Conceitos Básicos Linguagem C

PROF. MAURÍCIO A DIAS

MACDIASPAE@GMAIL.COM

Método

Método básico para construção de algoritmos

1. Compreender completamente o problema a ser resolvido, se possível dividindo em partes menores
2. Definir os dados de entrada
3. Definir os cálculos que serão executados
4. Definir os dados de saída
5. Construir o Algoritmo
6. Fazer o teste de mesa
7. Implementar em uma linguagem de programação, compilar e rodar

Aritmética Binária

Uma possível representação pra números binários é simplesmente utilizar os bits para representar números decimais

Como converter binário em decimais e decimais em binários?

Decompondo números decimais

O número 1352 pode ser entendido como:

$$2 * 10^0 + 5 * 10^1 + 3 * 10^2 + 1 * 10^3 =$$

$$2 + 50 + 300 + 1000 = 1352$$

Aritmética Binária

Os números decimais utilizam base 10, portanto os algarismos válidos são 0,1,2,3,4,5,6,7,8,9

A aritmética binária utiliza a base 2, portanto os algarismos válidos são apenas 0 e 1 chamados de *bits*

Como representar um número em aritmética binária?

Aritmética Binária

O número 127 em binário ficaria da seguinte forma 01111111

Como verificar?

Considerando que o bit mais significativo, ou seja, o que vale mais é o primeiro, e o menos significativo, ou seja, o que vale menos é o último. Iremos considerar portanto o número de trás para frente. Interpretamos o número assim:

$$1 * 2^0 + 1 * 2^1 + 1 * 2^2 + 1 * 2^3 + 1 * 2^4 + 1 * 2^5 + 1 * 2^6 + 0 * 2^7 =$$

$$1 + 2 + 4 + 8 + 16 + 32 + 64 + 0 = 127$$

Soma de números binários

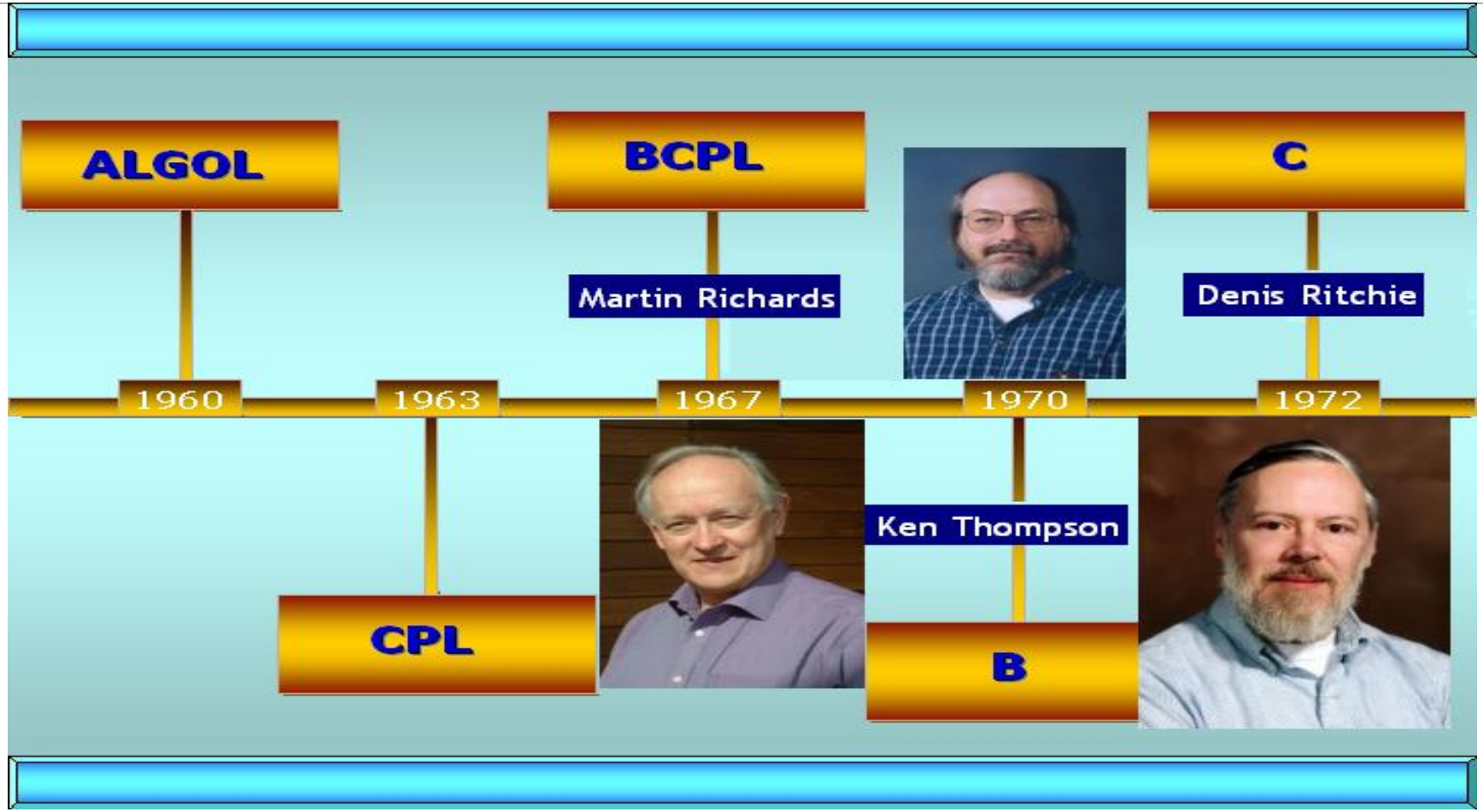
Somando o número 0101 com o número 1110 teremos

$$\begin{array}{r} 0101 \\ 1110 \quad + \\ \hline 1\,0\,0\,11 \end{array}$$

Assim como em base decimal 9 + 1 dá 0 e “vai 1”, em binário 1 + 1 dá 0 e “vai 1”

A resposta precisou de 5 bits para ser representada, caso houvessem somente 4 disponíveis a resposta estaria errada (em azul) e o bit extra chamaríamos de *overflow (em vermelho)*

Histórico da Linguagem



Histórico da Linguagem

Indentação de Código

```c
#include <stdio.h>

int main()
{
    int x;
    printf ("Digite o valor de X..: ");
    scanf ("%d", &x);
    if (x%2 == 0)
        printf ("X é Par\n");
    else
        printf ("X é Ímpar\n");
    system ("pause");
}
```

```c
#include <stdio.h>

int main()
{
int x;
printf ("Digite o valor de X..: ");
scanf ("%d", &x);
if (x%2 == 0)
printf ("X é Par\n");
else
printf ("X é Ímpar\n");
system ("pause");
}
```

Linguagem C

Termine todos os comandos com **;**

Quando ocorrer um erro de compilação, dê um duplo clique sobre a mensagem de erro para destacar o comando errado no programa

Verifique também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o **;**

Use comentários, iniciados por // ou entre /* */

```
/* isto é um comentário */
// isto também é um comentário
```

Linguagem C

A linguagem C é case-sensitive

Isso significa que há diferenças entre:

Area

AREA

arEa

Estrutura Básica de um Programa

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

Pré-Processador

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

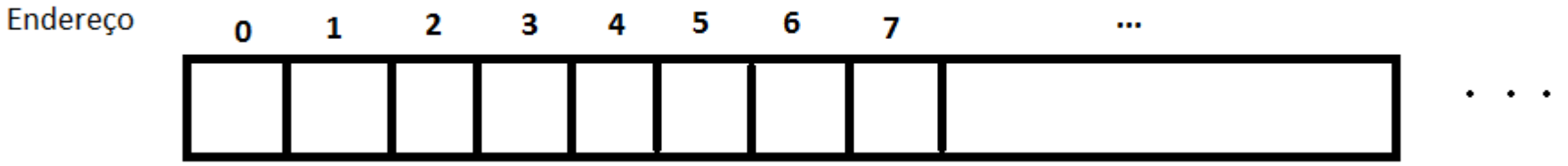
Variáveis

Imagine que a memória RAM do computador é um conjunto de espaços muito grande de armazenamento de *bits*

O endereço de cada variável seria o número que representa a posição do primeiro *bit* que é utilizado

Ex

- Se existissem apenas 20 espaços e um inteiro de 8 *bits* fosse alocado no começo da memória ele iria ocupar um espaço do *bit* 0 ao *bit* 7



Variáveis

Espaços de memória reservados para armazenar valores

- Númericos
- Caracteres
- Funções Lógicas

São identificados por nomes

Type	*Value Range*	*Comments*
char	–128 to 127	
unsigned char	0 to 255	
int	–32,768 to 32,767	16-bit
	–2,147,483,648 to 2,147,483,647	32-bit
unsigned int	0 to 65,535	16-bit
	0 to 4,294,967,295	32-bit
short int	–32,768 to 32,767	
unsigned short int	0 to 65,535	
long int	–2,147,483,648 to 2,147,483,647	
unsigned long int	0 to 4,294,967,295	
float	1.17×10^{-38} to 3.40×10^{38}	6-digit precision
double	2.22×10^{-308} to 1.79×10^{308}	15-digit precision

Variáveis

As variáveis utilizadas em programas representam espaços de memória reservados onde serão armazenados *bits*

Exemplo – quando declaramos *int num1* dizemos para o sistema operacional "reserve um espaço na memória suficiente para armazenar um inteiro"

Este espaço possui um endereço específico, cujo valor inicial pode ser acessado com

&num1 = 0x096FA34 (endereço de memória)

Atribuição

Atribui o valor da direita à variável da esquerda

O valor pode ser:

- uma *constante*, uma variável ou uma expressão

Exemplos

```
x = 4;  // lemos: x recebe 4
y = x + 2; // lemos: y recebe (x mais 2)
y = y + 4; // lemos: y recebe (y mais 4)
valor = 2.5;
sexo = 'F' // constantes devem estar entre aspas simples (apóstrofe)
```

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

Comando de saída de dados

%lf double

Em linguagem C a saída de dados utiliza o comando

PRINTF

A forma de utilização é:

printf(“o que deve ser impresso” , lista de variáveis)

Para limitar as casas decimais %.nf onde n é o número de casas decimais -> %.2f duas casas decimais

Código de formatação	Descrição
%c	Caracteres simples
%d	Inteiros decimais com sinal
%I	Inteiros decimais com sinal
%e	Notação cientifica (e minúsculo)
%E	Notacão cientifica (E maiúsculo)
%f	Ponto flutuante decimal
%g	Usa %e ou %f (qual for mais curto)
%G	Isa %E ou %F (qual for mais curto)
%o	Octal sem sinal
%s	Cadeia de caracteres
%u	Inteiros decimais sem sinal
%x	Hexadecimal sem sinal (letras minúsculas)
%X	Hexadecimal sem sinal (letras maiúsculas)
%%	Escreve o símbolo de porcentagem (%)

Comando de entrada de dados

Em linguagem C a saída de dados utiliza o comando

SCANF

A forma de utilização é:

scanf(“o que deve ser impresso” , lista de endereços)

Código de formatação	Descrição
%c	Caracteres simples
%d	Inteiros decimais com sinal
%I	Inteiros decimais com sinal
%e	Notação cientifica (e minúsculo)
%E	Notacão cientifica (E maiúsculo)
%f	Ponto flutuante decimal
%g	Usa %e ou %f (qual for mais curto)
%G	Isa %E ou %F (qual for mais curto)
%o	Octal sem sinal
%s	Cadeia de caracteres
%u	Inteiros decimais sem sinal
%x	Hexadecimal sem sinal (letras minúsculas)
%X	Hexadecimal sem sinal (letras maiúsculas)
%%	Escreve o símbolo de porcentagem (%)

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

Observação

OBS: o operador "/" (divisão) terá um resultado inteiro se os dois operandos forem inteiros. Para um resultado real, um dos dois operandos deve ser real (ou os dois)

Exemplo:

```
int X,Y;
float Z,U,T;
X=2; Y=3; U=3;
Z=X/Y;    // Z terá o valor zero
T=X/U;    // T terá o valor 0.666667
```

Chamadas de sistema

É possível mandar comandos para o prompt ou shell do sistema

```
#include<stdlib.h>
int system (const char *command);
```

Exemplo: system(“pause”)

Chamadas de Sistema

Para passar valores de variáveis com a função system( ) é preciso montar a string. Abaixo um exemplo que usa a função sprintf( ) para formatar o comando "cat /etc/passwd".

```
char dir[20] = "/etc/passwd";
char comando[100];
sprintf(comando, "cat %s", dir);
ret = system(comando);
```

O que é lógica?

Não existe definição formal

Seria uma maneira de representar argumentos e baseado no fato de serem falsos ou verdadeiros poder fazer inferências sobre assuntos (concluir coisas)

Proposições

São fatos, sentenças, que podem ser verdadeiras ou falsas

- Exemplos

- A sala é azul
- A grama é verde
- Programar é fácil =)

Lógica

Existem operações que podem ser feitas sobre conjuntos de proposições e que resultam em expressões cujo resultado é verdadeiro ou falso

Para saber como operar os conjuntos iremos aprender os operadores

Os operadores lógicos são como operadores matemáticos

- Soma
- Multiplicação
- Etc...

Negação

p	$\neg p$
V	F
F	V

Conjunção

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Disjunção

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Condicional

p	q	$p \Rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Equivalência

p	q	$p \Leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Vamos analisar uma sentença

$$((p \vee q) \Rightarrow (p \wedge q))$$

Porque aprendemos isso?

Cada estrutura de repetição e cada estrutura condicional que utilizarmos trabalham com esta ideia

Temos na linguagem C os conectores

- Conjunção - &&
- Disjunção - ||
- Equivalência - ==
- Diferença - !=

Exemplo em linguagem C

```
if ((x > 3 )&&(y < 4))

if((x > 3) || (y < 4))

x == 3

x != 3
```

O comando IF

há duas variações do comando if:

1 - **if** (condição) comando-if **else** comando-else

2 - **if** (condição) comando-if

O comando IF

Os comandos if e if-else são instruções que permitem a execução condicional de outros comandos.

Na forma completa, if-else, o comando-if é executado quando a condição é verdadeira, caso contrário, o comando-else é executado.

Há ocasiões em que o else é desnecessário, e por isso a linguagem C permite a outra construção if (sem o else) desse comando.

Quando temos mais de um comando dentro do comando if é necessário utilizar { }

O comando IF

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

int main ()
{
  int num1, num2, maior;
  printf("Entre com 2 numeros inteiros");
  scanf("%d %d", &num1, &num2);

  if (num1 > num2)
    maior = num1;
  else
    maior = num2;

  printf("O maior numero e: %d", maior);
  system("pause");
  return 0;
}
```

O comando IF

```
if ((num1 <= num2) && (num1 <= num3))
  {
    /* num1 é o menor, basta comparar num2 e num3 */
    if (num2 <= num3)
      printf("a ordem é %d %d %d\n", num1, num2, num3);
    else
      printf("a ordem é %d %d %d\n", num1, num3, num2);
  }
else if (num2 <= num1 && num2 <= num3)
  {
    /* num2 é o menor, basta comparar num1 e num3 */
    if (num1 <= num3)
      printf("a ordem é %d %d %d\n", num2, num1, num3);
    else
      printf("a ordem é %d %d %d\n", num2, num3, num1);
  }
else /* complete este trecho de programa */
```

Conceitos Básicos Linguagem C

PROF. MAURÍCIO A DIAS

MACDIASPAE@GMAIL.COM