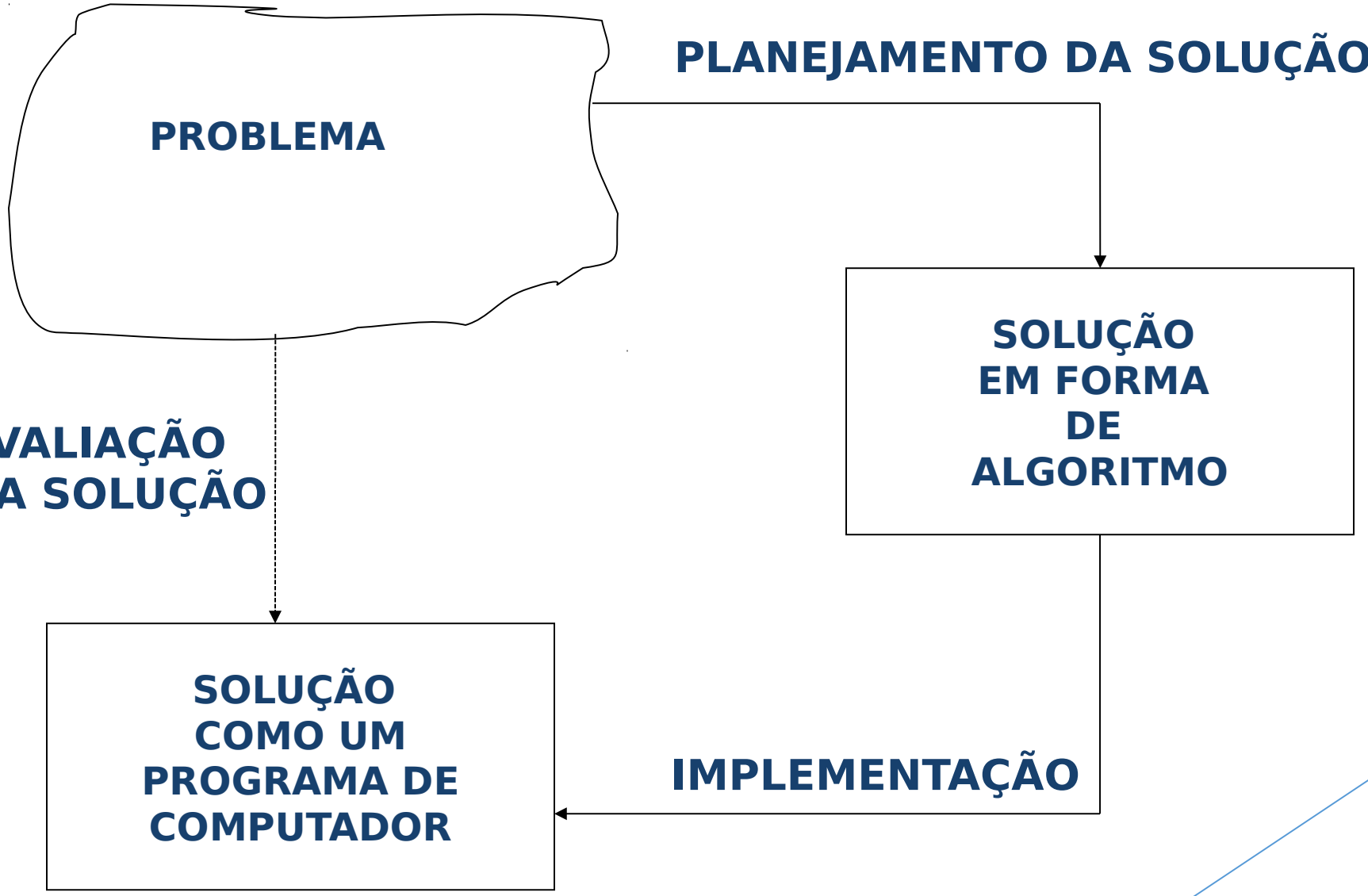
- Sequência ordenada, sem ambiguidade, de passos que levam à solução de um dado problema, em um tempo finito.
- Passos:
  - Simples
  - Não ambíguos
  - Ordenados
  - Efetivos

  - Entradas: 0 ou mais
  - Saídas: pelo menos uma

Alguns Paradigmas de Linguagens de Programação

- Procedural
  - Ex.: PASCAL, FORTRAN, ALGOL, BASIC
- Funcional
  - Ex.: ML, Miranda
- Lógico
  - Ex.: Prolog
- Orientado a objetos
  - Ex.: C++, Smalltalk, PASCAL, JAVA, PHP

Resolução de problemas e programação



Análise e Solução de Problemas

- Compreensão rigorosa do problema.
  - Um método de solução é escolhido e desenvolvido.
  - Descrição do processo de solução passo a passo (algoritmo).
  - Programação do algoritmo e depuração do programa.
  - Validação da solução.

Programa

- Roteiro que orienta o computador, mostrando-lhe a sequência de operações necessárias para executar uma determinada tarefa.
- Sequência de instruções que dirigem a CPU na execução de alguma tarefa.
- Composto por uma série de comandos e instruções.

- Como:
  - Conhecer as instruções (Comandos)
  - Saber como escrever as instruções (Sintaxe dos comandos)
  - Entender ações resultantes da execução das instruções (Semântica dos comandos)
  - Possuir raciocínio lógico para chegar a uma sequência de instruções que solucione o problema proposto.

Elementos importantes em C

- Identificação de Tokens
  - Ex: printf (“Exemplo de Texto”);
- *Case Sensitive*
- Espaços e tabs são ignorados
- Palavras-chave não podem ser usadas para nomear variáveis
- Cinco tipos básicos:
  - char, int, float, double e void.
- Comentários
  - // Texto
  - /* Texto */
- Delimitadores: ; {}

Palavras-Reservadas

- Essas palavras reservadas não podem ser usadas como constantes ou variáveis ou quaisquer outros nomes de identificadores

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

Tipos

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Tipos flutuantes

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Exercício

- Para obter o tamanho exato de um tipo ou uma variável em uma plataforma específica, você pode usar o operador sizeof.

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

int main() {
   printf("Storage size for int : %d \n", sizeof(int));

   return 0;
}
```

Printf

- **int printf(const char *format, ...)**

specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

Printf

```c
/* printf example */
#include <stdio.h>

int main()
{
   printf ("Characters: %c %c \n", 'a', 65);
   printf ("Decimals: %d %ld\n", 1977, 650000L);
   printf ("Preceding with blanks: %10d \n", 1977);
   printf ("Preceding with zeros: %010d \n", 1977);
   printf ("Some different radices: %d %x %o %#x %#o \n", 100, 100, 100, 100, 100);
   printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);
   printf ("Width trick: %*d \n", 5, 10);
   printf ("%s \n", "A string");
   return 0;
}
```

```
Characters: a A
Decimals: 1977 650000
Preceding with blanks:       1977
Preceding with zeros: 0000001977
Some different radices: 100 64 144 0x64 0144
floats: 3.14 +3e+000 3.141600E+000
Width trick:    10
A string
```

Exercício

- Experimente colocar um '**\n**' entre os **%s** na string de formato.
  - **printf("%s\n%s","teste1", "outra string");**

Sua vez ...

- Imprima a seguinte frase:
  - Aula de LP1 (String)
  - Alana Morais (String)
  - 2018 (inteiro)
  - 21/08 (inteiro char String)

Declaração de Variáveis

- Uma definição de variável informa ao compilador onde e quanto armazenamento deve ser criado para a variável.
- Todas as variáveis devem ser declaradas antes do seu uso.
- Uma definição de variável especifica um tipo de dados e contém uma lista de uma ou mais variáveis desse tipo da seguinte maneira:

```
int d = 3, f = 5;           // definition and initializing d and f.
byte z = 22;                // definition and initializes z.
char x = 'x';               // the variable x has the value 'x'.
```

Declaração de Variáveis

- Uma declaração de variável é útil quando você está usando vários arquivos e você define sua variável em um dos arquivos que estarão disponíveis no momento da vinculação do programa.

- A palavra-chave **extern** é usada para declarar uma variável em qualquer lugar do código.

```c
#include <stdio.h>

// Variable declaration:
extern int a, b;
extern int c;
extern float f;

int main () {

   /* variable definition: */
   int a, b;
   int c;
   float f;

   /* actual initialization */
   a = 10;
   b = 20;

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

   f = 70.0/3.0;
   printf("value of f : %f \n", f);

   return 0;
}
```

É muito importante dar bons nomes às suas variáveis! Nada de x1, x2, x3 ... Ok?

Exercício

- Crie um programa que receba uma matrícula, salário e telefone e, por fim, imprima tais valores na tela.
- Os valores deve ser passados diretamente para variável.
- Escolha os tipos mais adequados para cada uma das variáveis.

Constantes

- Constantes são identificadores que não podem ter seus valores alterados durante a execução do programa.
- Para criar uma constante existe o comando #define que, em geral é colocado no início do programa-fonte.
- Letras maiúsculas

```c
#include <stdio.h>

#define LENGTH 10
#define WIDTH  5
#define NEWLINE '\n'

int main() {
   int area;

   area = LENGTH * WIDTH;
   printf("value of area : %d", area);
   printf("%c", NEWLINE);

   return 0;
}
```

Constantes

- É possível usar o prefixo **const** para declarar constantes com um tipo específico da seguinte forma:

```c
#include <stdio.h>

int main() {
   const int  LENGTH = 10;
   const int  WIDTH = 5;
   const char NEWLINE = '\n';
   int area;

   area = LENGTH * WIDTH;
   printf("value of area : %d", area);
   printf("%c", NEWLINE);

   return 0;
}
```

Exercício

- Implemente a fórmula de transformação de uma temperatura do formato Celsius para o formato Kelvin.
- Imprima a temperatura em Kelvin na tela.

Operadores Aritméticos

Operator	Description	Example
+	Adds two operands.	A + B = 30
–	Subtracts second operand from the first.	A – B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Operadores Relacionais

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

Operadores Lógicos

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

Operadores de tarefa

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C %= A is equivalent to C = C % A

Sua vez ...

- TESTE OS OPERADORES DE TAREFA

Exercício

- Faça um programa que converta automaticamente os valores de uma distância em quilômetros para milhas.
- Atribua valores diretamente nas variáveis.
- Crie uma constante de conversão se necessário.

Dúvidas?

alanamm.prof@gmail.com