

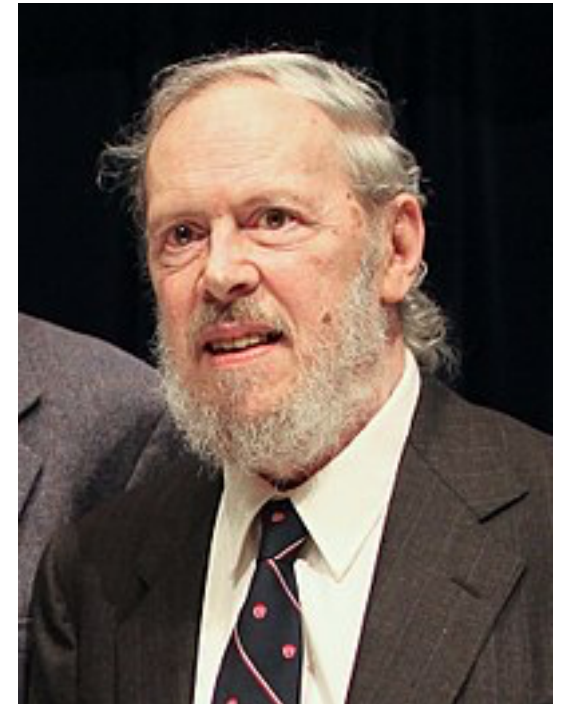
ALGORITMOS

AULA 01

Baseado nas aulas do Prof. Jorgiano Vidal

LINGUAGEM C

- ◆ Uma das grandes vantagens do C é que ele possui tanto características de "alto nível" quanto de "baixo nível".
- ◆ Linguagem de propósito geral
- ◆ criada em 1972,
por Dennis Ritchie Dennis



LINGUAGEM DE PROGRAMAÇÃO

- ◆ Uma **linguagem de programação** é um método padronizado para comunicar instruções para um computador
- ◆ Pode ser vista como um conjunto de palavras chaves e regras
 - Palavras chaves: while, if, for
- ◆ **Regras**
 - Avaliação de expressão: $a=b+c$
 - Condicional: if (EXPR) instrução

LINGUAGEM DE PROGRAMAÇÃO

Exemplo de Programa em C

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

```
int main(int argc, char ** argv) {  
    printf("Hello World!!!");  
    return 0;  
}
```

COMPILAÇÃO VS INTERPRETAÇÃO VS HÍBRIDO

◆ Compilação

- Traduz o programa em (LM) Linguagem de Máquina
- Executa-se o programa em LM: Mais rápido
- Código gerado nativo do processador

◆ Interpretação

- Um programa (interpretador) processa as instruções
- Execução mais lenta
- Necessário ter interpretador para executar

◆ híbrido

- O código é compilado
- Porém o resultado é algo que precisa ser interpretado.

A LINGUAGEM C

◆ Um programa em C

- Conjunto de funções/procedimentos
- Conjunto de variáveis locais e globais
- Ponto de início é a função main

◆ Retorna inteiro

Possui dois parâmetros

- int: quantidade de parâmetros
- *char[]: array com os parâmetros (string)

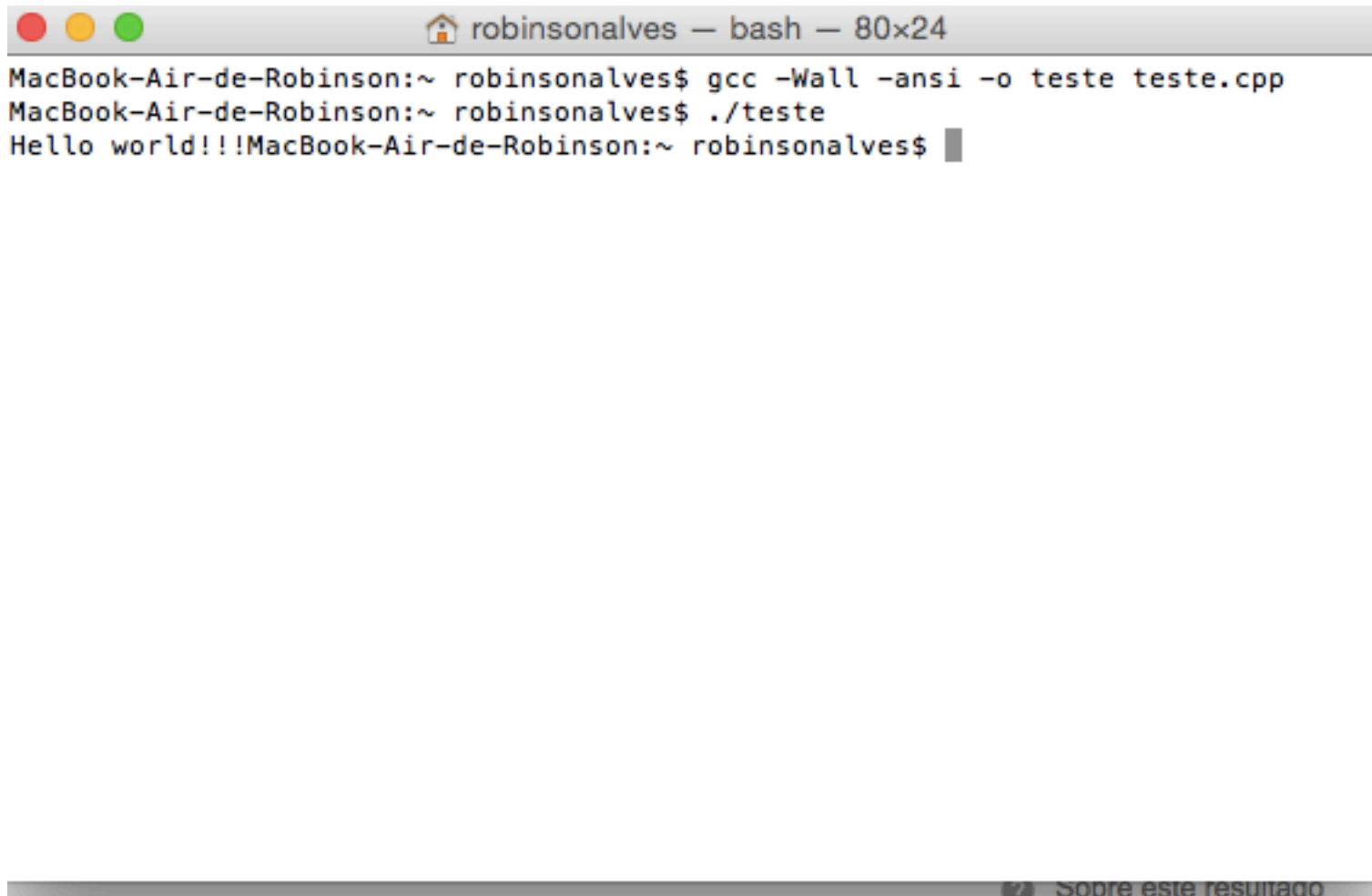
O COMPILADOR

- ◆ GCC - Gnu Compiler Collection
 - <http://gcc.gnu.org>
- ◆ clang - C Language
 - <http://clang.llvm.org>
- ◆ Existem outros

A COMPILAÇÃO

- ◆ Nome do comando: gcc
- ◆ Argumentos
 - Wall: mostra todos os "warnings"
 - ansi: verifica se o código respeita as regras do C ansi
 - o EXEC: gera o executável como nome EXEC
- ◆ Se não ocorrer erros/warnings o compilador terminará sem mostrar mensagens

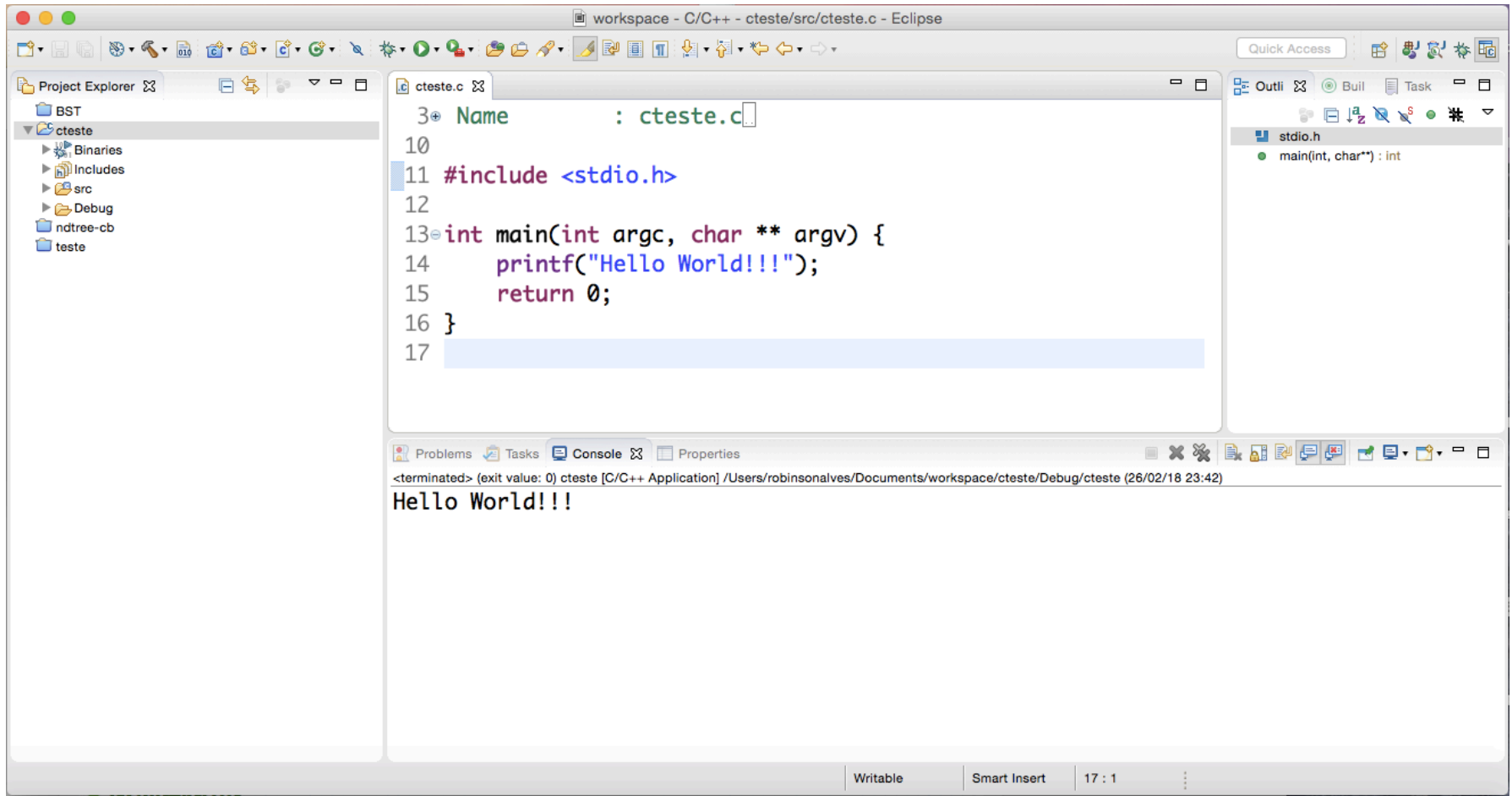
A COMPILAÇÃO - SHELL



```
MacBook-Air-de-Robinson:~ robinsonalves$ gcc -Wall -ansi -o teste teste.cpp
MacBook-Air-de-Robinson:~ robinsonalves$ ./teste
Hello world!!!MacBook-Air-de-Robinson:~ robinsonalves$
```

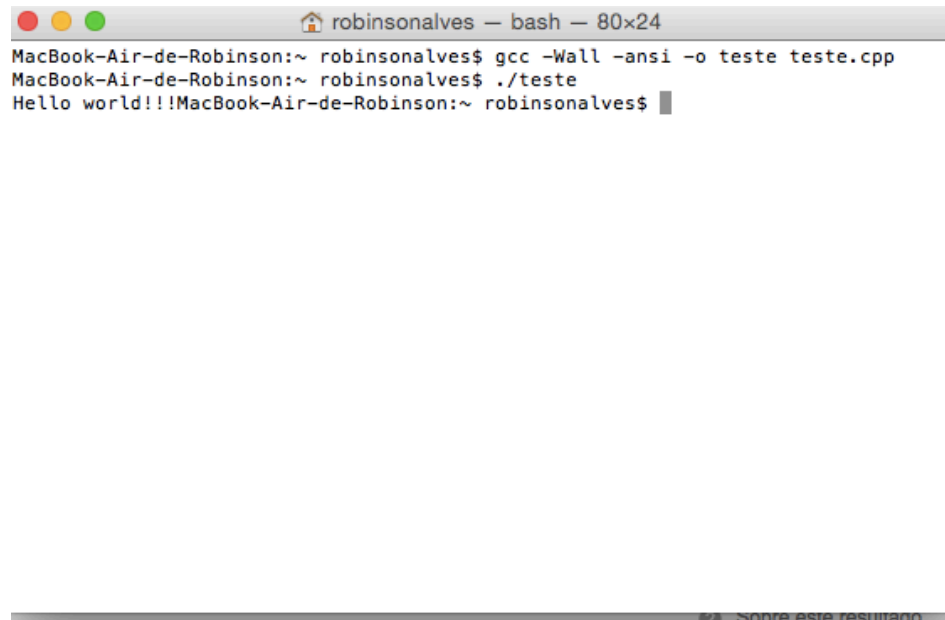
The image shows a terminal window with a title bar that reads "robinsonalves - bash - 80x24". The terminal content shows the compilation of a C++ program named "teste.cpp" into an executable named "teste" using the GCC compiler with flags "-Wall" and "-ansi". The program is then executed with the command "./teste", resulting in the output "Hello world!!!". The terminal prompt returns to the shell. At the bottom right of the terminal window, there is a small icon and the text "Sobre este resultado".

A COMPILAÇÃO - ECLIPSE



A EXECUÇÃO

- ◆ Depende do sistema operacional
 - no linux escreve-se o nome do programa na console
- ◆ Caminho pode ser absoluto ou relativo



```
MacBook-Air-de-Robinson:~ robinsonalves$ gcc -Wall -ansi -o teste teste.cpp
MacBook-Air-de-Robinson:~ robinsonalves$ ./teste
Hello world!!!MacBook-Air-de-Robinson:~ robinsonalves$
```

The image shows a terminal window with a title bar that reads "robinsonalves - bash - 80x24". The terminal content shows the compilation of a C++ program named "teste.cpp" into an executable named "teste" using the gcc compiler with flags -Wall and -ansi. The program is then executed with ./teste, resulting in the output "Hello world!!!". The terminal prompt returns to the user's home directory.

A EXECUÇÃO

- ◆ Depende do sistema operacional
 - no linux escreve-se o nome do programa na console
- ◆ Caminho pode ser absoluto ou relativo

A EXECUÇÃO

DEMONSTRAÇÃO

FUNÇÕES EM C

```
int soma(int a, int b) {  
    corpo  
}
```

FUNÇÕES EM C

- ◆ Uma função nada mais é do que uma subrotina usada em um programa.
- ◆ Na linguagem C, denominamos função a um conjunto de comandos que realiza uma tarefa específica em um módulo dependente de código.
- ◆ A função é referenciada pelo programa principal através do nome atribuído a ela.

FUNÇÕES EM C

- ◆ A utilização de funções visa modularizar um programa, o que é muito comum em programação estruturada.
- ◆ Desta forma podemos dividir um programa em várias partes, no qual cada função realiza uma tarefa bem definida.

FUNÇÕES EM C

◆ Esqueleto de uma função

```
tipo_de_retorno nome_da_função (parâmetros)
```

```
{
```

```
    instruções;
```

```
    retorno_da_função;
```

```
}
```

FUNÇÕES EM C

◆ Esqueleto de uma função

```
int soma(int a, int b) {  
    int s;  
    s = a + b;  
    return s;  
}
```

```
int soma(int a, int b) { int s; s = a + b; return s;  
}
```

FUNÇÕES EM C

- ◆ Esqueleto de uma função
 - Legibilidade
 - Uma instrução por linha Ponto-e-vírgula no final da linha

TIPOS

- ◆ Principais tipos:
 - char short int long float double
- ◆ Todos os tipos podem ser:
 - signed unsigned
- ◆ OBS: String é um array de char

VARIÁVEIS

- ◆ DEVE ser declarada
 - Declaração define tipo

```
int soma(int a, int b) {  
    int s;  
    s = a + b;  
    return s;  
}
```

VARIÁVEIS

- ◆ DEVE ser declarada
 - Declaração define tipo

```
int soma(int a, int b) {  
    int s;  
    s = a + b;  
    return s;  
}
```

ATRIBUIÇÃO

- ◆ Armazena um valor em uma zona de memória indicada pela variável
 - `VAR = EXPR;`
- ◆ Tipo deve ser compatível
- ◆ Variável deve ser declarada
- ◆ Exemplo:
 - `soma = a+b;`

EXPRESSÕES

- ◆ Lado direito da atribuições
- ◆ Operadores dependem do tipo dos operandos
 - + (soma), - (subtração), * (Multiplicação), / (Divisão inteira e real), % (resto da divisão)
- ◆ CUIDADO: Divisão inteira diferente da divisão real
 - Observar também tipos das variáveis
 $10/3 \neq 10.0/3.0$

MAIN

◆ Todo programa em C começa pela função main

- retorna um inteiro
- Possui dois parâmetros
 - ◆ inteiro com a quantidade de elementos no array de parâmetros
 - ◆ Array de strings

```
#include <stdio.h>  
int main(int argc, char ** argv) {  
    printf("Hello World!!!");  
    return 0;  
}
```

ENTRADA E SAÍDA

◆ Ler dados do teclado

■ Função scanf()

- ◆ Ler inteiro: `scanf("%d",&a);`
- ◆ Ler real: `scanf("%f",&x);`
- ◆ Ler string: `scanf("%s",nome);`

◆ Escrever dados no terminal

■ Função printf

- ◆ `printf("Um texto qualquer");`
- ◆ `printf("a soma de %d e %d é %d",a,b,soma);`
- ◆ `printf("A média foi de %.2f km/h",media);`
- ◆ Importante observar tipos das variáveis

◆ Detalhes serão vistos futuramente

ENTRADA E SAÍDA

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

```
int soma(int a, int b){  
    return a+b;
```

```
}
```

```
int main(int argc, char ** argv) {  
    int a,b;  
    scanf("%d",&a);  
    scanf("%d",&b);  
    printf("Soma é %d",soma(a,b));  
    return 0;
```

```
}
```

ENTRADA E SAÍDA

```
#include <stdio.h> // biblioteca E/S  
int soma(int a, int b); // assinatura da função  
int main(int argc, char ** argv) {  
    int a,b;  
    scanf("%d",&a);  
    scanf("%d",&b);  
    printf("Soma é %d",soma(a,b));  
    return 0;  
}  
int soma(int a, int b){  
    return a+b;  
}
```

Dúvidas

