Aula 01 - Introdução à Linguag C
Programação: Estrut. Sequencial

IC2

Prof: Anilton Joaquim da Silva

anilton.ufu@outlook.com

Primeiro Programa

- O Algoritmo em linguagem C, abaixo, descreve para o computador os passos necessários para se escrever a mensagem “Olá Mundo!” na tela do computador.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf ( "Hello world!\n “ );
    return 0;
}
```

Saída de dados padrão (tela)

- Um programa está executando a saída de dados quando envia para “fora” do programa tais dados. Exemplos comuns de saída de dados são a escrita em arquivo, o envio de mensagens na rede ou, impressão ou, mais comum, a exibição de dados na tela (stdout), do computador.
- Para enviar dados para a saída padrão do C, usamos a expressão **printf** , seguido do dado a ser impresso na tela.
- Forma geral: **printf("expressão de controle", argumentos);**
- **expressão** de controle : \n – nova linha, %c – char, %d – int, %f – float, e outros.
- **argumentos**: constantes, varáveis, expressões, e função.

Compilação e Execução

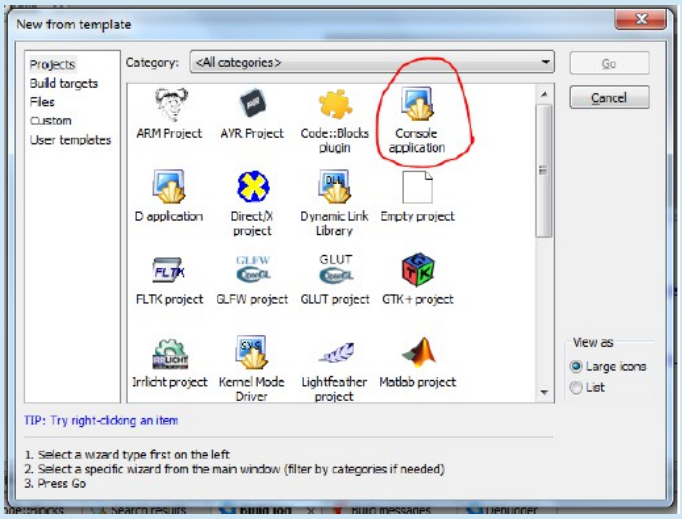
- Para colocarmos nossos algoritmos em execução, o primeiro passo é escrevê-los, usando um editor de textos qualquer que salve arquivos em texto puro, como o notepad, vim, gedit, etc. A este arquivo com o código chamaremos código fonte ou simplesmente fonte (extensão . c).
- A sequência de passos que compõem a compilação é a seguinte:
  - **Código Fonte → Pré-processador → Fonte Expandido → Compilador → Arquivo Objeto → Ligador → Executável**
- A compilação traduz o código que você escreveu para uma linguagem inteligível ao computador, salvando-o em um arquivo chamado arquivo objeto. Por exemplo, a compilação transformaria o código "Olá Mundo!" escrito acima em algo como:

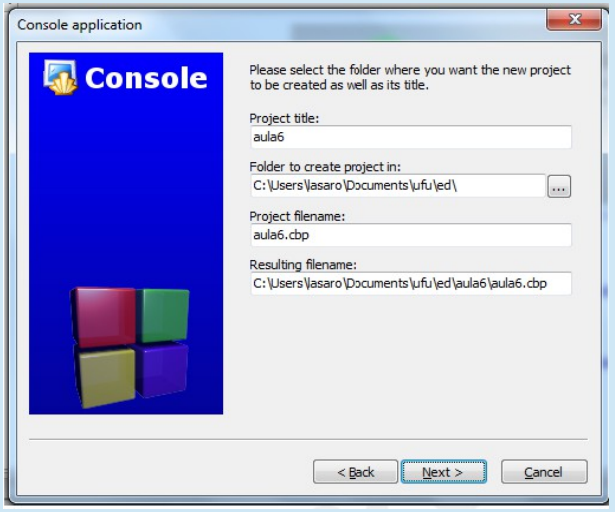
```
...
CALL   write(0x1,0x400623,0xe)
GIO    fd  1  "Olá Mundo!"
RET
```

OBS: para um primeiro programa:
primeiroProg.c
primeiroProg.obj
primeiroProg.exe

A IDE Code::Blocks

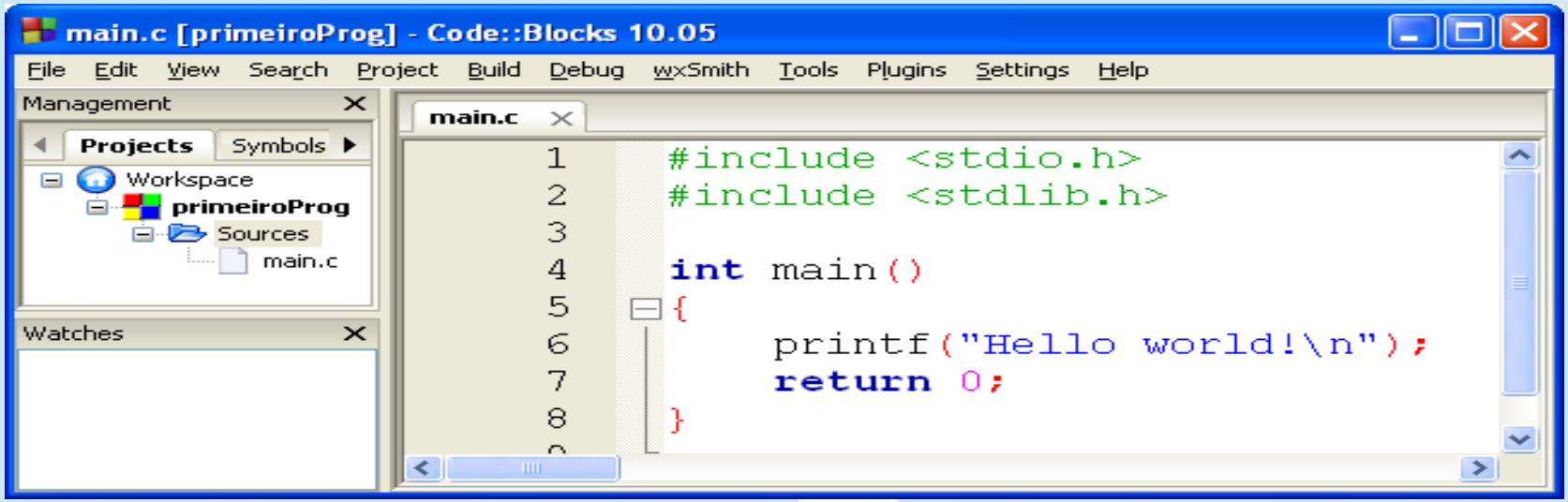
- Criando um Projeto: clique em **File** e, em seguida, **New**, **Project**;
- Escolha **Console Application** e então clique em **Go**;
- Escolha **C** e clique em **Next**;
- Em **Project title** escreva algo como **primeiroProg**; em **Folder to create the project in**, clique no botão com . . . e escolha uma pasta para salvar o projeto. Pode ser a pasta Meus Documentos ou uma pasta qualquer em um pen drive. Clique então **Next** e, na tela seguinte, clique em **Finish**.





A IDE Code::Blocks

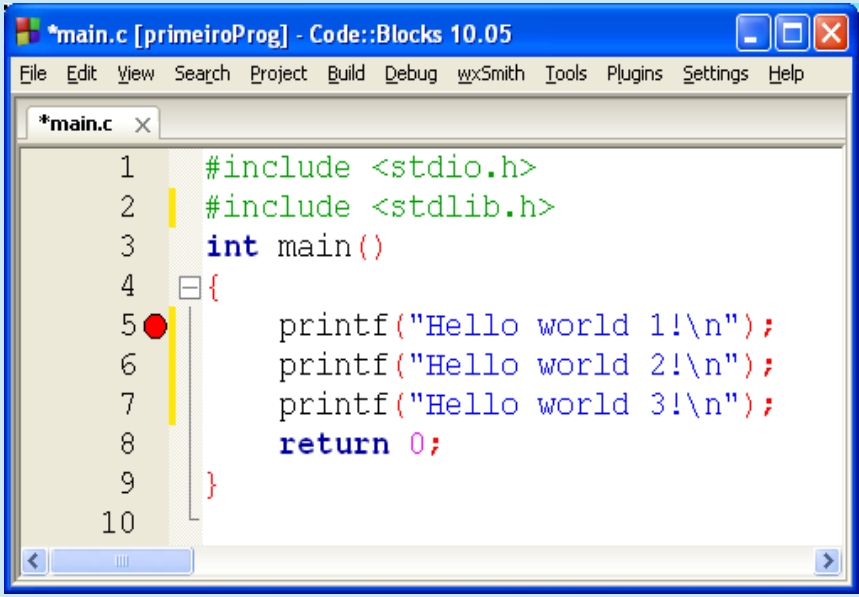
- Seu projeto foi criado. Agora abra o arquivo main.c, que está na pasta sources, dando um clique duplo no nome do arquivo. Observe que o Code::Blocks criou automaticamente um programa básico.



- Clique em em build and run. 
- Parabéns, você acaba de executar seu primeiro programa.

Depuração

- Todo programa é comum encontrar erros (bugs) de codificação e de lógica. Uma das formas de achar os bugs do seu programa é fazer com que o computador execute seu programa passo a passo, isto é, linha a linha, e acompanhar esta execução verificando se o programa faz o que você espera.



- Para depurar, clique ao lado direito do número 5 (quinta linha do programa), até que uma bolinha vermelha apareça, como na figura. A bolinha vermelha é, na verdade, um sinal de pare, e diz ao computador que deve, ao executar seu programa, parar ali.
- Clique no menu **Debug** e então em **Start** ou, alternativamente, pressione a tecla **F8** ( ). Observe que a execução parou onde você esperava.
- Agora, clique em **Debug** e **Next Line** ou aperte F7 ( ), no teclado, sucessivamente para ver o que acontece. Observe que cada linha é executada passo a passo.

Declaração de Variáveis

- Na linguagem C, toda variável (representar os dados) deve ser declarada (criada) no início do corpo da função que a contem. A declaração de uma variável tem pelo menos duas partes:
  - Tipo: tipo de dado, ou seja, se é um número, ou uma palavra, ou uma caractere, etc;
  - Nome: usado para referenciar a variável quando se precisa ler, usar em uma equação ou escrever a mesma;
- Algumas regras simples devem ser seguidas na hora de se nomear uma variável:
  - o nome só pode conter os caracteres [a-z], [A-Z], [0-9] e o "_";
  - o nome não pode começar com números.

Declaração de Variáveis

- Tipos básicos:
  - **int** - representando um número inteiro (16 bits – 2^15). ex. 3, 4, -5;
  - **float** - representando um número real, com casas decimais separadas por ponto " . " (32 bits – 3.4E^38). ex. 3.1416, 0.75, -1.2;
  - **char** - representando um caractere (1 byte): letra, dígito, ... identificado por apóstrofes. Exemplo '5', 'a', 'Z', '.', 'e', '-', '#'.
- Modificadores:
  - unsigned (unsigned int (16 bits: 0 – 2^16))
  - long (long int (32 bits: -2^31 a (2^31) – 1))
- Exemplo:

São exemplos de declarações de variáveis válidas:

```
1 int nota1, nota2;
  float media;
3 char _caractere;
```

São exemplos de declarações inválidas:

```
1 int 1nota, 2nota;
  float #media;
3 char nome completo;
```

Atribuição e uso de variáveis

```
int inteiro1, inteiro2;
float real;

inteiro1 = 0;
inteiro2 = 10;
real = 10.0;
```

```
int inteiro1 = 0,
    inteiro2 = 10;
float real = 10.0;
```

Parâmetros são variáveis:

```
float area_retangulo(float altura, float base)
{
    //Calcula e retorna a area do retangulo.
    return altura * base;
}
int main()
{
    float area;
    area = area_retangulo(2.0, 3.5);
    printf ( " \narea do retangulo: %f " ,  area);

    return 0;
}
```

Entrada de dados padrão (teclado)

- De forma semelhante ao printf, há um comando para leitura denominado scanf.
- Forma geral: **scanf("expressão de controle", argumentos);**

- **expressão** de controle: %c – char, %d – int, %f – float, ...
- **argumentos**: lista de variáveis (endereços: &variável)

- Este comando permite ler valores digitados (teclado = stdin) pelo usuário atribuindo as variáveis argumentos.

Entrada e Saida de dados

- Exemplos

- char letra;
- int idade;
- float altura;
- printf ("Informe a letra inicial de seu nome e sua idade, e altura: “);
- // a seguir eh feita a leitura
- scanf (“ %c %d %f ” , &letra, &idade, &altura);
- printf (“ \n A letra eh %c “ , letra);
- printf (" , sua idade eh %d, e sua altura eh %f \n “ , idade, altura);

Operadores

- Aritméticos:
  - = (atribuição) , + (soma), - (subtração), * (multiplicação), / (divisão) e % (resto da divisão de int)

OBS: a += b ➔ a = a + b; x *= y ➔ x = x * y; i++; ➔ i = i + 1; i--; ➔ i = i - 1;

- Relacionais:
  - == (teste de igualdade), != (diferente), > (maior que), < (menor que), >= (maior ou igual) e <= (menor ou igual)
- Lógicos:
  - **&& (and), || (or), ! (not)**
- Funções
  - abs(X): obtém o valor absoluto de X;
  - sqrt(X): calcula a raiz quadrada de X;
  - pow(X, Y): calcula potencia - X elevado a Y (X^Y)
  - log(X): calcula o logaritmo de X;
  - mod(X,Y): obtém o resto da divisão de X por Y;
  - trunca(X): obtém a parte inteira de X;
  - round(X): arredonda o valor de X;
  - sin(X): calcula o valor do seno de X;
  - cos(X): calcula o valor do cosseno de X;
  - tan(X): calcula o valor da tangente de X.

Lendo e imprimindo Strings

- Declarando uma string:

  char mome[30]; → vetor de caracteres

- Lendo uma string:

  scanf (" %s" , nome); → lê somente uma única palavra

  gets (nome); → lê uma frase com até 30 caracteres

- Imprimindo uma string:

  printf (" %s " , nome);

  ou

  puts (nome);

  → imprime nome, palavra ou frase