## Representação Numérica

- Números de 16 bits
  - 1 bit para sinal
  - 15 bits para número
  - Faixa de representação: $[-(2^{15} - 1), (2^{15} – 1)]$
  - Números negativos representados usando complemento de dois (inverte dígitos binários e soma 1)

| Decimal | Binário | Codificado |
|---|---|---|
| 7 | 0111 | 0111 |
| 6 | 0110 | 0110 |
| 5 | 0101 | 0101 |
| 4 | 0100 | 0100 |
| 3 | 0011 | 0011 |
| 2 | 0010 | 0010 |
| 1 | 0001 | 0001 |
| 0 | 0000 | 0000 |
| -1 | -0001 | **1**111 |
| -2 | -0010 | **1**110 |
| -3 | -0011 | **1**101 |
| -4 | -0100 | **1**100 |
| -5 | -0101 | **1**011 |
| -6 | -0110 | **1**010 |
| -7 | -0111 | **1**001 |

  - Adição e subtração feitas através da soma

```
Exemplo: p/ número de 5 bits
  Decimal  Binário   Representação
     12d    01100b       01100b
    -10d   -01010b      +10110b
      2d        ?b       00010b
```

  - Multiplicação e Divisão otimizadas por compiladores para potências de 2
    - $X \cdot 2^k = X$ deslocado de k casas à esquerda
    - $X / 2^k = X$ deslocado de k casas à direita
    - Associações: $X \cdot 10 = X \cdot (2^3 + 2^1)$

```
Exemplo: p/ número de 8 bits
 Decimal   Binário     Representação
    12d    00001100b     0000 1100b
  x 8d   x 00001010b   x 00001000b
   96d            ?b     01100000b
```

## *Álgebra das Proposições*

- Desenvolvido no século XIX por George Boole.
- 1930 – Alan Turing mostra que com a álgebra booleana apenas 3 funções lógicas são necessárias para determinar se uma sentença é verdadeira ou falsa: E, OU e NÃO.

| ***x*** | ***y*** | ***x* E *y*** **( $x \wedge y$ )** | ***x* OU *y*** **( $x \vee y$ )** | **NÃO *x*** **( $\neg x$ )** | **NÃO *y*** **( $\neg y$ )** |
|---|---|---|---|---|---|
| V | V | V | V | F | F |
| V | F | F | V | F | V |
| F | V | F | V | V | F |
| F | F | F | F | V | V |

**Exercício:** Dada a função $f = (x \wedge \neg y) \vee (\neg x \wedge y)$, encontrar a sua tabela verdade:

| $x$ | $y$ | $\neg y$ | $\neg x$ | $(x \wedge \neg y)$ | $(\neg x \wedge y)$ | $f = (x \wedge \neg y) \vee (\neg x \wedge y)$ |
|---|---|---|---|---|---|---|
| V | V | F | F | F | F | F |
| V | F | V | F | V | F | V |
| F | V | F | V | F | V | V |
| F | F | V | V | F | F | F |

**Exercício:** Dada a tabela verdade a seguir, encontre a função *f(x, y, z)* correspondente:

| $x$ | $y$ | $z$ | $f(x, y, z)$ | |
|---|---|---|---|---|
| V | V | V | F | = produto das somas |
| V | V | F | **V** | **= soma dos produtos** |
| V | F | V | F | = produto das somas |
| V | F | F | **V** | **= soma dos produtos** |
| F | V | V | **V** | **= soma dos produtos** |
| F | V | F | **V** | **= soma dos produtos** |
| F | F | V | F | = produto das somas |
| F | F | F | F | = produto das somas |

$f = (x \wedge y \wedge \neg z) \vee (x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge y \wedge z) \vee (\neg x \wedge y \wedge \neg z)$

$f = (x \wedge \neg z) \wedge (\cancel{y \wedge \neg y}) \vee (\neg x \wedge y) \wedge (\cancel{z \wedge \neg z})$

$f = (x \wedge \neg z) \vee (\neg x \wedge y)$

# Introdução à Programação

## *Algoritmo*

- **Ação** é um *acontecimento* que, partindo de um *estado inicial* e após um período de tempo *finito*, atinge um *estado final previsível* e *bem definido*.
- **Algoritmo** é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações.

Na vida quotidiana, os algoritmos são encontrados freqüentemente quando usamos um eletrodoméstico, preparamos uma receita de bolo, preenchemos uma declaração de impostos, pagamos uma conta de água/luz nos correios, etc.

Como você implementaria uma algoritmo para fazer uma ligação a partir de um telefone público?

```
Algoritmo // Ligação em um telefone público

    Fazer ligação;

fim do algoritmo
```

O algoritmo anterior resume todas as ações envolvidas no processo de realizar uma ligação a partir de um telefone público em uma única: *fazer a ligação*. No entanto, um algoritmo só é considerado **completo** quando seus comandos são de total entendimento do seu destinatário. Se não for, terá de ser desdobrado em novos comandos, constituindo um **refinamento** do algoritmo inicial.

```
Algoritmo-Ref1 // Ligação em um telefone público (1º Refinamento)

    Tirar o telefone do gancho;
    Introduzir o cartão;
    Teclar o número desejado;
    Conversar;
    Colocar o telefone no gancho;
    Retirar o cartão;

fim do algoritmo
```

Este primeiro refinamento já explicita todo o processo de realização de uma ligação. Mas e se o número discado estiver ocupado? Ou se o cartão não tiver nenhuma unidade? Ou ainda, se ao tirar o telefone do gancho não for ouvido o sinal de linha por alguma eventualidade?

```
Algoritmo-Ref2 // Ligação em um telefone público (2º Refinamento)

    Tirar o telefone do gancho;
    Se der sinal de linha então
        Introduzir o cartão;
        Se o cartão tiver pelo menos uma unidade então
            Teclar o número desejado;
            Se der o sinal de chamar então
                Conversar;
            fim do Se
            Colocar o telefone no gancho;
            Retirar o cartão;
        fim do Se
    fim do Se

fim do algoritmo
```

Com este último algoritmo, a possibilidade de seu conjunto de ações ser exercutado em um tempo finito é bem maior. Isto porque várias eventualidades condicionais foram consideradas evitando, por exemplo, que a pessoa tente discar o número quando seu cartão não possui créditos. No entanto, é necessário aumentar as chances de sucesso na efetuação da chamada tentando, para isso, eliminar todas as barreiras a ela. Por exemplo, comprar outro cartão caso o atual não possua nenhuma unidade, refazer a ligação caso o sinal de chamada não possa ser ouvido, etc. Esta persistência requer que o algoritmo se auto-execute várias vezes até realizar a ligação.

```
Algoritmo-Final // Ligação em um telefone público (Último Refinamento)

    Enquanto não conseguir fazer a ligação e conversar faça
        Tirar o telefone do gancho;
        Se der sinal de linha então
            Introduzir o cartão;
            Se o cartão tiver pelo menos uma unidade então
                Teclar o número desejado;
                Se der o sinal de chamar então
                    Conversar;
                fim do Se
                Colocar o telefone no gancho;
                Retirar o cartão;
            fim do Se

            Senão
                Retirar o cartão;
                Colocar o telefone no gancho;
                Comprar um cartão novo;
            fim do Senão
        fim do Se

        Se está com o telefone na mão
            Colocar o telefone no gancho;
        fim do Se
    fim do Enquanto

fim do algoritmo
```

## *Etapas para o Desenvolvimento de um programa*

- *Criar o código fonte*
  - Linguagem humana
  - Conjunto de sentenças ou comandos para instruir o computador a realizar as tarefas desejadas
- *Compilar o código fonte ⇒ código objeto (.OBJ)*
  - Linguagem de máquina
  - Em Unix: `cc programa.c` (gera programa *.O*)
- *Ligar o código objeto ⇒ arquivo executável (.EXE)*
  - *Linker* gera arquivo executável ligando o código objeto compilado com os códigos objetos das bibliotecas usadas pelo programador (ex.: `stdlib.h`, `math.h`)
  - No *Mingw* e no *GCC* do Linux compila e liga: `gcc programa.c –o programa.exe`
- *Executar o programa*

## *História da Linguagem de Programação C*

- Linguagem B
  - criada por Ken Thompson da Bell Telephone Laboratories
- Linguagem C
  - sucessora da linguagem B
  - criada por Dennis Ritchie na Bell (1972)
  - propósito: criação do sistema operacional UNIX
  - problema: divergências de implementações do C
  - solução: criação do padrão C ANSI (1983)
    - programa em C ANSI pode ser compilado em um compilador C ANSI não importando a máquina na qual irá rodar (evitar uso de bibliotecas de uma plataforma específica, usar diretivas do compilador quando necessário, adotar padrão ANSI, etc.)
  - 1978 – Kernighan e Ritchie fazem o livro "The C Programming Language", conhecido como K&R
  - 1999 – C99: publicação da norma ISO para C ANSI, incluindo novas características (algumas de C++)
    - tipo `bool`
    - `void`
    - `inline`
    - comentários usando `//`

## *Primeiro programa em C*

- Programa que escreve a frase "Bom dia!" na tela

```
#include <stdio.h>

main()
{
   printf("Bom dia!\n");
   return 0;
}
```

## *Características da Linguagem C*

- Programação estruturada
  - Bloco simples
  - Compartimentalização (secciona e esconde do resto do programa instruções necessárias a uma tarefa)
- Declaração de variáveis
  - Para o compilador conhecer espaço de memória gasto
- Funções – blocos de programas
  - Permite esconder parte do código e variáveis (encapsulamento)
  - Saber o que as rotinas fazem; não como fazem.
- Testes de condição
- Laços
- Sensível ao caso (variavel ≠ Variavel)

## *Formato de um programa*