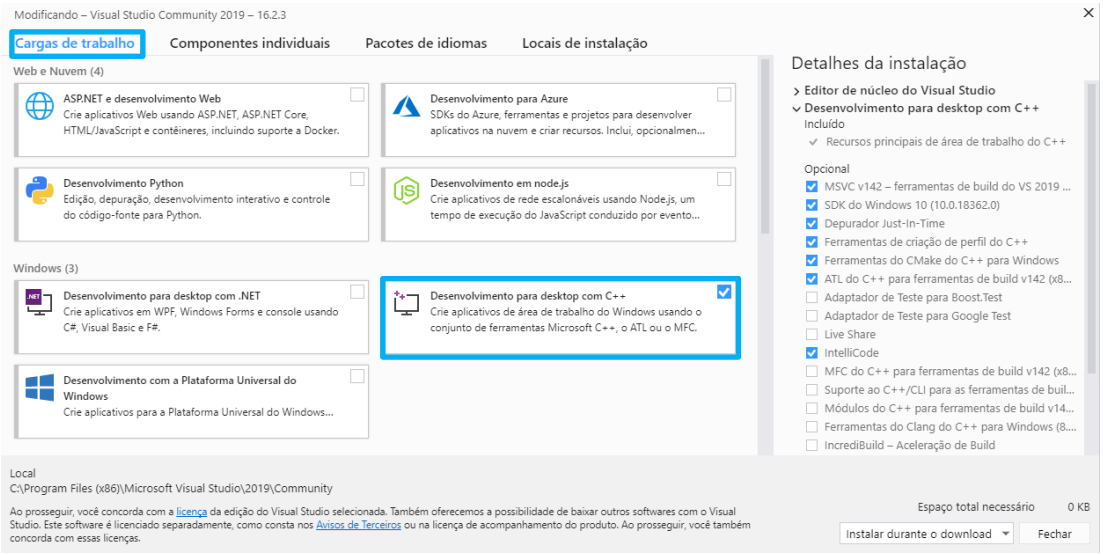
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE COMPUTADORES

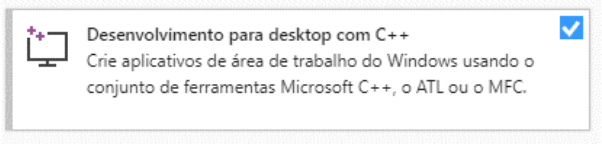
VISUAL STUDIO

INSTALAÇÃO

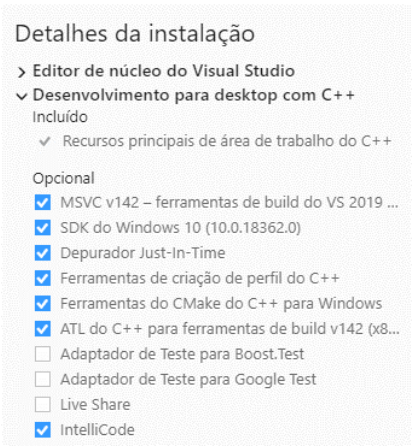
Em Programação de Computadores utilizaremos a última versão do Visual Studio Community Edition com a linguagem C++. Ela não é instalada por padrão e precisa ser selecionada manualmente através da guia **Cargas de trabalho**:



Só é necessário instalar a carga de trabalho “Desenvolvimento para Desktop com C++”:



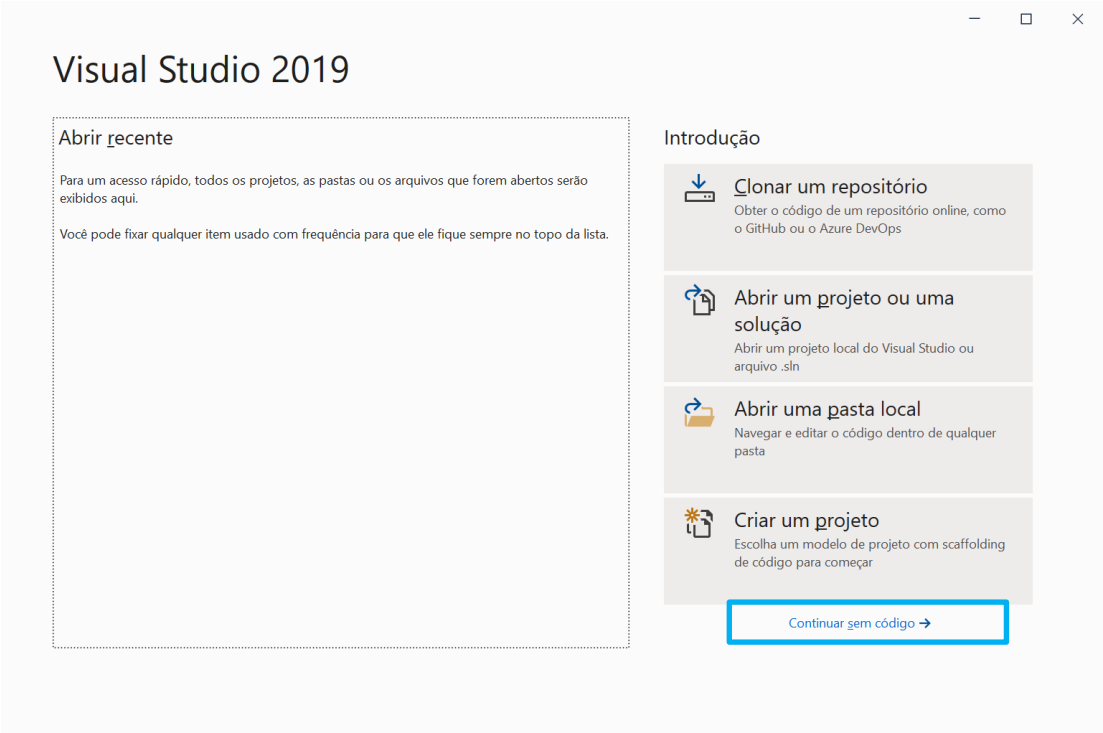
Não é necessário alterar nenhum detalhe da instalação:



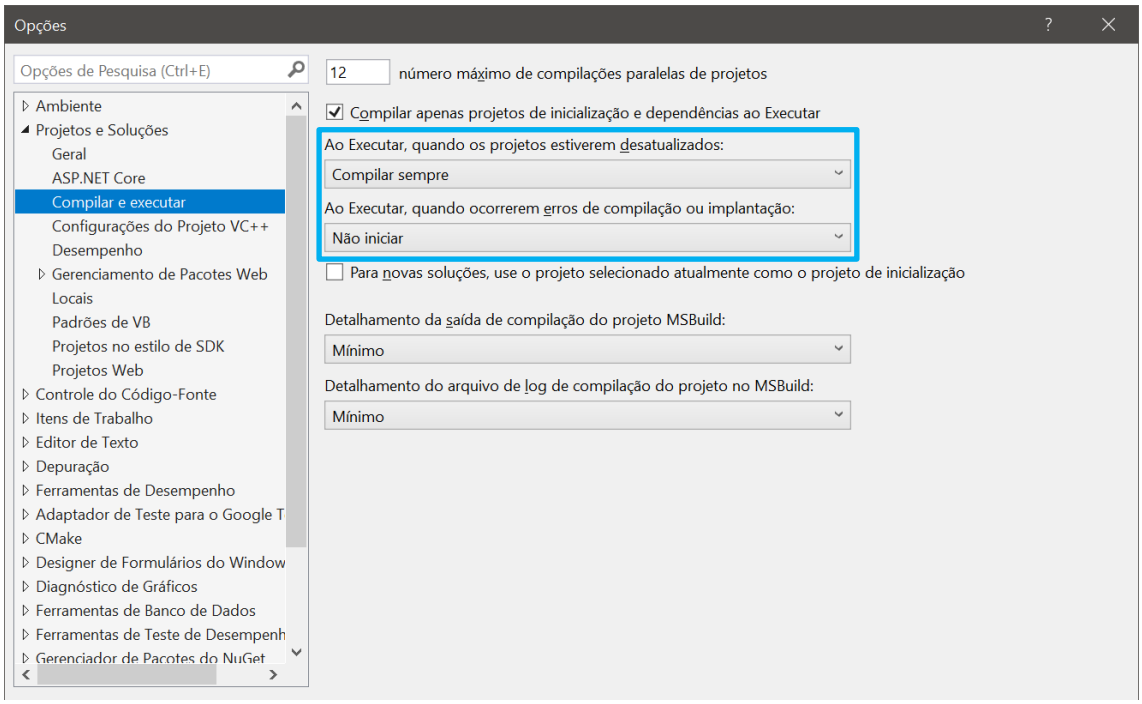
VISUAL STUDIO

CONFIGURAÇÃO

Após a instalação, abram o Visual Studio, selecionem a opção continuar sem código:

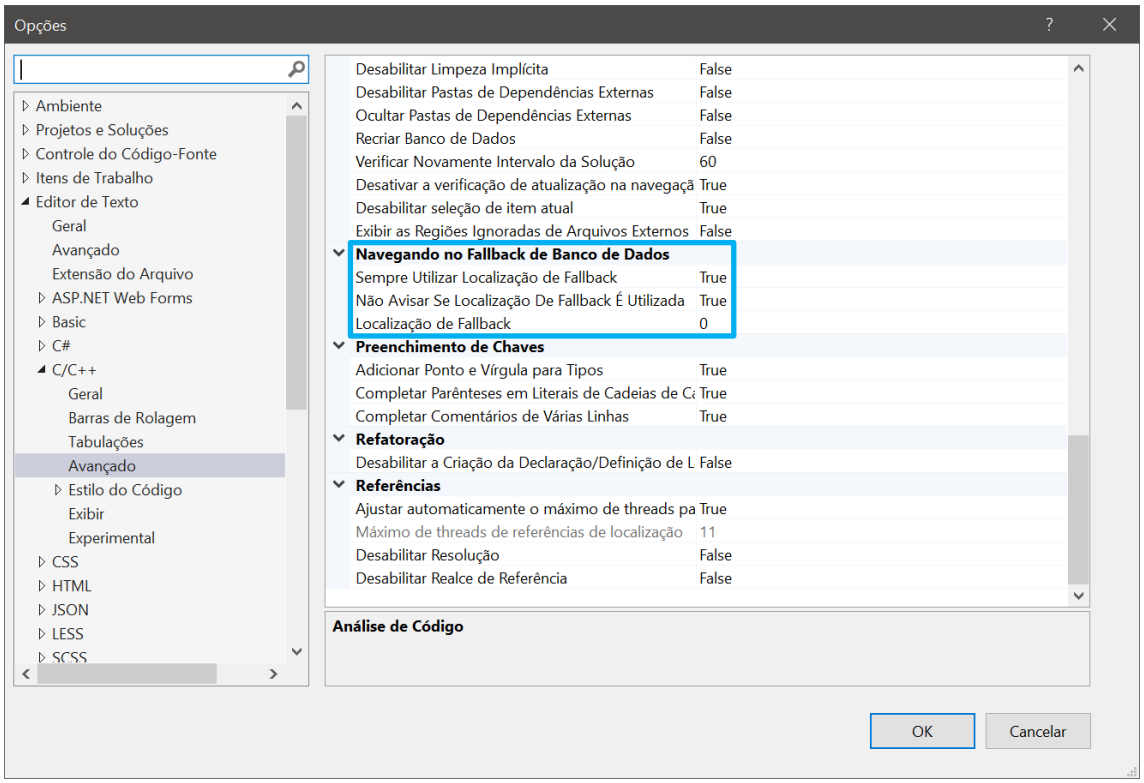


E façam as seguintes alterações através do menu `Ferramentas > Opções`:



Essa alteração evita que o Visual Studio fique repetindo as mesmas perguntas cada vez que um programa é compilado e executado. A importância dessa alteração será vista na aula "Ambiente de Trabalho".

O Visual Studio cria arquivos temporários para controlar a navegação dentro do código. Para evitar que estes arquivos ocupem desnecessariamente espaço na pasta dos projetos, modifiquem a seguinte configuração através do menu `Ferramentas > Opções > Editor de Texto > C/C++ > Avançado`.



A configuração acima fará com que os arquivos temporários do Visual Studio sejam criados na pasta de arquivos temporários do Windows, no lugar de serem criados dentro da pasta do projeto, reduzindo consideravelmente o tamanho dos projetos e tornando-os, assim, mais fáceis de transportar e copiar de um computador para outro.

Feito isso, os únicos arquivos que serão criados dentro do projeto serão aqueles resultantes do processo de compilação. Eles ficam localizados dentro de pastas chamadas **Debug** (ou **Release**), que são criadas tanto na pasta da Solução quanto na pasta do Projeto. Todas as pastas Debug e Release podem ser removidas sem nenhum prejuízo. Elas são recriadas cada vez que o projeto é compilado.

Com essas dicas, o projeto do Visual Studio conterá apenas o código fonte do programa (arquivos .cpp e .h) e alguns arquivos para a solução (.sln) e projeto (.vcxproj) e pode ser facilmente compactado e transportado, ocupando pouco espaço em disco.

VISUAL STUDIO

ACENTUAÇÃO

Quando utilizamos o Visual Studio para criar programas em C++ que rodam no Prompt de Comando do Windows, os acentos não são exibidos corretamente.

```
#include <iostream>
using namespace std;

int main()
{
    cout << "coração" << endl;    // exibe "coraþÒo"
    char nome[20];                // para armazenar um nome
    cin >> nome;                  // digitando "coração"
    cout << nome;                 // exibe "coração"
}
```

Esse é um problema que acontece apenas no Windows e apenas com programas que exibem sua saída como texto no Prompt de Comando. Programando para o ambiente gráfico do Windows, ou nos sistemas operacionais Linux e MacOS, a acentuação funciona normalmente. Isto ocorre porque o Prompt de Comando utiliza páginas de código ASCII para configurar a exibição de texto. Existem várias páginas para atender os diferentes idiomas, como por exemplo: 437 (Estados Unidos), 850 (Multilíngue), 932 (Japão), 1252 (Europa Latina I), etc. O Prompt de Comando do Windows possui inclusive um comando, chamado *chcp* (Change Code Page) para alternar entre as várias páginas de código disponíveis. O página de código utilizada por padrão com o Visual Studio, a página 850, não interpreta corretamente os acentos.

SOLUÇÃO 1

Uma solução para o problema é utilizar a biblioteca `<locale>`.

```
#include <iostream>
#include <locale>
using namespace std;

int main()
{
    locale::global(locale{"pt-BR"});

    char nome[20];
    cout << "coração" << endl;    // exibe "coração"
    cin >> nome;                  // digitando "coração"
    cout << nome;                 // exibe cora╬Æo
}
```

Entretanto, como pode ser constatado pela execução do exemplo acima, o uso dessa biblioteca apenas inverte o problema: a exibição da primeira palavra “coração” fica correta, mas a exibição da palavra “coração” lida do usuário fica errada.

É possível contornar isso alternando as configurações para cada comando de saída, a depender se ele exibe algo vindo do usuário ou do próprio programa.

```
#include <iostream>
#include <locale>
using namespace std;

int main()
{
    char nome[20];

    locale::global(locale{"pt-BR"});
    cout << "coração" << endl;      // exibe "coração"

    cin >> nome;                    // digitando "coração"
    locale::global(locale{"C"});
    cout << nome << endl;           // exibe "coração"
}
```

A solução acima funciona, mas é extremamente inconveniente.

SOLUÇÃO 2

Uma solução mais prática é alterar a código de página do Prompt de Comando. Por padrão, o Windows usa o código de página 850 para o Prompt de Comando. Acontece que ele, embora seja um código de página descrito como “Multilíngue”, não obtém a acentuação da forma correta.

O código de página 1252 foi originalmente baseado em um padrão da American National Standards Institute (ANSI) que eventualmente se tornou o padrão ISO 8859-1, voltado para atender as línguas latinas da Europa, incluindo o português. Esse deveria ser o código de página padrão para os programas escritos em língua portuguesa.

A mudança do código de página pode ser feita no próprio Prompt de Comando executando "`chcp 1252`". Mas cada vez que o Prompt de Comando é fechado, é preciso refazer a chamada a `chcp`. Uma solução mais simples é inserir uma chamada às funções `SetConsoleCP` e `SetConsoleOutputCP` no início de cada programa:

```
#include <windows.h>
#include <iostream>
using namespace std;

int main()
{
    SetConsoleCP(1252);           // entrada
    SetConsoleOutputCP(1252);     // saída

    char nome[20];
    cout << "coração" << endl;    // exibe "coração"
    cin >> nome;                  // digitando "coração"
    cout << nome;                 // exibe "coração"
}
```

Observe que é necessário incluir o arquivo de cabeçalho windows.h para usar as funções.

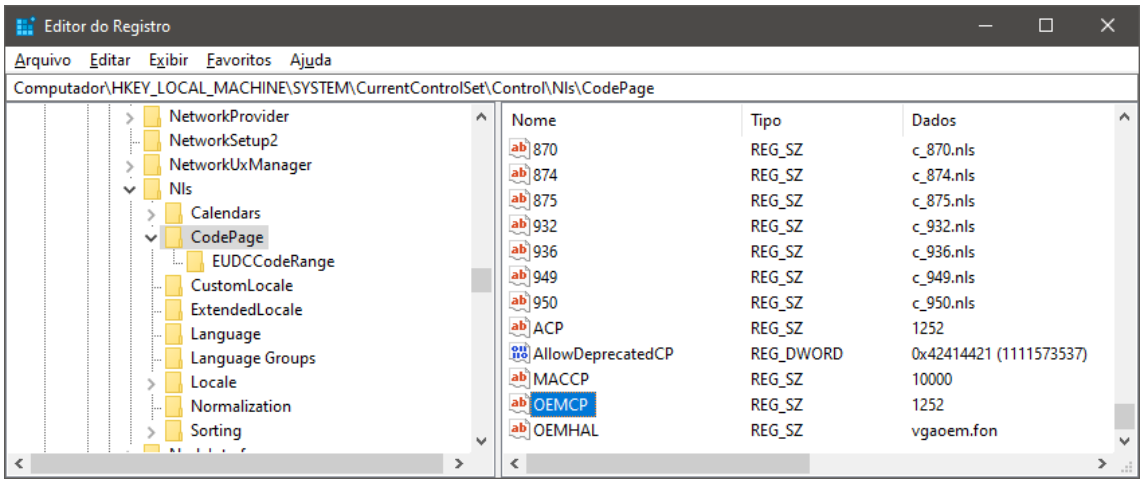
SOLUÇÃO 3

A única forma de obter a acentuação sem fazer nenhuma modificação no programa, é através da mudança da chave OEMCP no registro do Windows. Essa chave define o código de página usada pelo Prompt de Comando.

Para fazer isso é preciso abrir o Editor do Registro do Windows:

- Vá para o Menu Iniciar
- Digite RegEdit
- Abra o Editor do Registro

Com o editor aberto, localize a chave OEMCP no caminho abaixo:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage



Modifique o valor da chave OEMCP de 850 para 1252 e reinicie o Windows. Com essa mudança, o programa abaixo acentua sem precisar fazer nenhuma chamada de função adicional:

```
#include <iostream>
using namespace std;

int main()
{
    char nome[20];
    cout << "coração" << endl;    // exibe "coração"
    cin >> nome;                  // digitando "coração"
    cout << nome;                 // exibe "coração"
}
```

Observe que existe outra chave, chamada ACP (ANSI Code Page), com o valor 1252. Mudando a língua do Windows para Português no Painel de Controle, muda a chave ACP para 1252, mas por alguma razão, não altera a chave OEMCP, que é a utilizada pelo Prompt de Comando.

Embora essa seja uma solução definitiva para o problema da acentuação, ela pode ter efeitos colaterais em outros programas que assumem a configuração padrão do OEMCP. Por essa razão, **não recomendo utilizar essa solução**.