# Computação 1

## Introdução a Linguagem C

**Prof. Luiz Fernando Carvalho**

luizfcarvalho@utfpr.edu.br

# Algoritmos Computacionais

Lógica de programação

Programador formula a solução

Linguagem de Programação


Aplicação/Programa

# Linguagem de Programação

- Uma linguagem de programação é um conjunto de ferramentas, regras de sintaxe e símbolos ou códigos que nos permitem escrever programas de computador;
- A primeira e mais primitiva linguagem de computador é a própria linguagem de máquina (0's e 1's);
- Antigamente, um programa era difícil de ser criado, longo e principalmente caro de construir;
- Era também difícil de ser entendi por outros programadores;
- Essa complexidade levou à necessidade de desenvolver novas técnicas e ferramentas;

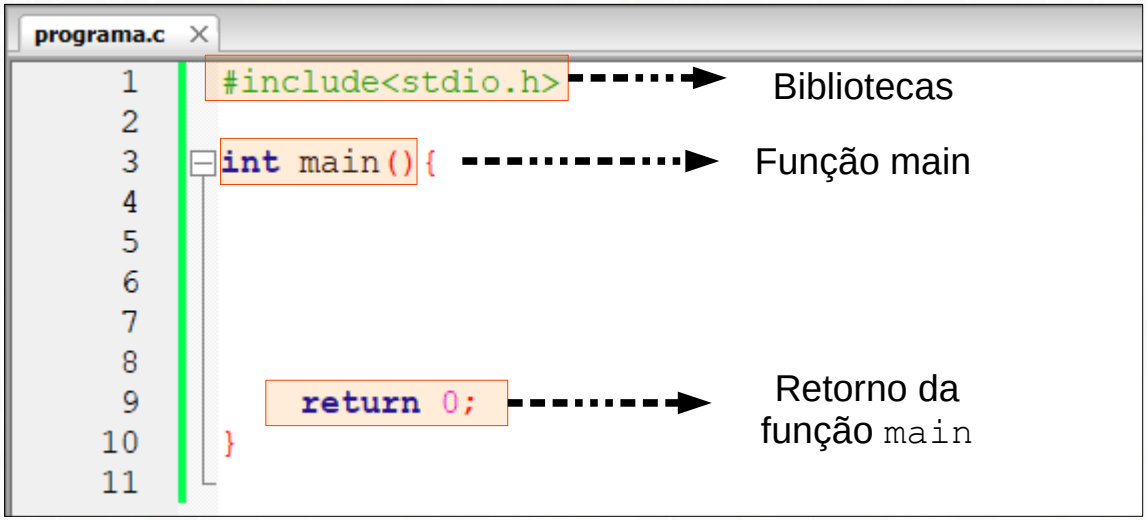
# Linguagem C

- A linguagem C nasceu na década de 70 (Dennis Ritchie), sendo padronizada pela ANSI (1989);
- A linguagem C foi desenvolvida a partir das linguagem B (criada por Ken Thompson) e BCPL (criada por Martin Richards);
  - Inseriu conceito de tipos de dados;
  - Independente de hardware;
- A linguagem C é uma linguagem genérica utilizada para a criação de programas diversos:
  - Planilhas e editores de textos, sistemas operacionais, programas de automação, banco de dados, automação industrial, dentre outros.

# Linguagem C - Características

- Comumente chamada de linguagem de “nível médio”
  - Combina elementos de alto nível com funcionalidade do *assembly* (linguagem de baixo nível);
  - Manipula bits, bytes e endereços;
- Linguagem portável
  - Possível adaptar o mesmo software a diferente tipos de computadores;
  - Obs.: Desde que siga os padrões (por exemplo, ANSI).

# Estrutura básica em C

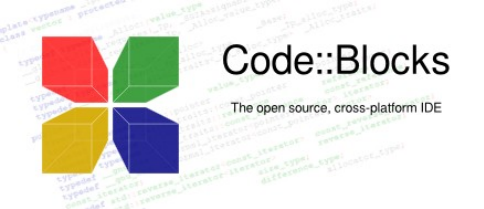



Bibliotecas contém conjunto de funções e instruções previamente estabelecidas e que podem ser usadas pelo programa

A execução do programa inicia-se pela função `main`

# IDE

- *Integrated Devolpment Environment* (IDE)
  - Interface gráfica;
  - Facilita a edição do código fonte;
  - Facilita a compilação e análise de código;
  - Facilita a depuração do código;
  - Possibilita o desenvolvimento de projeto de software maiores;






# Informações Gerais

- Uma variável sempre deve ser declarada antes de ser usada;
  - Declarar indica ao compilador para reservar um espaço na memória para guardar o valor dessa variável;

Exemplos de declaração

```
int main(){

  int num1;
  int num2;
  float resultado;
  float multiplicacao;
          .
          .
          .
```

=

```
int main(){

  int num1, num2;
  float resultado, multiplicacao;

                  .
                  .
                  .
```

# Entrada e saída de dados


Entrada:

```
scanf(formato, &variável);
```

Saída:

```
printf(conteúdo, [parâmetros]);
```

Exemplos:

```
scanf(“%f”, &n);
scanf(“%d”, &num);
scanf(“%d”, &cont);
```

Exemplos:

```
printf(“%f”, n);
printf(“Exame”);
printf(“O numero e’ %d”, num);
printf(“Sao necessarias %d moedas”, cont);
```

Ambas as funções `scanf` e `printf` pertencem a biblioteca `stdio.h`

# Saída de dados

Sintaxe geral do comando `printf`

```
printf("Mensagem escrita na tela", lista_argumentos);
```

Exemplos:

```
int num = 5;
char letra = 'C';
float pi = 3.1415;

printf("O valor de num e' %d", num);
printf("O valor de num e' %d e o valor de letra e' %c", num, letra);
printf("num = %d, letra = %c e pi = %f", num, letra, pi);
```

**%c → caractere (char)**
**%d ou %i → inteiro (int)**
**%f → ponto flutuante (float)**

# Saída de dados

- Códigos usados na função `printf()`

| Código | Formato |
|---|---|
| `%c` | Caractere |
| `%d ou %i` | Inteiro decimal com sinal |
| `%e` | Notação científica |
| `%f` | Ponto flutuante decimal |
| `%g` | Menor representação entre `%f` e `%e` |
| `%o` | Octal sem sinal |
| `%s` | *String* de caracteres |
| `%u` | Inteiro sem sinal (`unsigned int`) |
| `%x` | Hexadecimal sem sinal |
| `%%` | Símbolo % |

Lista completa em: http://www.cplusplus.com/reference/cstdio/printf/

# Caracteres de escape no `printf`

| Caractere | Significado |
|---|---|
| `\a` | Caractere (invisível) de aviso sonoro (bip) |
| `\n` | Caractere (invisível) de nova linha |
| `\t` | Caractere (invisível) de tabulação horizontal |
| `\\` | Caractere de barra invertida ‘\’ |
| `\’` | Caractere de aspas simples |
| `\’’` | Caractere de aspas duplas |
| `\?` | Sinal de interrogação |

Obs.: Não deve conter espaço entre \ e o símbolo desejado

# Saída com acentos (Windows)

```c
#include<stdio.h>
#include<locale.h> //para usar a função setlocale()


int main(){
    printf(“Apareça acentos: á, ê, õ ...\n”);
    //Altera a codificação de saída para o padrão do S.O., em Português…

    setlocale(LC_ALL, “ ”);
    printf(“Apareça acentos: á, ê, õ ...\n”);

    return 0;
}
```

Observação: A alteração no Windows vale também para número! O padrão para casas decimais é mostrar com ponto “.”, e com o `setlocalte()` mostrará com vírgula “,”

# Saída com acentos (Linux)

```c
#include<stdio.h>
#include<locale.h> //para usar a função setlocale()


int main(){
    printf("Apareça acentos: á, ê, õ ...\n");
    //Altera a codificação de saída para o padrão do S.O., em Português…

    setlocale(LC_ALL, " ");
    printf("Apareça acentos: á, ê, õ ...\n");

    return 0;
}
```

Não há necessidade de fazer ajustes se o Linux estiver instalado já com suporte à língua portuguesa. Porém, a recomendação é que use o `setlocale()`. A numeração com casas decimais também será separada por ","

# Entrada de dados

- A função `scanf()` permite que o usuário forneça dados pelo teclado para o programa

```
scanf("expressão de controle", lista de argumentos);
```

Indica ao compilador o TIPO de dado está se esperando que o usuário digite

- **"Expressão de controle"**: "%formato", parecido com o `printf`...
- **Argumento:** &*nome_variável*
  - Indica o endereço de memória onde se armazena o conteúdo de *nome_variável*

```
scanf("%i", &variavel_inteira);
scanf("%f", &variavel_float);
scanf("%c", &variavel_char);
```

# Entrada de dados

- Relembrando alguns aspectos sobre variáveis:
  - Possui um nome e tipo;
  - É associado a um endereço de memória;
  - Ao utilizar uma variável, acessa-se o seu conteúdo/valor localizado no endereço de memória associado a essa variável;
  - Ao atribuir um valor para uma variável, modifica-se o seu conteúdo/valor;
  - O & serve para obter o endereço de memória associado a uma variável;

```
int idade = 10;
printf("Endereco de memoria associado a variavel idade: %p \n",&idade);
printf("Valor armazenado por idade: %i \n", idade);
scanf("%i", &idade); //recebe um int e armazena no end. de memória associado à variável idade
printf("Endereco de memoria associado continua igual: %p \n",&idade);
printf("Novo valor armazenado por idade: %i \n", idade);
```

O que acontece se utilizar o `scanf()` sem & ?

# Exemplo de saída e entrada de dados

```c
#include<stdio.h>

int main(){
    int idade;

    printf(“Digite sua idade: ”);
    scanf(“%i”, &idade);

    printf(“Sua idade e’: %i anos”, idade);
    printf(“Daqui 5 anos você tera %i anos”, idade + 5);

    return 0;
}
```

# Comentários

- Em regra, são utilizados para a documentação do código;
- Objetiva facilitar o entendimento do programa por outros programadores ou o próprio criador;
- Comentários na linguagem C podem ser feitos de duas formas:

`/* <código> */` Para um determinado trecho de código (várias linhas)

`// <código>` Para uma linha de código, sendo colocado em seu início