

# Entrada, Tipos e Armazenamento de Dados.

Prof.: Leonardo Tórtoro Pereira

[leonardop@usp.br](mailto:leonardop@usp.br)

# Dados em C

- C possui 4 tipos de dados principais
  - ◆ *char, int, float, double*
- Cada um é capaz de armazenar dados diferentes, com tamanhos diferentes e intervalo diferente dos números que armazena
- Existem palavras-chave que podem modificar o tamanho e intervalo
  - ◆ *signed, unsigned, short e long*

# Dados em C

→ *char*

- ◆ Armazena um único caractere e requer 1 byte de quase todos os compiladores

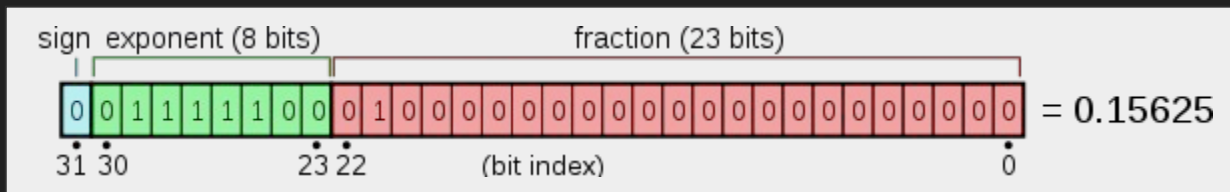
→ *int*

- ◆ Armazena um inteiro e necessita de 4 bytes

# Dados em C

→ *float*

- ◆ Armazena números decimais com precisão única e necessita de 4 bytes

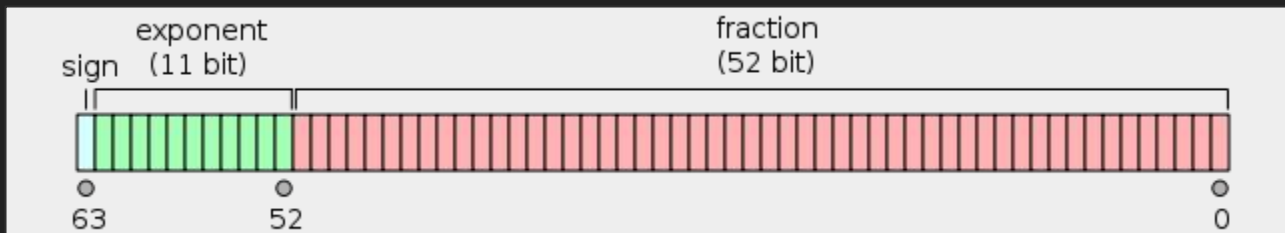


$$(-1)^{b_{31}} \times 2^{(b_{30}b_{29}\dots b_{23})_2 - 127} \times (1.b_{22}b_{21}\dots b_0)_2$$

# Dados em C

→ *double*

- ◆ Armazena números decimais com precisão dupla e necessita de 8 bytes



$$(-1)^{\text{sign}} (1.b_{51}b_{50}\dots b_0)_2 \times 2^{e-1023}$$

# Dados em C

## → *signed*

- ◆ Força a variável a ter um sinal
- ◆ Normalmente é redundante, exceto para *char*

## → *unsigned*

- ◆ Força a variável a ser positiva
- ◆ Aumenta o valor máximo que ela pode guardar

## → *short*

- ◆ Diminui o tamanho (2 bytes) e intervalo de um *int*

# Dados em C

→ *long*

- ◆ Na maioria dos compiladores atuais, não muda nada do *int*
- ◆ Mas ao usar duas vezes (*long long*) aumenta seu tamanho para 8 bytes e seu intervalo

# Verificando tamanho dos dados

```
#include <stdio.h>
int main()
{
    int a = 1;
    char b = 'G';
    double c = 3.14;
    long int d = 123453;
    unsigned int e = 231435242;
    long long int f = 1234533334564;
    //print da variável e seu tamanho
    printf("A character. Value %c and size %lu byte.\n", b, sizeof(char));
    printf("An integer. Value %d and size %lu byte.\n", a, sizeof(a));
    printf("A double. Value %lf and size %lu byte.\n", c, sizeof(double));
    printf("A long int. Value %ld and size %lu byte.\n", d, sizeof(d));
    printf("An unsigned int. Value %u and size %lu byte.\n", e, sizeof(e));
    printf("A long long int. Value %lld and size %lu byte.\n", f, sizeof(f));
    return 0;
}
```



# Constantes

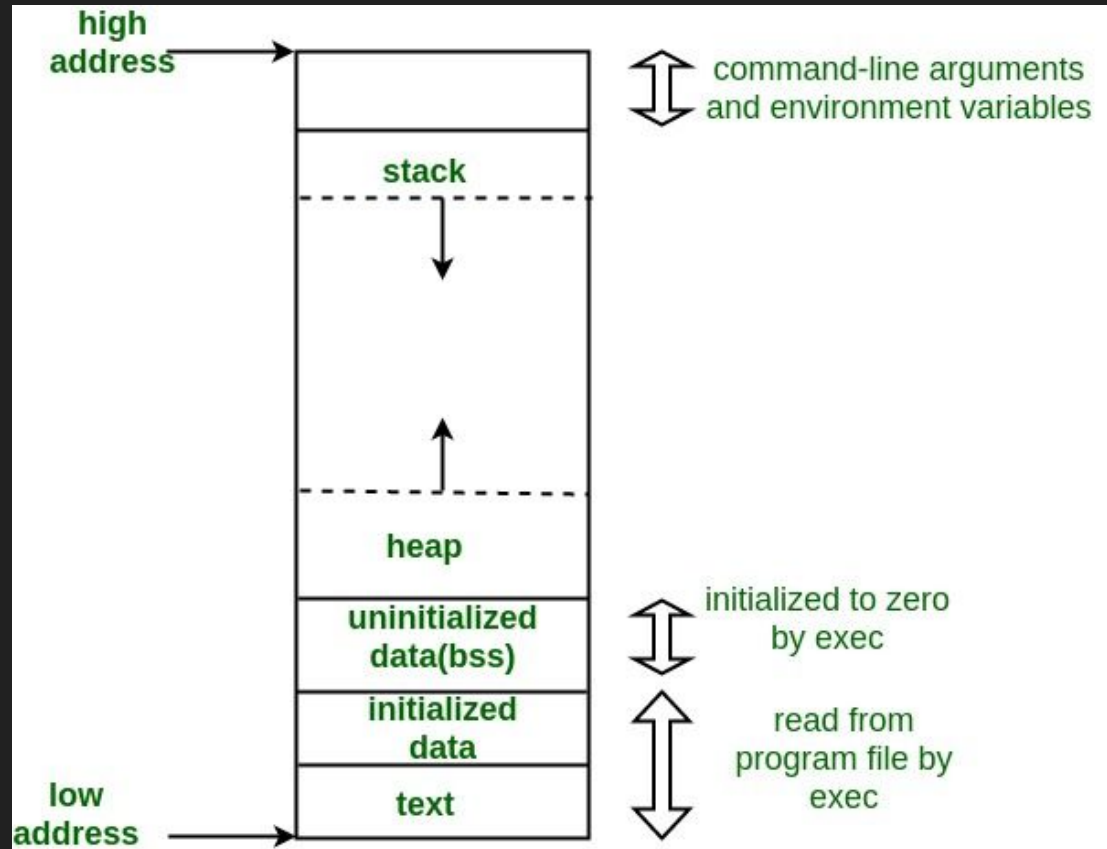
- É possível definir uma constante através da notação *#define*
- Ela também é útil para definir uma variável booleana

# Constantes e Booleano

```
#define BOOL char
#define FALSE 0
#define TRUE 1
int main()
{
    BOOL a = TRUE;
    printf("Booleano = %d", a);

    return 0;
}
```

# Armazenamento de Dados



Fonte: <https://www.geeksforgeeks.org/memory-layout-of-c-program/>

# Memória de um programa

- Segmento de texto
  - ◆ Segmento de código, contém instruções de execução
- Dados inicializados
  - ◆ Variáveis globais e estáticas inicializadas pelo programador
- Dados não inicializados
  - ◆ Dados que não foram inicializados e são inicializados pelo *kernel* para seu correspondente aritmético a 0

# Memória de um programa

## → Stack

- ◆ Onde as variáveis são guardadas, além de informações sobre contexto das funções cada vez que elas são chamadas
- ◆ A função guarda espaço na stack para suas variáveis automáticas e temporárias
- ◆ Toda variável tem seu endereço na *stack*
  - Pode ser acessado com o caractere &

# Memória de um programa

## → Heap

- ◆ Onde a alocação dinâmica de memória ocorre
- ◆ Administrada pelas funções *malloc*, *realloc* e *free*

# Imprimindo endereço da variável

```
#include <stdio.h>
int main()
{
    int a = 1;
    char b = 'G';
    double c = 3.14;
    long int d = 123453;
    unsigned int *z, e = 231435242;
    long long int f = 1234533334564;
    //print do endereço e seu tamanho
    printf("An integer. Address %p and size %lu byte.\n", &a, sizeof(a));
    printf("A character. Address %p and size %lu byte.\n", &b, sizeof(char));
    printf("A double. Address %p and size %lu byte.\n", &c, sizeof(double));
    printf("A long int. Address %p and size %lu byte.\n", &d, sizeof(d));
    printf("An unsigned int. Address %p and size %lu byte.\n", &e, sizeof(e));
    printf("A long long int. Address %p and size %lu byte.\n", &f, sizeof(f));
    z=&a;
    printf("A pointer to a char. Value %u, Address %p, size %lu byte", *z, z, sizeof(z));
    return 0;
}
```



# Entrada e Saída de dados

# Leitura e Entrada de Dados

- A entrada de dados padrão de C é conhecida como *stdin*
- Para ler os dados de lá, é muito comum usar a função *scanf()*, muito similar ao *printf()*
  - ◆ Primeiro, são fornecidas as máscaras dos valores a serem lidos, entre aspas duplas ""
  - ◆ Em seguida, a vírgula e cada ENDEREÇO de variável que receberá os valores fornecidos na entrada, separados por vírgula

# Leitura e Entrada de Dados

```
#include <stdio.h>

int main ()
{
    char str[80], str2[80];
    int i, h;
    float f;

    printf ("Uma string: ");
    scanf ("%79s",str);
    printf ("Outra string: ");
    scanf ("%s",str2);
    printf ("Um int: ");
    scanf ("%d",&i);
    printf("Um float: ");
    scanf("%f", &f);
    printf ("Um hexadecimal: ");
    scanf ("%x",&h);
    printf ("String: %s\nString2: %s\nInteiro: %d\nFloat: %f\nHexa %#x (%d).\n",str, str2, i, f, h, h);
    printf("Float com algumas casas decimais: %.2f", f);
    return 0;
}
```

# Conversão de Tipos

# Conversão de Tipos

- Tipos podem ser convertidos entre si de maneira automática (ou implícita) ou explícita (*casting*)

# Conversão Implícita

- A conversão implícita é feita pelo compilador, geralmente quando mais de um tipo está presente em uma expressão
  - ◆ Tem como objetivo evitar a perda de dados
- Dá um *upgrade* do tipo de dado para o maior tipo de dado entre as variáveis presentes
- Pode haver perda de dados mesmo assim
  - ◆ Perda de sinal e *overflow*

## Implicit Type Conversion



Fonte: <https://www.geeksforgeeks.org/type-conversion-c/>

# Conversão Implícita

```
//https://www.geeksforgeeks.org/type-conversion-c/  
int main()  
{  
    int x = 10;    // integer x  
    char y = 'a';  // character c  
  
    // y implicitly converted to int. ASCII  
    // value of 'a' is 97  
    x = x + y;  
  
    // x is implicitly converted to float  
    float z = x + 1.0;  
  
    printf("x = %d, z = %f", x, z);  
    return 0;  
}
```



# Conversão Explícita

- É feita pelo programador. O usuário pode mudar o tipo de uma variável para o outro que desejar
- Sintaxe: (tipo) expressão
- Usado para computar expressões com variáveis de diferentes tipos de dados

# Conversão Explícita

```
int main()
{
    double x = 1.2;
    // Explicit conversion from double to int
    int sum = (int)x + 1;
    printf("sum = %d", sum);
    return 0;
}
```

Exemplo!

# Média Aritmética

```
#include <stdio.h>

int main ()
{
    int numero1, numero2, media;
    scanf("%d", &numero1);
    scanf("%d", &numero2);
    media = (numero1 + numero2)/2;
    printf("%d", media);
    return 0;
}
```

Como Podemos Melhorar?

# Média Aritmética V2

```
#include <stdio.h>
#define TOTALNUMBERS 2.0

int main ()
{
    int numero1, numero2;
    float media;
    scanf("%d", &numero1);
    scanf("%d", &numero2);
    media = (numero1 + numero2)/TOTALNUMBERS;
    printf("%.2f", media);
    return 0;
}
```

# Operadores Bitwise

# Operators in C

	Operator	Type
Unary operator	++, --	Unary operator
Binary operator	+, -, *, /, %	Arithmetic operator
	<, <=, >, >=, ==, !=	Relational operator
	&&,   , !	Logical operator
	&,  , <<, >>, ~, ^	Bitwise operator
	=, +=, -=, *=, /=, %=	Assignment operator
Ternary operator	?:	Ternary or conditional operator



Operadores em C

Fonte: <https://www.geeksforgeeks.org/bitwise-operators-in-c-cpp/>



# Operadores Bitwise

- São operadores que trabalham a nível de bits
- **&**
  - ◆ Realiza a operação AND em cada bit de dois números
  - ◆  $a \& b$
  - ◆ Pode calcular se um número é par ou ímpar
- **|**
  - ◆ Realiza a operação OR em cada bit de dois números
  - ◆  $a | b$

# Operadores Bitwise

```
#include <stdio.h>

int main()
{
    int x = 13;
    (x & 1) ? printf("Odd") : printf("Even");
    return 0;
}
```

# Operadores Bitwise



$\wedge$

- ◆ Realiza a operação XOR em cada bit de dois números
- ◆  $a \wedge b$



$\sim$

- ◆ Realiza a operação NOT em cada bit de um número
- ◆  $\sim a$

# Operadores Bitwise

→ <<

◆ Desloca à esquerda em “b” lugares do número “a”

◆  $a \ll b$

→ >>

◆ Desloca à direita em “b” lugares do número “a”

◆  $a \gg b$

→ Não devem ser usados em números negativos!

→ Não deslocar mais que o tamanho da variável!

# Operadores Bitwise

Funcionam como multiplicação e divisão por 2:

```
#include <stdio.h>
int main()
{
    int x = 20;
    printf("x << 1 = %d\n", x << 1);
    printf("x >> 1 = %d\n", x >> 1);
    return 0;
}
```

# Operadores Bitwise

```
int main() {  
    // a = 5(00000101), b = 9(00001001)  
    unsigned char a = 5, b = 9;  
    // The result is 00000001  
    printf("a = %d, b = %d\n", a, b);  
    printf("a&b = %d\n", a & b);  
    // The result is 00001101  
    printf("a|b = %d\n", a | b);  
    // The result is 00001100  
    printf("a^b = %d\n", a ^ b);  
    // The result is 11111010  
    printf("~a = %d\n", a = ~a);  
    // The result is 00010010  
    printf("b<<1 = %d\n", b << 1);  
    // The result is 00000100  
    printf("b>>1 = %d\n", b >> 1);  
    return 0;  
}
```

# Referências

- [https://en.wikipedia.org/wiki/C\\_data\\_types](https://en.wikipedia.org/wiki/C_data_types)
- <https://www.geeksforgeeks.org/data-types-in-c/>
- <http://www.cplusplus.com/doc/tutorial/variables/>
- <https://www.geeksforgeeks.org/memory-layout-of-c-program/>
- <https://www.learncpp.com/cpp-tutorial/79-the-stack-and-the-heap/>
- <http://www.cplusplus.com/reference/cstdio/scanf/>

# Referências

- <https://www.geeksforgeeks.org/bitwise-operators-in-c-cpp/>
- [https://en.wikipedia.org/wiki/Bitwise\\_operations\\_in\\_C](https://en.wikipedia.org/wiki/Bitwise_operations_in_C)



# Curiosidades

- <https://www.geeksforgeeks.org/interesting-facts-about-data-types-and-modifiers-in-c-cpp/>
- <https://www.geeksforgeeks.org/character-arithmetic-c-c/>
- <https://www.geeksforgeeks.org/static-variables-in-c/>
- <https://www.geeksforgeeks.org/problem-with-scanf-when-there-is-fgetsgetsscanf-after-it/>
- <https://www.geeksforgeeks.org/clearing-the-input-buffer-in-c-c/>
- <https://www.geeksforgeeks.org/type-conversion-c/>