Aula 02 - Introdução C e Entrada/Saída

Prof. Me. Claudiney R. Tinoco

profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Bacharelado em Ciência da Computação (BCC)
Bacharelado em Sistemas de Informação (BSI)

Programação Procedimental (PP)
GBC014 - GSI002

Introdução

- Linguagem de Máquina
  - Computador entende apenas pulsos elétricos;
  - Presença ou não de pulso;
  - 1 ou 0.
- Tudo no computador deve ser descrito em termos de 1's ou 0's (binário)
  - Difícil para humanos ler ou escrever;
  - $00011110 = 30$.

- Linguagem Assembly
  - Uso de mnemônicos;
  - Conjunto de 0's e 1's é agora representado por um código;
  - $10011011 \rightarrow$ ADD.
- Linguagem Assembly - Problemas
  - Requer programação especial (Assembly);
  - Conjunto de instruções varia com o computador (processador);
  - Ainda é muito difícil programar.

- Linguagens de Alto Nível
  - Programas são escritos utilizando uma linguagem parecida com a linguagem humana;
  - Independente da arquitetura do computador;
  - Mais fácil programar;
  - Uso de compiladores.

- Primórdios
  - Uso da computação para cálculos de fórmulas;
  - Fórmulas eram traduzidas para linguagem de máquinas;
  - Por que não escrever programas parecidos com as fórmulas que se deseja computar?

- FORTRAN (FORmula TRANsform)
  - Em 1950, um grupo de programadores da IBM liderados por John Backus produz a versão inicial da linguagem;
  - Primeira linguagem de alto nível.
- Várias outras linguagens de alto nível foram criadas
  - gol-60, Cobol, Pascal, etc.

- Uma das mais bem sucedidas foi uma linguagem chamada C
  - Criada em 1972 nos laboratórios por Dennis Ritchie;
  - Revisada e padronizada pela ANSI em 1989;
    - ANSI: American National Standards Institute
    - Padrão mais utilizado

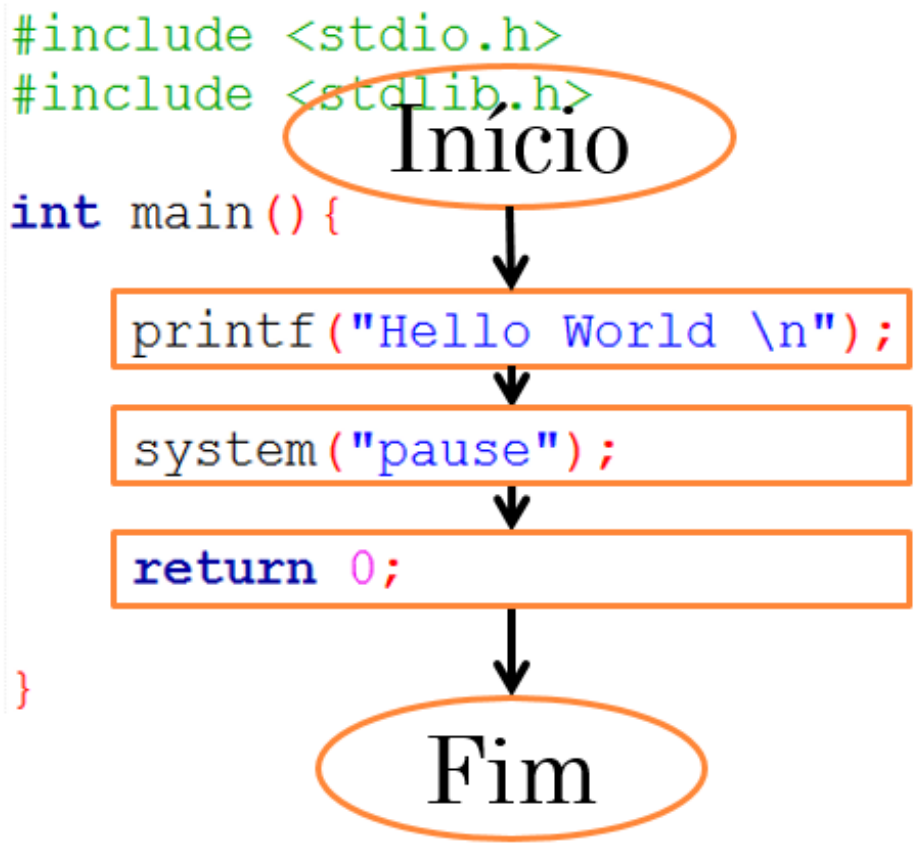
Primeiro programa em C

- Por que escrevemos programas?
  - Temos dados ou informações que precisam ser processados;
  - Esse processamento pode ser algum cálculo ou pesquisa sobre os dados de entrada;
  - Desse processamento, esperamos obter alguns resultados (Saídas);

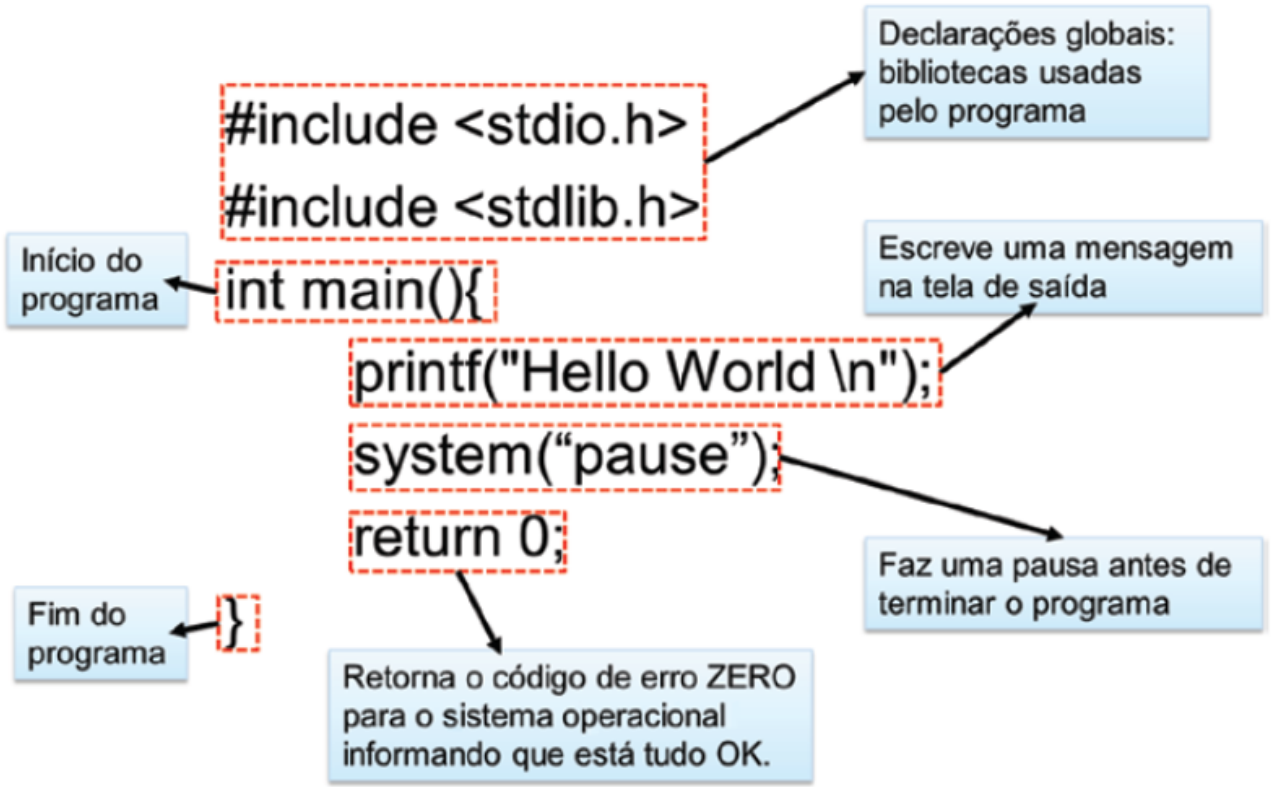
Primeiro programa em C

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

int main(){

    printf("Hello World \n");

    system("pause");

    return 0;

}
```

Primeiro programa em C



Primeiro programa em C



Conceitos Básicos C

Comentários

- Permitem adicionar uma descrição sobre o programa. São ignorados pelo compilador.

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

int main(){
    /*
    A função printf serve
    para escrever na tela
    */
    printf("Hello World \n");

    //faz uma pausa no programa
    system("pause");

    return 0;

}
```

Variáveis

- Matemática
  - é uma entidade capaz de representar um valor ou expressão;
  - pode representar um número ou um conjunto de números
  - $f(x) = x^2$

Variáveis

- Computação
  - Posição de memória que armazena uma informação
  - Pode ser modificada pelo programa
  - Deve ser **definida** antes de ser usada

Declaração de variáveis

- Precisamos informar ao programa quais dados queremos armazenar
- Precisamos também informar o que são esses dados (qual o tipo de dado)
  - Um nome de uma pessoa
    - Uma cadeia de caracteres (“Claudiney” - 9 caracteres)
  - O valor da temperatura atual
    - Um valor numérico (com casas decimais)
  - A quantidade de alunos em uma sala de aula
    - Um valor numérico (número inteiro positivo ou zero)
  - Se um assento de uma aeronave está ocupado
    - Um valor lógico (ocupado: verdadeiro / desocupado: falso)

Variáveis

- Declaração de variáveis em C
  - <tipo de dado> nome-da-variável;
- Propriedades
  - Nome
    - Pode ter um ou mais caracteres
    - Nem tudo pode ser usado como nome
  - Tipo
    - Conjunto de valores aceitos
  - Escopo
    - global ou local

Variáveis

- Nome
  - Deve iniciar com letras ou underscore (_);
  - Caracteres devem ser letras, números ou underscores;
  - Palavras chaves não podem ser usadas como nomes;
  - Letras maiúsculas e minúsculas são consideradas diferentes

Variáveis

- Nome
  - Não utilizar espaços nos nomes
    - Exemplo: nome do aluno, temperatura do sensor,
  - Não utilizar acentos ou símbolos
    - Exemplos: garça, tripé, o,Θ
  - Não inicializar o nome da variável com números
    - Exemplos: 1A, 52, 5ª
  - Underscore pode ser usado
    - Exemplo: nome_do_aluno : caracter
  - Não pode haver duas variáveis com o mesmo nome

Lista de Palavras Reservadas (32)							
if	break	case	char	const	continue	default	do
else	while	for	float	goto	double	extern	auto
int	return	long	short	signed	sizeof	register	static
void	switch	union	enum	volatile	unsigned	typedef	struct

Variáveis

- Quais nomes de variáveis estão corretos:
  - Contador
  - contador1
  - comp!
  - .var
  - Teste_123
  - _teste
  - int
  - int1
  - 1contador
  - -x
  - Teste-123
  - x&

Variáveis

- Corretos:
  - Contador, contador1,Teste_123, _teste, int1
- Errados
  - comp!, .var, int, 1contador, -x, Teste-123, x&

Variáveis

- Tipo
  - Define os valores que ela pode assumir e as operações que podem ser realizadas com ela
- Exemplo
  - tipo **int** recebe apenas valores inteiros
  - tipo **float** armazena apenas valores reais

Tipos básicos em C

- **char**: um byte que armazena o código de um caractere do conjunto de caracteres local
  - ***caracteres sempre ficam entre 'aspas simples'!***

```
char sexo; // pode receber'M' ou 'F'
char UnidadeTemperatura; //pode receber'C' para Celsius
                         //ou 'F' para Fahrenheit
char opcoes; // pode ser '1', '2' ,'3' ou '4'
```

- **int**: um inteiro cujo tamanho depende do processador, tipicamente 16 ou 32 bits

```
int NumeroAlunos;
int Idade;
int NumeroContaCorrente;
int N = 10; // o variável N recebe o valor 10
```

Tipos básicos em C

- Números reais
  - Tipos: *float*, *double* e *long double*
  - A parte decimal usa **ponto** e **não vírgula**!
  - **float**: um número real com precisão simples

```
float Temperatura; // por exemplo, 23.30
float MediaNotas; // por exemplo, 7.98
float TempoTotal; // por exemplo, 0.0000000032 (s)
```

  - **double**: um número real com precisão dupla
    - Números muito grandes ou muito pequenos

```
double DistanciaGalaxias; // número muito grande
double MassaMolecular; // em Kg, número muito pequeno
double BalancoEmpresa; // valores financeiros
```

Tipos básicos em C

- Números reais
  - Pode-se escrever números reais usando notação científica

```
double TempoTotal = 0.000000003295;

// notação científica
double TempoTotal = 3.2950e-009;
```

equivale à $3{,}295 \times 10^{-9}$

Palavra chave	Tipo	Armazenamento	intervalo	Precisão
char	Caracter	1 byte	-128 to 127	
signed char	Caractere com sinal	1 byte	-128 to 127	
unsigned char	Caractere sem sinal	1 byte	0 to 255	
int	Inteiro	2 ou 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	
signed int	Inteiro com sinal	2 ou 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	
unsigned int	Inteiro sem sinal	2 ou 4 bytes	0 to 65,535 or 0 to 4,294,967,295	
short	Inteiro curto	2 bytes	-32,768 to 32,767	
signed short int	Inteiro curto com sinal	2 bytes	-32,768 to 32,767	
unsigned short	Inteiro curto sem sinal	2 bytes	0 to 65,535	
long	Inteiro longo	4 ou 8 bytes	-9223372036854775808 to 9223372036854775807	
signed long int	Inteiro longo com sinal	4 ou 8 bytes	-9223372036854775808 to 9223372036854775807	
unsigned long	Inteiro longo sem sinal	4 ou 8 bytes	0 to 18446744073709551615	
float	Ponto flutuante com precisão simples	4 bytes	1.2E-38 to 3.4E+38	6 casas dec.
double	Ponto flutuante com precisão dupla	8 bytes	2.3E-308 to 1.7E+308	15 casas dec.
long double	Ponto flutuante com precisão dupla longo	10 bytes	3.4E-4932 to 1.1E+4932	19 casas dec.

Atribuição

- Operador de Atribuição: =
  - nome_da_variável = expressão, valor ou constante;

O operador de atribuição "=" armazena o valor ou resultado de uma expressão contida à sua **direita** na variável especificada à sua **esquerda**.

- Ex.:

```
int main( ){
    int x = 5; // x recebe 5
    int y;
    y = x + 3; // y recebe x mais 3

    return 0;
}
```

  - A linguagem C suporta múltiplas atribuições
    - x = y = z = 0;

Comando de saída

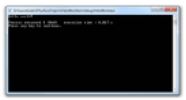
- **printf()**
  - ***print** formatted*
  - Comando que realiza a impressão dos dados do programa na tela

  - O texto a ser escrito deve ser sempre definido entre **"aspas duplas"**

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

int main(){
    printf("Esse texto sera escrito na tela");

    return 0;
}
```

Comando de saída

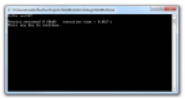
- **printf()**
  - Quando queremos escrever dados formatados na tela usamos a forma geral da função, a qual possui os tipos de saída.
  - Eles especificam o formato de saída dos dados que serão escritos pela função **printf()**.



printf("%tipo_de_saida",expressão);

printf("%tipo1 %tipo2",expressão1, expressão2);

  - Podemos misturar o texto a ser mostrado com os especificadores de formato



printf("texto %tipo_de_saida texto" expressão);

Comando de saída

- **printf()**
  - Especificadores de formato

Alguns tipos de saída	
%c	escrita de um caractere (**char**)
%d ou %i	escrita de números inteiros (**int** ou **char**)
%u	escrita de números inteiros sem sinal (**unsigned**)
%f	escrita de número reais (**float** ou **double**)
%s	escrita de vários caracteres
%p	escrita de um endereço de memória
%e ou %E	escrita em notação científica

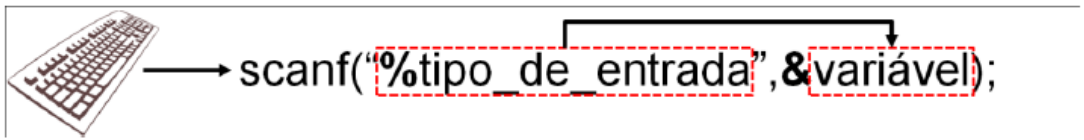
Comando de saída

- **printf()**
  - Exemplos

```c
int main(){
    printf("Esse texto sera escrito na tela");

    int x = 10;
    float y = 20;
    printf("%d",x);

    printf("%d %f",x,y);

    printf("Valor de x eh %d e o de y eh %f",x,y);

    return 0;
}
```

Comando de entrada

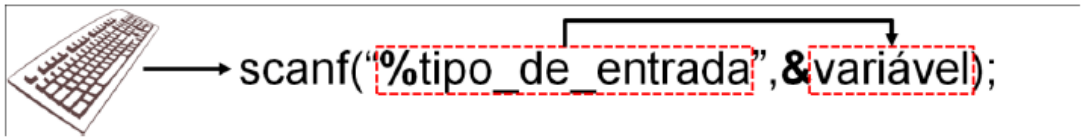
- **scanf()**
  - Comando que realiza a leitura dos dados da entrada padrão (no caso o teclado)
  - scanf("tipo de entrada", lista de variáveis)



  - O tipo de entrada deve ser sempre definido entre **"aspas duplas"**
  - Na linguagem C, é necessário colocar o símbolo & antes do nome de cada variável a ser lida pelo comando **scanf()**.
    - O símbolo & indica qual é o endereço da variável que vai receber os dados lidos

Comando de entrada

- **scanf()**
  - Comando que realiza a leitura dos dados da entrada padrão (no caso o teclado)
  - scanf("tipo de entrada", lista de variáveis)



  - O tipo de entrada deve ser sempre definido entre **"aspas duplas"**

```c
int main(){
    int x;
    scanf("%d",&x);

    printf("Valor de x: %d",x);

    return 0;
}
```

Comando de entrada

- **scanf()**

```c
int main(){
    int x,z;
    float y;
    //Leitura de um valor inteiro
    scanf("%d",&x);
    //Leitura de um valor real
    scanf("%f",&y);
    //Leitura de um valor inteiro e outro real
    scanf("%d%f",&x,&y);
    //Leitura de dois valores inteiros
    scanf("%d%d",&x,&z);
    //Leitura de dois valores inteiros com espaço
    scanf("%d %d",&x,&z);

    return 0;
}
```

Comando de entrada

- **getchar()**
  - Comando que realiza a leitura de um único caractere

```c
int main(){
    char c;
    c = getchar();
    printf("Caractere: %c\n", c);
    printf("Codigo ASCII: %d\n", c);

    return 0;
}
```

Constantes

- Como uma variável, uma constante também armazena um valor na memória do computador.
- Entretanto, esse valor não pode ser alterado: é constante.
- Para constantes é obrigatória a atribuição do valor.

Constantes

- Usando **#define**
  - Você deverá incluir a diretiva de pré-processador **#define** antes de início do código:
  - **Cuidado: não colocar ";"**

```
#define PI 3.1415
```

- Usando **const**
  - Usando **const**, a declaração não precisa estar no início do código
  - A declaração é igual a de uma variável inicializada

```
const double pi = 3.1415;
```

Sequências de escape

- São constantes predefinidas
- Elas permitem o envio de caracteres de controle não gráficos para dispositivos de saída

Código	Comando
\a	som de alerta (bip)
\b	retrocesso (backspace)
\n	nova linha (new line)
\r	retorno de carro (carriage **return**)
\v	tabulação vertical
\t	tabulação horizontal
\'	apóstrofe
\"	aspa
\\	barra invertida (backslash)
\f	alimentação de folha (form feed)
\?	símbolo de interrogação
\0	caractere nulo (cancela a escrita do restante)

- Exemplo

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

int main(){
    printf("Hello World\n");
    printf("Hello\nWorld\n");
    printf("Hello \\ World\n");
    printf("\"Hello World\"\n");

    return 0;
}
```

- Saída

```
Hello World
Hello
World
Hello \ World
"Hello World"
```

Tipos Booleanos em C

- Um tipo booleano pode assumir dois valores:
  - verdadeiro ou falso (true ou false)
- Na linguagem C não existe o tipo de dado booleano. Para armazenar esse tipo de informação, use-se uma variável do tipo **int** (número inteiro)
  - Valor 0 significa falso
  - Números + ou – : verdadeiro
- Exemplos:

```
int AssentoOcupado = 1; // verdadeiro
int PortaAberta = 0; // falso
```

Referências

✓ Básica

- BACKES, André. *"Linguagem C: completa e descomplicada"*. Elsevier Brasil, 2013.
- **DAMAS, Luís. "Linguagem C". Grupo Gen-LTC, 2016.**
- **MIZRAHI, Victorine V. "Treinamento em linguagem C", 2a. ed., São Paulo, Pearson, 2008.**

✓ Extra

- BACKES, André. *"Programação Descomplicada Linguagem C"*. Projeto de extensão que disponibiliza vídeo-aulas de C e Estruturas de Dados. Disponível em: `https://www.youtube.com/user/progdescomplicada`. Acessado em: 25/04/2022.

✓ Baseado nos materiais do professor:

- Prof. André Backes (UFU)

Dúvidas?

Prof. Me. Claudiney R. Tinoco
profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)