Antônio Henrique Reis

Programando em Linguagem C e C++

Com o Microsoft Visual Studio

editora
VIENA
1ª Edição
Bauru/SP
Editora Viena
2015

Sumário

Lista de Siglas e Abreviaturas **15**

1. Introdução **17**
1.1. O Primeiro Programa em C/C++ 20
1.2. Ambiente de Desenvolvimento do Microsoft Visual Studio 20
1.3. Editando o Programa 21
1.4. Verificando o Código Fonte C Utilizando o VS 22
1.5. Apresentando uma String na Tela 23
1.6. Compilando o Programa 23
1.7. Executando o Programa 24
1.8. Verificando o Código Fonte C++ Utilizando o VS 26

2. Construindo Programas C/C++ **31**
2.1. Indentação do Código Fonte 33
2.2. Case Sensitive 33
2.3. Comentários 34
2.4. Códigos de Escrita na Tela 35

3. Variáveis **41**
3.1. Variáveis na Linguagem C/C++ 43
3.1.1. Exemplo Usando Variáveis em Linguagem C 43
3.1.2. Exemplo Usando Variáveis em Linguagem C++ 44
3.2. Regras de Nomes das Variáveis 45
3.3. Declarações de Variáveis 46
3.4. Tipos de Dados das Variáveis 46
3.5. Inicializando Variáveis 47
3.6. Constantes 47

4. Operadores **53**
4.1. Operador de Atribuição 55
4.2. Operadores Aritméticos 55
4.2.1. Exemplo Usando Operadores Aritméticos em Linguagem C 56
4.2.2. Primeiro Exemplo Usando Operadores Aritméticos em Linguagem C++ 57
4.2.3. Segundo Exemplo Usando Operadores Aritméticos em Linguagem C++ 58
4.3. Operadores Aritméticos de Atribuição 59
4.4. Operadores de Incremento e de Decremento 60
4.4.1. Exemplo de Incremento 60
4.4.2. Exemplo de Decremento 61
4.5. Operadores de Incremento ou Decremento Pré-fixado ou Pós-fixado 61

5. Funções de Entrada e Saída de Dados **69**
5.1. Entrada, Processamento e Saída de Dados 71
5.2. Entrada e Saída de Dados em Linguagem C 72
5.2.1. Primeiro Exemplo de Entrada e Saída de Dados em Linguagem C 72
5.2.2. Segundo Exemplo de Entrada e Saída de Dados em Linguagem C 74

5.3. Entrada e Saída de Dados em Linguagem C++ ... 75
5.3.1. Primeiro Exemplo de Entrada e Saída de Dados em Linguagem C++ ... 75
5.3.2. Segundo Exemplo de Entrada e Saída de Dados em Linguagem C++ ... 77
5.4. Lendo Dados do Tipo: String ... 80
5.4.1. Exemplo de Entrada e Saída de Dados do Tipo String em Linguagem C ... 80
5.4.2. Exemplo de Entrada e Saída de Dados do Tipo String em Linguagem C++ ... 81

6. Estruturas de Decisões ... 85
6.1. Operadores Relacionais ... 87
6.2. Desvio Condicional Simples ... 89
6.2.1. Primeiro Exemplo de Desvio Condicional Simples em Linguagem C ... 90
6.2.2. Segundo Exemplo de Desvio Condicional Simples em Linguagem C++ ... 91
6.2.3. Terceiro Exemplo de Desvio Condicional Simples em Linguagem C ... 91
6.3. Desvio Condicional Composto ... 92
6.3.1. Primeiro Exemplo de Desvio Condicional Composto em Linguagem C ... 93
6.3.2. Segundo Exemplo de Desvio Condicional Composto em Linguagem C ... 94
6.4. Desvio Condicional Encadeado ... 95
6.4.1. Primeiro Exemplo de Desvio Condicional Encadeado ... 95
6.4.2. Segundo Exemplo de Desvio Condicional Encadeado ... 97
6.5. Desvio Condicional de Múltiplos Casos ... 99
6.6. Desvio Condicional Composto com o Operador Ternário ... 101

7. Operadores Lógicos ... 109
7.1. Operador Lógico .E. ... 111
7.1.1. Primeiro Exemplo do Operador Lógico .E. ... 111
7.1.2. Segundo Exemplo do Operador Lógico .E. ... 112
7.2. Operador Lógico .OU. ... 114
7.3. Operador Lógico .NÃO. ... 115

8. Estruturas de Repetição ... 123
8.1. Laços de Repetição com o Teste Lógico no Início Usando While ... 126
8.1.1. Exemplo de Laço de Repetição com o Teste Lógico no Início Usando While ... 128
8.2. Laços de Repetição com uma Quantidade Indeterminada de Repetições Usando While ... 130
8.2.1. Primeiro Exemplo de Laço de Repetição com uma Quantidade Indeterminada de Repetições Usando While ... 131
8.2.2. Segundo Exemplo de Laço de Repetição com uma Quantidade Indeterminada de Repetições Usando While ... 133
8.3. Laços de Repetição com uma Quantidade Determinada de Repetições Usando For ... 135
8.3.1. Primeiro Exemplo de Laço de Repetição com uma Quantidade Determinada de Repetições Usando a Estrutura For ... 135
8.3.2. Segundo Exemplo de Laço de Repetição com uma Quantidade Determinada de Repetições Usando a Estrutura For ... 136
8.3.3. Terceiro Exemplo de Laço de Repetição com uma Quantidade Determinada de Repetições Usando a Estrutura For ... 137
8.4. Laços de Repetição com o Teste Lógico no Final do Laço Usando Do-while ... 138

8.4.1. Primeiro Exemplo de Laço de Repetição com uma Quantidade Indeterminada de Repetições Usando Do-while 139
8.4.2. Segundo Exemplo de Laço de Repetição com uma Quantidade Indeterminada de Repetições Usando Do-while 140

9. Estruturas de Dados 151
9.1. Estruturas de Dados do Tipo: Vetor – Tabelas em Memória 155
9.1.1. Primeiro Exemplo de Estruturas de Dados do Tipo: Vetor 155
9.1.2. Segundo Exemplo de Estruturas de Dados do Tipo: Vetor 158
9.2. Estruturas de Dados do Tipo: Matriz 160
9.2.1. Primeiro Exemplo de Estruturas de Dados do Tipo: Matriz 161
9.2.2. Segundo Exemplo de Estruturas de Dados do Tipo: Matriz 166

10. Programação Estruturada 173
10.1. Procedimentos e Funções 175
10.2. Introdução as Funções 175
10.3. Protótipo de Funções 179
10.4. Declaração das Funções 181
10.5. Funções Que Não Recebem e Nem Retornam Valores 182
10.6. Funções Simples 182
10.7. Variáveis Locais 184
10.8. Variáveis Globais 185
10.9. Classes de Armazenamento 187
10.9.1. Classe de Armazenamento - Auto 187
10.9.2. Classe de Armazenamento - Extern 187
10.9.3. Classe de Armazenamento - Static 189
10.9.4. Classe de Armazenamento - Register 190
10.10. Refinamento Sucessivo 191
10.11. Funções que Retornam um Valor 194
10.12. Funções com Chamada por Valor 196
10.13. Funções com Chamada por Referência 198
10.14. Funções Recursivas 201

11. Trabalhando com Ponteiros em Linguagem C/C++ 213
11.1. Ponteiro, O Que é Isto? 215
11.2. Declarando Uma Variável do Tipo Ponteiro 217
11.3. Usando uma Variável do Tipo Ponteiro 217
11.4. Outro Exemplo do Uso de Uma Variável do Tipo Ponteiro 219

12. Bibliotecas de Funções da Linguagem C/C++ 227
12.1. Arquivos de Cabeçalho 229
12.2. As Funções e As Suas Bibliotecas 230
12.3. A Função Strlen( ) 231
12.4. A Função Strcat_s( ) 233
12.5. A Função Strcmp( ) 235
12.6. Funções atoi( ) e atof( ) 237
12.7. A função pow( ) 239
12.8. A Função Rand( ) 240

12.9. A Função Sqrt( ) 241

13. Representação de Dados - Struct 245

13.1. Um Estrutura Simples 247

13.1.1. Um Exemplo Prático do Uso de Uma Estrutura 249

14. O Pré-Processador 257

14.1. A Diretiva #define 259

14.2. Macros 261

14.3. A Diretiva #Include 262

15. Entrada e Saída de Dados com Arquivos em Disco 269

15.1. Criando um Arquivo 272

15.2. Lista de Modos da Classe Ios 274

15.3. Lendo Dados Gravados 277

Referências 283

Glossário 287

Lista de Siglas e Abreviaturas

ABNT	*Associação Brasileira de Normas Técnicas.*
ANSI	*American National Standards Institute.*
ASCII	*American Standard Code for Information Interchange.*
BCPL	*Basic Combined Programming Language.*
IDE	*Integrated Development Environment.*
I/O	*Input / Output.*
OOP	*Object-Oriented Programming - Programação Orientada a Objetos.*
VS	*Visual Studio Express 2013 for Desktop.*

1 Introdução

1.1. O Primeiro Programa em C/C++
1.2. Ambiente de Desenvolvimento do Microsoft Visual Studio
1.3. Editando o Programa
1.4. Verificando o Código Fonte C Utilizando o VS
1.5. Apresentando uma String na Tela
1.6. Compilando o Programa
1.7. Executando o Programa
1.8. Verificando o Código Fonte C++ Utilizando o VS

1. Introdução

Nos laboratórios da empresa americana: "**Bell Labs**", nos anos 70, **Dennis M. Ritchie** e **Ken Thompson** criaram a **linguagem C**. Com origem na linguagem B de Thompson, que foi uma evolução da linguagem BCPL. B era a primeira letra de BCPL e C a segunda, portanto os autores acharam que seria lógico chamar a linguagem de C.

A estrutura modular da **linguagem C** resulta em programas mais legíveis e documentados. Quando as técnicas de OOP (Object-Oriented Programming - Programação Orientada a Objetos) são utilizadas, se faz o uso da **linguagem C++**.

A **linguagem C** é utilizada no desenvolvimento de sistemas operacionais e em diversos tipos de aplicações. Tem influência direta nas linguagens C++, Java, C#, PHP e JavaScript.

Esta linguagem é reconhecida entre os programadores como uma linguagem de "alto/baixo nível", ou seja, é possível trabalhar como uma linguagem próxima da linguagem humana, fazendo operações do tipo: "a=b+c;" e também com operações de baixo nível, como, por exemplo: ligando ou desligando portas no hardware do computador.

***Nota**: Linguagem de baixo nível é a linguagem de programação que utiliza instruções do processador. Portanto, estão diretamente relacionadas a arquitetura do computador. Um exemplo desta linguagem é o Assembly. A linguagem de alto nível distancia-se da arquitetura do computador e fica mais próxima da linguagem humana. Não há necessidade de se conhecer as características do processador.*

Como a utilização da **linguagem C** tornou-se muito popular foi criado um padrão para a linguagem, chamado de: "**Padrão ANSI**". É chamado desta maneira porque "**ANSI**" é uma sigla que representa: "**American National Standards Institute**", algo parecido com a sigla: "**ABNT**", que no Brasil significa: "**Associação Brasileira de Normas Técnicas**".

Antes do "**Padrão ANSI**" a única referência para a **linguagem C** era o livro: "**Linguagem de Programação C"**, de **Brian Kernighan** e **Dennis Ritchie**. Este livro causou algumas divergências entre compiladores da **linguagem C**, então, o comitê ANSI teve que intervir para criar um padrão a ser utilizado por todos eles.

A **linguagem C++** é uma sequência natural da linguagem C.

A **linguagem C** utiliza a técnica da programação estruturada, mas C++ agrega a técnica da programação orientada a objetos.

No início dos anos 80, **Bjarne Stroustrup**, da "**Bell Labs**", foi quem desenvolveu a **linguagem C++**. Esta linguagem teve como objetivo fazer algumas simulações orientadas a eventos, mas a principal meta foi manter a compatibilidade com a **linguagem C**, para manterem protegidas as milhares de linhas de códigos fonte escritos em linguagem C. Esta é a razão principal de poder utilizar os compiladores atuais para **C++** e utilizar a estrutura da **linguagem C** e tudo funcionar corretamente.

As **linguagens C e C++** são uma das melhores alternativas para entrar no mundo da programação de computadores.

Seja bem-vindo e agora o trabalho será iniciado com programas simples para você ir se acostumando com a sintaxe das **linguagens C/C++**.

1.1. O Primeiro Programa em C/C++

Os estudos serão iniciados com um programa que escreve na tela do computador a frase: "**Meu primeiro programa**". Veja, abaixo, como é o código fonte do programa:

```
main( )
{
   printf("Meu primeiro programa");
}
```

Um programa em C consiste de uma ou mais funções que especificam o que deve ser feito. No exemplo acima **main( )** é uma função. Qualquer programa em C começa com a função **main( )** que marca o ponto inicial da execução do programa.

Todas as instruções devem estar dentro de um bloco de comandos envoltos em uma chave que abre "{" e uma chave que fecha "}".

A linha: **printf ("Meu primeiro programa")**; é uma chamada a função **printf**, com o argumento "**Meu primeiro programa**". Esta sequência de caracteres entre aspas é chamada de: **cadeia de caracteres**, ou também **string**. A palavra **printf** é uma função de biblioteca que escreve o argumento na tela do seu computador. Os parênteses devem estar presentes mesmo quando não há argumentos.

As instruções C são sempre encerradas por um **ponto-e-vírgula (;)**.

Em **linguagem C++** o mesmo programa poderia ter sido escrito da seguinte forma:

```
main( )
{
   cout << "Meu primeiro programa";
}
```

A única diferença é que foi utilizada a forma de escrever na tela usando o fluxo de saída: "cout" que escreve na tela a frase que está entre aspas: "Meu primeiro programa".

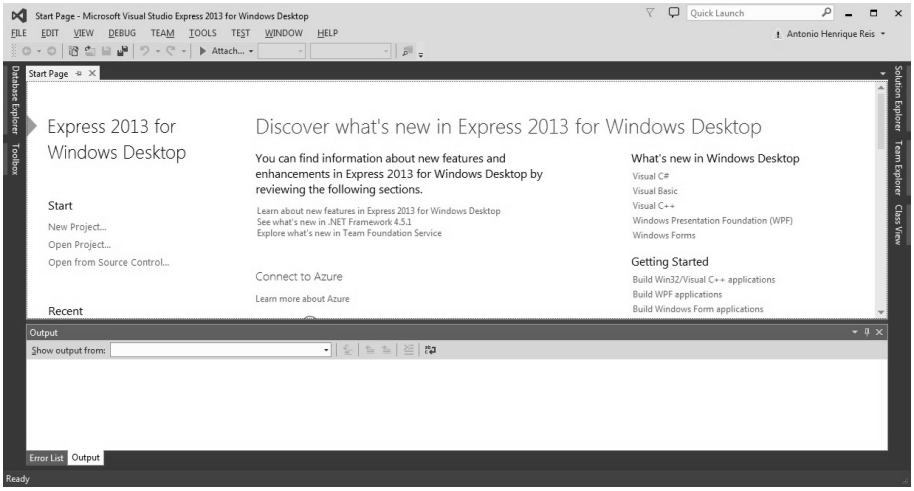
1.2. Ambiente de Desenvolvimento do Microsoft Visual Studio

Para escrever os programas deste livro, foi escolhido o ambiente de desenvolvimento da **Microsoft®**, o **Visual Studio C++ Express 2013 for Desktop**, que pode ser acessado e instalado direto do site da **Microsoft**. Você deve verificar a forma de utilização do mesmo para seus estudos.

Este sistema tem uma interface **IDE** (Integrated Development Environment), ou seja: um ambiente integrado de desenvolvimento de software, bastante amigável na qual é possível escrever o código fonte, compilar, executar e testar o programa no mesmo local.

Para acessar o programa depois de instalado, conforme instruções do site da **Microsoft**, proceda da seguinte forma:

1. Clique no botão **Iniciar** ( )do **Windows**;
2. Clique em **Todos os Programas**, localize e clique na pasta: **Visual Studio 2013**, e em seguida, em: **VS Express 2013 for Desktop**;
3. Aguarde a inicialização do programa.



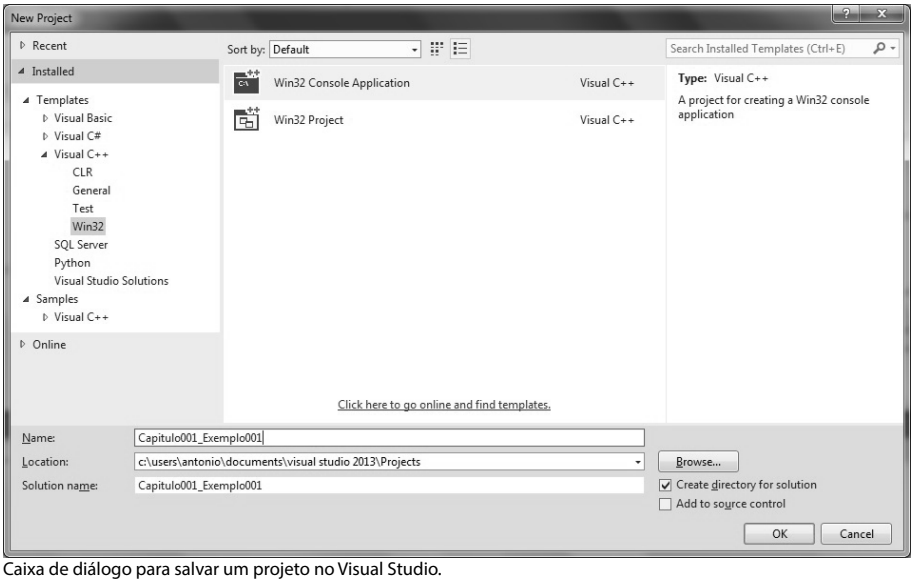
Tela inicial do Microsoft Visual Studio.

Esta é a tela inicial do **Visual Studio**, onde encontra-se o acesso aos menus e a interface de desenvolvimento de software. No ambiente profissional do **Microsoft Visual Studio** é possível desenvolver programas em várias linguagens como C, C++, C# (pronuncia-se C Sharp), Asp.net e também o Visual Basic.

1.3. Editando o Programa

Inicialmente é necessário preparar o ambiente de trabalho do **Visual Studio** para que seja possível desenvolver programas utilizando as **linguagem C/C++**. Para preparar o ambiente de trabalho, proceda da seguinte forma:

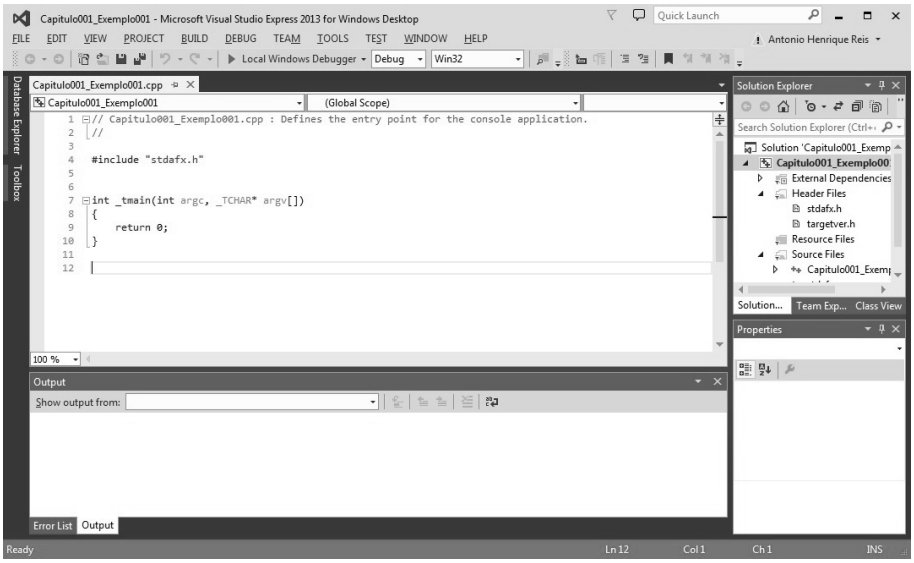
1. Na tela inicial do **Visual Studio**, acesse o menu: **FILE** e, em seguida, clique na opção **New Project (Ctrl+Shift+N)**;
2. A caixa de diálogo **New Project** será exibida:



Caixa de diálogo para salvar um projeto no Visual Studio.

3. Escolha o tipo de projeto que será desenvolvido;
4. No lado esquerdo da caixa de díalogo, escolha a opção **Visual C++** e, em seguida, clique na opção **Win32**;
5. Ao centro da janela, clique em **Win32 Console Application**, que é o tipo de programa que será desenvolvido;
6. Defina um nome para o projeto, na caixa de textos **Name**;
7. Verifique se na caixa de textos **Location** está definido o caminho da pasta padrão do projeto, conforme a imagem anterior;
8. Clique no botão **OK** e na próxima tela, clique no botão **Finish**.

Após realizar estes passos o ambiente de trabalho deverá ficar semelhante à imagem a seguir:



Janela de codificação do Visual Studio.

Note que agora o **Visual Studio** já está pronto para trabalhar com o código fonte do projeto do primeiro programa que será desenvolvido.

A partir deste momento será utilizado apenas o código fonte para referência da **linguagem C/C++**. A partir de agora, o ambiente do **VS Express 2013 for Desktop** será referido apenas como **VS**.

1.4. Verificando o Código Fonte C Utilizando o VS

Observe com maiores detalhes o código apresentado na imagem anterior:

```
// Capitulo001_Exemplo001.cpp : Defines the entry point for the
console application.
//

#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
   return 0;
}
```

O **VS** traz várias linhas já preenchidas, basta digitar os comandos dentro das chaves da função principal **main**, que no **VS** está identificada como **_tmain**.

As linhas que começam pelos caracteres das barras "//", são comentários de final de linha e o compilador não irá conferir estas linhas.

A linha escrita: **#include "stdafx.h"** irá incluir um cabeçalho com informações da biblioteca padrão para os programas em C. É uma diretiva do pré-processamento.

Arquivos com sobrenome: "**.h**", são identificados como "**headers**", ou seja, uma inclusão de um cabeçalho com informações para funções que serão utilizadas no programa.

Note que a esquerda da "**_tmain**" o termo "**int**", informa que haverá um retorno de um valor inteiro, o que é possível verificar pela linha escrita: "**return 0;**".

Dentro dos parênteses na **_tmain**, há uma lista de parâmetros que poderão ser recebidos pela função (**int argc, _TCHAR* argv**[]).

A partir deste momento os comandos serão escritos dentro das chaves da função " **_tmain**", se você estiver utilizando outra interface **IDE**, basta fazer os ajustes necessários para o seu ambiente de desenvolvimento.

O padrão adotado para o desenvolvimento de todo o projeto, é o **ANSI (**American National Standards Institute), portanto independente da **IDE** utilizada os comandos que foram inseridos dentro das chaves, irão funcionar em todas as **IDE's**.

1.5. Apresentando uma String na Tela

Será feita agora a inclusão do comando que vai apresentar na tela a string com a frase "**Meu primeiro programa**".

```
// Capitulo001
//

#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
   printf("Meu primeiro programa");
   return 0;
}
```

O comando que vai fazer esta tarefa é a chamada da função **printf** escrito na linha:

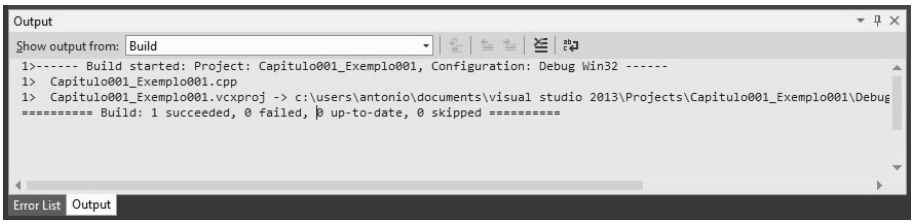
```
   printf("Meu primeiro programa");
```

1.6. Compilando o Programa

Para compilar o programa desenvolvido, proceda da seguinte maneira:

1. Na **Barra de Menus**, acesse o menu **BUILD** e, em seguida, clique em **Build Solution (F7)**;
2. O compilador irá verificar se a sintaxe do código fonte está correta.

É possível verificar se não há erros no código, por meio da janela **Output**, localizada na parte inferior da tela, conforme a imagem a seguir:

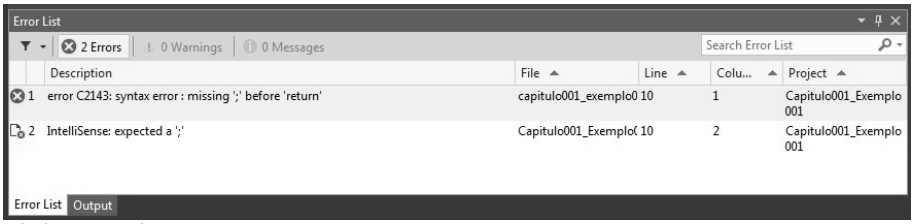


Tela de OutPut do VS.

Repare na última linha a frase: "**Build: 1 succeeded, 0 failed, ...**" onde fica possível identificar que o código foi compilado com sucesso sem nenhuma falha.

Se houver algum erro de sintaxe a janela: **Error List** irá aparecer. Nesse caso, basta dar dois cliques no primeiro erro encontrado, interpretar o que diz na janela sobre este erro e corrigí-lo.

Veja na figura a seguir a janela de erros:



Tela de Error List do VS.

Deve-se sempre corrigir o primeiro erro e depois compilar novamente o programa, pois um erro na **linguagem C/C++** poderá levar a ocorrência de muitos erros.

Nesse caso, foi retirado propositalmente o ponto-e-vírgula (";") de uma das linhas, forçando o aparecimento de erros.

1.7. Executando o Programa

Depois de corrigir todos os erros de sintaxe, o programa deve ser executado. Basta acessar o menu **DEBUG**, e clicar no comando **Start Debugging (F5)**.

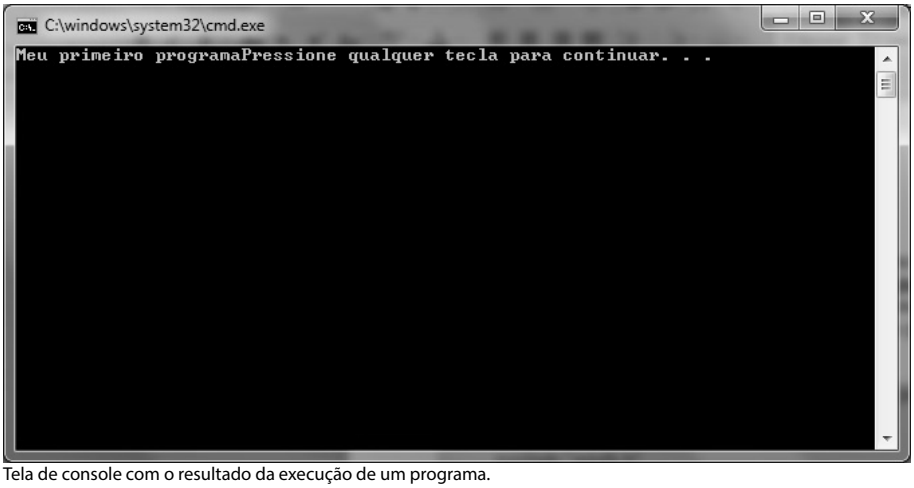
Você vai perceber que o programa irá mostrar uma janela com a mensagem e logo desaparecerá. Para fazer uma pausa na tela até que uma tecla seja pressionada, deve-se colocar uma linha de comando para esta tarefa, conforme o exemplo a seguir:

```
// Capitulo001
//

#include "stdafx.h"
#include "stdlib.h"

int _tmain(int argc, _TCHAR* argv[])
{
   printf("Meu primeiro programa");

   system("PAUSE");
   return 0;
}
```

Foi inserida a diretiva: **#include "stdlib.h"**, que instrui o pré-processamento para incluir um arquivo no programa com as definições para o uso do comando: "**system("PAUSE");**" que irá fazer uma pausa na tela e esperar uma tecla ser pressionada.

Depois de compilado, o programa foi novamente executado por meio do comando **Start Debugging (F5)**, localizado no menu **DEBUG**.

Veja o resultado, após a execução do comando:

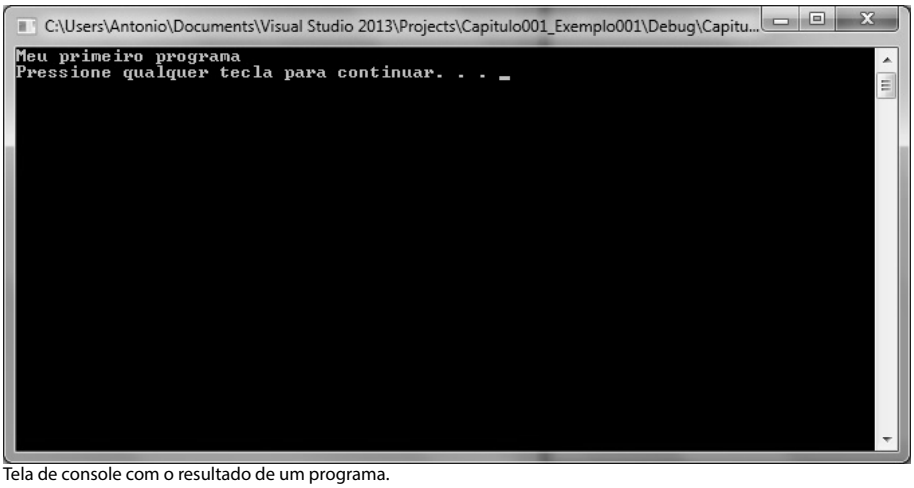


Tela de console com o resultado da execução de um programa.

Repare que a mensagem "**Pressione qualquer tecla para continuar. . .**" foi apresentada como resposta do comando **system("PAUSE")**, mas ela apareceu junto da mensagem "**Meu primeiro programa**". Para que a mesma fosse apresentada na linha de baixo seria necessario adicionar ao código, um comando para mudar de linha, que é o "**\n**". Veja o exemplo no código fonte a seguir:

```
// Capitulo001
//
#include "stdafx.h"
#include "stdlib.h"

int _tmain(int argc, _TCHAR* argv[])
{
   printf("Meu primeiro programa \n");

   system("PAUSE");
   return 0;
}
```

Veja a seguir o resultado na janela de console:



Tela de console com o resultado de um programa.

Observe agora que o programa escreve a mensagem na tela "**Meu primeiro programa**" e realiza uma quebra de linha quando encontra o código "**\n**".

Veja abaixo outros códigos que poderiam ser utilizados:

Códigos	Significado
\n	Nova linha.
\t	Tabulação.
\b	Retrocesso.
\a	Alarme (sinal sonoro).
\\	Para a própria barra.
\0	Zero (null).
\'	Apóstrofo.
\"	Para aspas duplas.

1.8. Verificando o Código Fonte C++ Utilizando o VS

Para obter o mesmo resultado utilizando a **linguagem C++** o código fonte deve ser escrito da seguinte forma:

```
// Capitulo001
//

#include "stdafx.h"
#include <iostream>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
   cout << "Meu primeiro programa \n";
```

```
    system("PAUSE");
    return 0;
}
```

A diretiva **#include "stdafx.h"** instrui ao pré-processamento para buscar as definições dentro do arquivo de cabeçalho. Como foram usadas as aspas duplas, o pré-processamento vai procurar primeiro na pasta atual e depois na pasta **INCLUDE**.

Na diretiva **#include <iostream>** o arquivo será procurado na pasta **INCLUDE**. Este arquivo possui as definições padrões de leitura e gravação (**I/O**) para C/C++.

A linha: **using namespace std**; agrega um identificador e todos os nomes que forem declarados no interior da mesma.

O resultado deste programa é o mesmo da **linguagem C**.

Exercícios Práticos

1. O que deve ser feito, quando o programa estiver sendo executado, para que ele faça uma pausa e fique esperando uma tecla para encerrar o programa? Explique o proce-dimento de forma completa.

2. Depois de escrito o código fonte do programa, o que deve ser feito para testar o programa? Explique o procedimento passo-a-passo.

***Dica**: Para responder as questões 1 e 2 consulte os tópicos: **1.6. Compilando o programa** e também o **1.7. Executando o programa** .*

3. Analise o programa abaixo e responda as perguntas a seguir:

```
// Exercício do Capitulo 01

#include "stdafx.h"
#include "stdlib.h"

int main( )
{
   printf(Escrevendo uma mensagem na tela”)

   sistem("Pause);
   return 0;
}
```

a) O código fonte acima tem a finalidade de escrever uma mensagem na tela. Analise a linha de comando que faz esta tarefa e corrija os erros que encontrar. Anote abaixo o que estava errado.

b) Na linha que está acima do comando **return,** há um comando para fazer uma pausa na tela, mas o programador escreveu o comando errado. Escreva o comando certo nas linhas abaixo:

4. Escreva um programa que apresente na tela o seu nome, na linha de baixo o nome de sua mamãe e na outra linha o nome de sua avó.

5. Escreva um programa que apresente na tela a seguinte saída de dados:

```
MEU
   SEGUNDO
      PROGRAMA
         EM LINGUAGEM C
```

***Dica**: Experimente usar o comando de tabulação da função: printf. Em caso de dúvida consulte a tabela que está no final do capítulo: 1.7 - Executando o programa*

6. Escreva um programa que apresente na tela a linha a seguir:

```
\\\Terceiro programa\\\
```

7. Escreva um programa que apresente na tela a frase:

```
Meu "querido amigo" mudou-se!
```

8. Escreva um programa que exiba na tela as frases:

```
Primeira linha.
         Aqui tem uma linha em branco
Terceira linha.
         Aqui tem outra linha em branco
Quinta linha.
         Aqui temos...
         ...duas linhas em branco
Oitava linha.
```

Anotações

Anotações