

Uma jornada de mil milhas começa sempre com um simples passo. Lao Tse.

PRIMEIROS PASSOS

Para poder construir programas em C é necessário utilizar um ambiente de desenvolvimento. Por ambiente de desenvolvimento entende-se um software que é composto de uma série de elementos, bem como de operações descritas na **Tabela 1**. Portanto este software pode transformar um conjunto de instruções digitadas e armazenadas num arquivo com extensão **.c** (ponto c), em um programa executável (arquivo com extensão **.exe**), conforme o esquema da **Figura 1**.

Tabela 1

Elemento	Operação
Editor	Ambiente onde os programas de computador podem ser escritos: arquivo.c .
Compilador	Traduz os comandos simbólicos de uma linguagem de alto nível para a linguagem de máquina: arquivo.o .
Ligador	Congrega o programa digitado com as funcionalidades disponíveis em bibliotecas em um único arquivo.exe (executável).

É importante observar que, uma vez construído um arquivo executável, as instruções nele contidas só poderão ser modificadas se todas as operações (edição, compilação e ligação) forem refeitas.

O ambiente a ser utilizado é BloodShed Dev-C++ que pode ser utilizado para desenvolver programa em

C ou em C++. O seu compilador usa a implementação Mingw do GCC (Gnu Compiler Collection). Uma de suas vantagens é que ele é “Free Software” (GNU General Public License), ou seja, ele pode ser distribuído e copiado à vontade. O endereço onde ele está disponibilizado é:

<http://www.bloodshed.net/dev/devcpp.html>

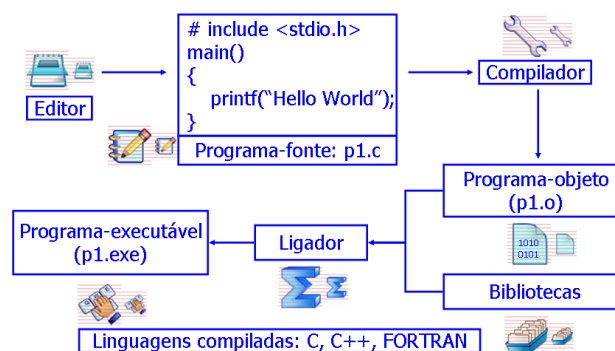


Figura 1: Operações para gerar um programa executável.

Antes de começar a trabalhar com o Dev-C++ é importante que você tenha acesso aos computadores do Pólo. Para tanto, você deve seguir os seguintes passos:

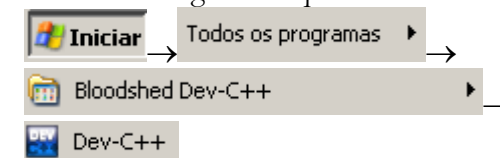
(1) Verificar se o computador está ligado. Caso não esteja, ligue o computador usando o botão da torre vertical. Verifique se o monitor está ligado também. Uma vez ligado o computador aperte as teclas Crtl+Alt+Del. Irá aparecer um quadro pedindo as seguintes informações:

User_name:

Password:

(2) Digitar um nome de usuário (user name) e uma senha (password) e depois digite a tecla ↵ (enter) para acessar a sua área de trabalho.

(3) Procurar e iniciar o programa Dev-C++, apertando sempre o botão esquerdo do mouse, utilizando a seguinte sequência:



Após realizar a sequência descrita acima, irá aparecer a tela descrita na Figura 2.

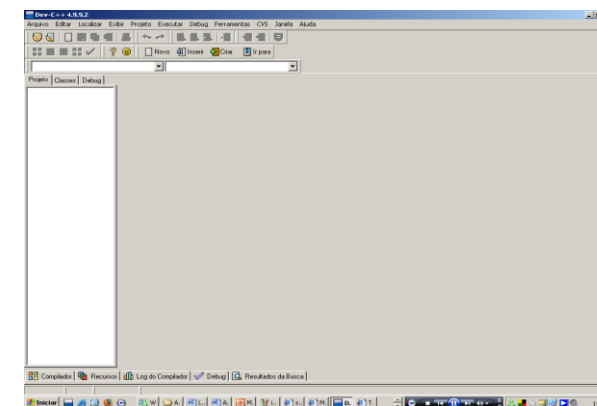
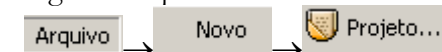


Figura 2: Iniciando o uso do Dev_C++.

Para se criar um programa no Dev_C++ aperte a seguinte sequência de botões:



Após esta sequência de comandos irá aparecer uma janela igual à Figura 3 a seguir. Nesta janela aparece

nome: projeto1, como sendo o nome do seu projeto. Este nome foi colocado automaticamente, mas se você quiser, pode mudá-lo para por exemplo, proj_Prog01. **A mudança de nome do projeto, se você for fazê-lo tem que ser feita aqui**, porque este nome será colocado na janela da figura 6.

A seguir, ainda nesta janela, selecione a opção “Console Application”, cuja imagem é fornecida na Figura 4.

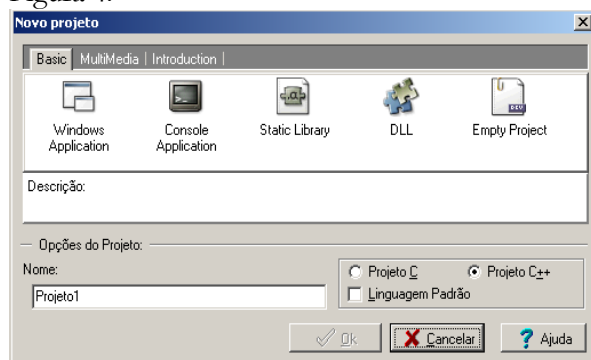


Figura 3: Opções de Projeto no Dev_C++.



Console
Application

Figura 4: Escolha de Projeto.

Neste ponto cabe esclarecer que no Windows existem dois tipos de programas de usuário:

(1) Aplicações para o console (Console Applications) que são executadas numa janela de texto (também conhecida como Prompt do MS-DOS).

(2) Aplicações gráficas (GUI Application ou Windows Application), que usam janelas, menus e outros elementos visuais como parte da interface com o usuário.

Enfatizando, se você for mudar o nome do projeto, isto tem que ser neste ponto, pois se você o fizer depois, poderá acarretar erros de compilação.

A seguir você deverá pressionar o botão de Ok, da Figura 5.



Figura 5: Botão de confirmação.

Então, irá aparecer uma janela igual a da Figura 6. Nesta janela, você deverá ir até a pasta **meu computador** e então ir até a pasta **da área Z**, que é a sua área no pólo. Na área Z, crie uma pasta nova, com o nome, por exemplo, de **LAB_PC** e salve o projeto que você está criando nesta pasta.

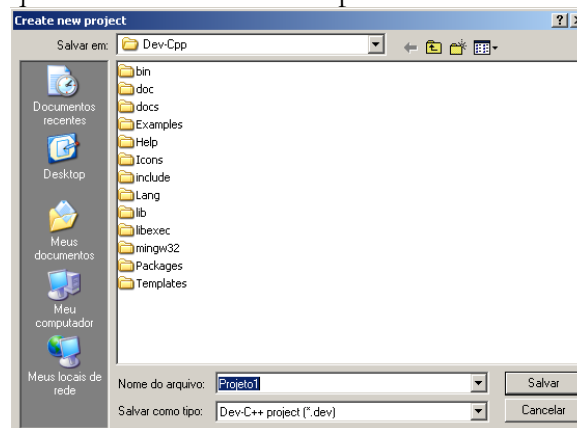


Figura 6: Janela para selecionar o local onde serão salvos os arquivos do seu programa.

Em hipótese alguma você deve gravar os arquivos em outro lugar que o diretório pertencente a sua área de trabalho.

Observe que após a operação de salvar o projeto, irá aparecer a tela da Figura 7.

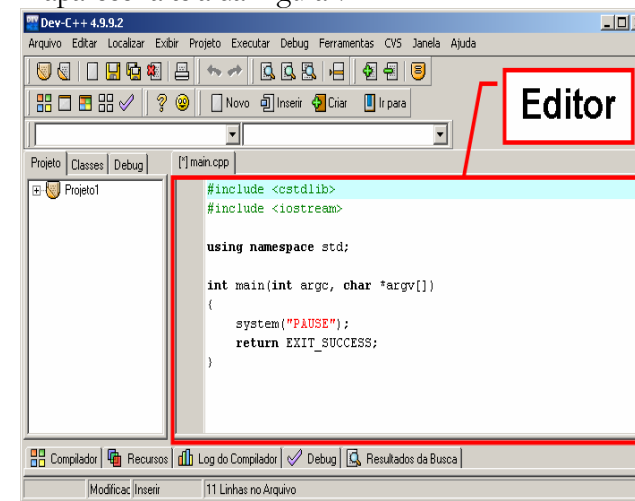


Figura 7: Tela do Dev_C++.

Observe que na Figura 7 existem alguns elementos cuja descrição das operações está sendo dada na Tabela 2, mostrada a seguir.

PRIMEIRO PROGRAMA EM C

Para testar o funcionamento do ambiente de programação apague o código inicialmente fornecido e então digite o programa descrito na Listagem 1, mas **NÃO DIGITE OS NÚMEROS NA LATERAL ESQUERDA.**

Tabela 2: Operações no Dev_C++.

Elemento	Operação
----------	----------

Editor	Permite a modificação do arquivo.c que contém as instruções do programa.
 Compilar	Realiza a compilação do programa digitado no editor de texto.
 Executar	Executa o programa previamente compilado.
 Compilar e Executar	Compila e executa.
 Recompilar	Recompila todos os arquivos.

Listagem 1: Meu primeiro programa em C.

```

1: # include <stdio.h>
2: #include <stdlib.h>
3: main()
4: {
5: printf("Hello World ");
6: system("pause");
7: }
```

Após digitar o programa da **Listagem 1**, clique nos botões: **arquivo, salvar como** e salve o que você digitou no mesmo diretório onde você salvou o projeto, mas para isso escolha um nome para o seu programa, por exemplo **prog01**.

OBS: Você deve escolher um nome diferente do nome que você deu ao projeto.

Uma vez salvo o seu programa, clique nos botões **Compile e Execute** (em inglês: compile and run). Feito isto, se não houver nenhum erro de digitação de sua

parte, vai aparecer na tela do computador o prompt do MS-DOS, descrito na Figura 8, a seguir.

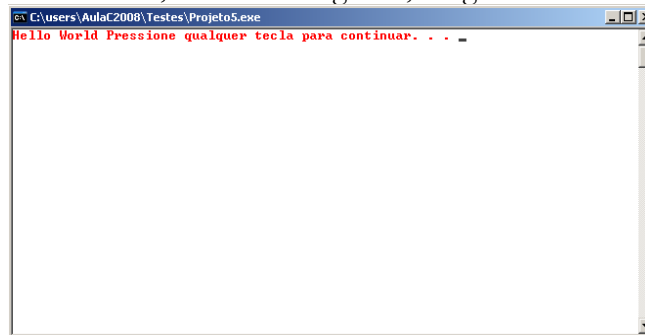


Figura 8: Tela de saída ao se executar o programa da Listagem 1.

Vamos agora entender os comandos presentes no programa em C da **Listagem 1**:

(a) Na linha 1 está o comando:

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

O símbolo **#** significa que o compilador deve executar uma ação. No caso, o comando a ser realizado pelo compilador é o de incluir (include) uma biblioteca, a **stdio.h**. Arquivos **.h** também são conhecidos como arquivos do tipo header (cabeçalho).

A **stdio.h** (standard input/output) é uma biblioteca que possui comandos relativos à entrada e saída de dados do computador. No caso do programa da **Listagem 1** é usado o comando **printf** que faz com que o programa envie uma mensagem ao prompt do MS-DOS.

(b) Na linha 2:

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

Esta linha realiza ação semelhante ao que foi comentado na linha 1, com a diferença que **stdlib.h** (standard library) é uma biblioteca que dispõe de comandos para o sistema (como é o caso do comando **system** na linha 7 do programa), alocação dinâmica de memória, geração de números pseudo-aleatórios, dentre outros.

(c) Na linha 3:

```
main()
```

Todo programa escrito em linguagem C deve ter uma função principal (**main()**). Além disso, é importante notar que a linguagem C diferencia letras maiúsculas de minúsculas. Deste modo, se for escrito **MAIN()** ou **Main()** o compilador irá fornecer uma mensagem de erro de sintaxe.

(d) Nas linhas 4 e 7:

```
{
}
```

Para delimitar as ações que serão realizadas na função principal (**main()**) é necessário utilizar os símbolos **{** e **}**. De forma geral, estes símbolos servem para delimitar blocos de instruções.

(e) Na Linha 5:

```
printf("Hello World");
```

Comando da biblioteca **<stdio.h>** que permite que o

usuário escreva uma mensagem na tela do prompt do MS-DOS. Para tanto, é necessário que a mensagem a ser escrita esteja entre aspas. Além disso, é importante notar que o comando **printf** é um comando da linguagem C e como tal deve ser sempre seguido por ponto-e-vírgula (;). Para melhor memorizar este comando, lembre-se que **print** em inglês significa imprimir.

(f) Linha 6:

```
system("pause");
```

Comando da biblioteca `<stdlib.h>` que permite que o usuário pare a execução do programa até que seja digitado a tecla ↵ (enter). Deste modo, esse comando permite, que seja visualizada a tela do prompt do MS-DOS mostrando o resultado da execução do programa. Agora experimente retirar o comando `system("pause")` do programa para ver o que acontece. Para fazer isso basta colocar a linha em comentário. Para se colocar uma frase qualquer em comentário, basta colocar `//` antes da frase, isto é:

```
// system("pause");
```

Observe que ao se fazer isto o comando irá mudar de cor, passando para a cor azul, indicando assim que ele é um comentário. Todo comentário é ignorado pelo compilador, isto é, ele não é executado. Para se verificar os efeitos desta modificação no código do programa é necessário apertar novamente os botões de Compilar e depois Executar.

Ao se compilar e executar novamente o programa, a tela do computador vai piscar rapidamente e você não

se conseguirá ver a mensagem escrita pelo comando `printf()`, pois tão logo a operação tenha sido realizada com sucesso o programa e o prompt do MS-DOS serão fechados.

O comando `//` que coloca uma frase do programa em comentário tem uma forma equivalente, que é a `/* */`. Os comentários são utilizados para se fazer a documentação do programa, como indicado na Listagem 2.

Como segundo exercício, digite e rode o programa da Listagem 2, abaixo. Para tanto **feche as janelas do seu primeiro programa, crie um novo projeto, do mesmo modo que você criou o primeiro, digite a listagem 2, compile e execute- a.**

Para fechar as janelas vá ao menu, clique em file/arquivo e a seguir clique em fechar projeto.

OBS: Cada vez que você for fazer um exercício, é necessário criar um novo projeto. Cada projeto só pode ter uma única função main(), logo cada projeto só pode ter um único programa.

A seguir digite o programa da Listagem 2.

Listagem 2: Meu segundo programa em C.

```
/* Meu segundo programa em C. */
# include <stdio.h>    // Biblioteca
# include <stdlib.h>   // Mais uma
main()                // Função Principal
{                      // Início dos comandos
    printf("  * ");
    printf(" *** ");
    printf(" *****");
    printf(" *****");
    system("pause"); // Para o programa
```

```
} // Fim dos comandos
```

PERGUNTAS BÁSICAS

PB1: Qual função deve estar presente em todos os programas em C?

PB2: Como devem terminar todas as instruções em C?

PB3: Como é delimitado um bloco de instruções em C?

PB4: Para que serve a linha `#include <stdio.h>` em um programa em C?

PB5: Qual o significado de `stdio`?

PB6: Identifique e corrija os erros de compilação dos programas a seguir:

```
// Programa 1
#include <stdio.h>
Main()
{
    printf("Hello World");
}
```

```
// Programa 2
#include <stdio.h>
main
{
    printf("Ola Mundo");
}
```

```

/* Programa 3 */
#include <stdio.h>
main()
{
    printf(Chega de Hello World);
}

```

PB7: Experimente digitar o comando, a seguir, na Listagem 2 : puts(“ O que acontece?”); Qual a diferença entre este comando e o comando **printf**.
Notar que **puts** também é um comando de **stdio.h**.

PB8: Faça uma mudança no programa **PB7** de modo que a figura na tela tenha o seguinte formato:

```

      *
     ***
    *****
   *
  *
 *
*

```

Dica: Você pode usar o comando **puts**.

Observe que tanto o comando **printf** como o **puts**, são comandos da linguagem C que são usados para se fazer o computador escrever na tela do MS-DOS. O **puts** escreve a frase que está entre aspas e muda de linha. O **printf**, escreve e não muda de linha.

O comando **puts** é usado para escrever somente caracteres, ou cadeia de caracteres. A palavra **put** em inglês significa COLOQUE e a letra **s** significa STRING. Então **puts** significa coloque a string (cadeia

de caracteres). O comando **printf** é mais geral e serve para escrever qualquer tipo de dado, tais como números inteiros, reais, letras, etc. Pode-se se fazer o **printf** mudar de linha. Para tanto basta que utilizemos o **\n** antes de se fechar às aspas.

PB9: Como exercício, digite e rode o programa da Listagem 3, abaixo. Não se esqueça de fechar as janelas do seu programa anterior, para criar um novo projeto, e digitar a listagem 3.

Listagem 3

```

/* Meu terceiro programa em C. */
# include <stdio.h>    // Biblioteca
#include <stdlib.h>    // Mais uma
main()                // Função Principal
{                      // Início dos comandos
    int valor1;
    float valor2;
    valor1 =3;
    valor2 =67.4;
    printf(“ primeiro numero= %d Segundo numero %f”,
    valor1, valor2);
    system(“pause”); // Para o programa
}                      // Fim dos comandos

```

PB10: Modifique o programa da listagem 3 colocando
valor1 =30;
valor2 =177.4;

Agora, no programa da listagem 3, coloque no **printf** um **\n** logo após o **%d** e outro **\n** logo após o **%f**. O que acontece?

Agora, substitua o comando **%f**, por **%.2f** e depois o **%.1f**. O que você observou?

PB11: Usando o comando **printf** faça um programa que imprima na tela o seguinte:

Nome: Luciana Gimenès da Silva
Endereço: Av Paulo Rebelo Junior 236 – Guaratinguetá
CPF: 01953815970
RG: 8515547-0

Observe que para fazer esse programa você vai ter que fechar o programa anterior, criar um novo projeto e digitar os comandos referentes as bibliotecas (**#include**), função principal e tudo mais.

PB13: Refaça o programa **PB12** usando o comando **puts**.

PB14: Usando o comando **printf** faça um programa que imprima a seguinte figura na tela:

```

      A
     AAA
    A  A
   AAAAA
  A      A
 A        A

```