

Este artigo mostra forma de centralizar e manipular a saída do texto em C++

Para ajustar de forma simples você pode usar espaços via string - `std::string(30, ' ')`

Neste caso você está colocando 30 caracteres espaço ' ' na tela

veja o código e saída dele e lembre de colocar `#include <string>`

```
#include <iostream>
```

```
#include <string>
```

```
int main()
{
    int Numero01, Numero02;
    std::cout << std::string(30, ' ') << "Digite o primeiro numero: ";
    std::cin >> Numero01;
    std::cout << std::string(30, ' ') << "Numero01 = " << Numero01 << std::endl;
    std::cout << std::string(30, ' ') << "Digite o segundo numero: ";
    std::cin >> Numero02;
    std::cout << std::string(30, ' ') << "Numero02 = " << Numero02 << std::endl << std::string(30, ' ');
    system("PAUSE");
}
```

```
1  #include <iostream>
2
3  int main()
4  {
5      int Numero01, Numero02;
6      std::cout << std::string(30, ' ') << "Digite o primeiro numero: ";
7      std::cin >> Numero01;
8      std::cout << std::string(30, ' ') << "Numero01 = " << Numero01 << std::endl;
9      std::cout << std::string(30, ' ') << "Digite o segundo numero: ";
10     std::cin >> Numero02;
11     std::cout << std::string(30, ' ') << "Numero02 = " << Numero02 << std::endl << std::string(30, ' ');
12     system("PAUSE");
13
14
15 }
```

C:\Users\curso\source\repos\RecebendoDados\Debug\RecebendoDados.exe

```
30 Digite o primeiro numero: 10
Numero01 = 10
Digite o segundo numero: 20
Numero02 = 20
Pressione qualquer tecla para continuar. . .
```

Existem claro outras opções mais elaboradas

Existe uma função de nome `std::setw(n)`. A função `setw` recebe como parâmetro o tamanho do campo de texto que você deseja colocar na tela e justifica este texto à direita. Portanto na função `setw(n)` este `n` indica o tamanho de caracteres que o campo de saída do fluxo `cout` vai possuir, ou seja, define a largura do campo a ser usada nas operações de saída.

Como assim?

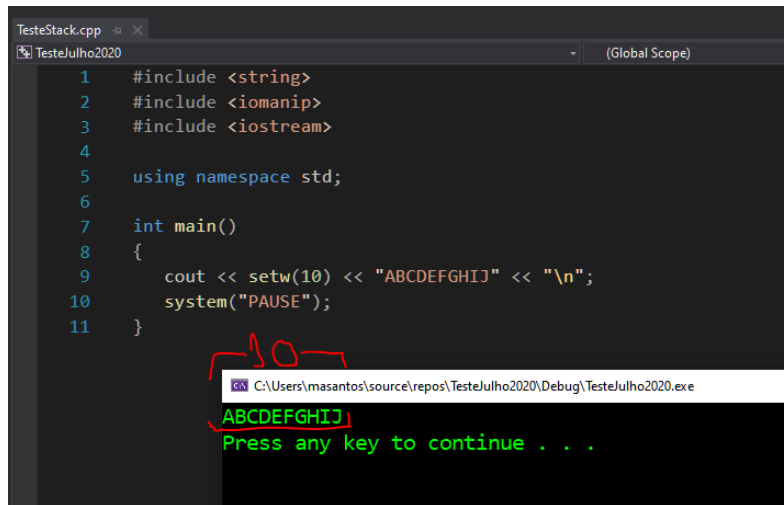
Considere e execute o código abaixo

```
include <string>
#include <iomanip>
#include <iostream>

using namespace std;

int main()
{
    cout << setw(10) << "ABCDEFGHJ" << "\n";
    system("PAUSE");
}
```

Observe que no código acima queremos colocar na tela as letras ABCDEFGHIJ que possui 10 caracteres. Ocorre que se fizermos `setw(10)` não desloca nada, pois você definiu que o campo deve ter a largura de 10 caracteres, ou seja, que o campo de saída do fluxo de `cout` deve ter tamanho 10. Mas ABCDEFGHIJ já tem 10 de largura/tamanho. Logo a saída não desloca nada pois o texto vai se encaixar nesta largura de tamanho 10 e não sobra nada.

The image shows a screenshot of a C++ IDE. The top part displays a code editor with the following code:

```
1 #include <string>
2 #include <iomanip>
3 #include <iostream>
4
5 using namespace std;
6
7 int main()
8 {
9     cout << setw(10) << "ABCDEFGHJ" << "\n";
10    system("PAUSE");
11 }
```

The bottom part shows the output of the program in a console window. The output is:

```
ABCDEFGHJ
Press any key to continue . . .
```

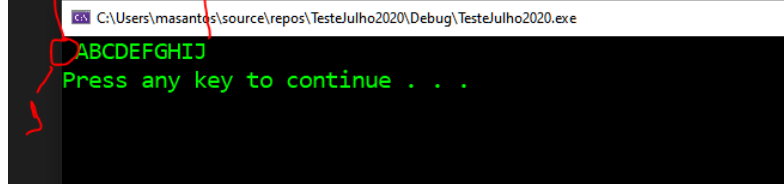
A red bracket is drawn over the number 10 in the code, and a red bracket is drawn over the output text "ABCDEFGHJ".

Agora tente colocar `setw(11)` e veja o que acontece. Neste caso o tamanho, a largura que especificou em `setw` é 11. Como o texto tem 10 caracteres o comando `setw` justifica à direita este texto e o desloca uma unidade

```
#include <string>
#include <iomanip>
#include <iostream>

using namespace std;

int main()
{
    cout << setw(11) << "ABCDEFGHJIJ" << "\n";
    system("PAUSE");
}
```

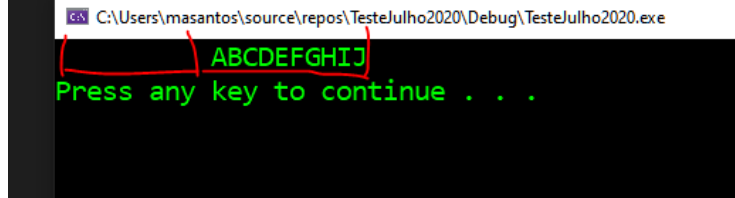


E se você colocar `setw(20)`? Veja que agora a largura do fluxo será 20 e aí o texto será deslocado 10 e ocupará as 10 posições restantes à direita totalizando a largura de 20 especificada por `setw(20)`

```
#include <string>
#include <iomanip>
#include <iostream>

using namespace std;

int main()
{
    cout << setw(20) << "ABCDEFGHJIJ" << "\n";
    system("PAUSE");
}
```



Você pode usar o código abaixo para por exemplo centralizar seu texto através da função `std::setw`

```
#include <string>
#include <iomanip>
#include <iostream>
```

```
using namespace std;

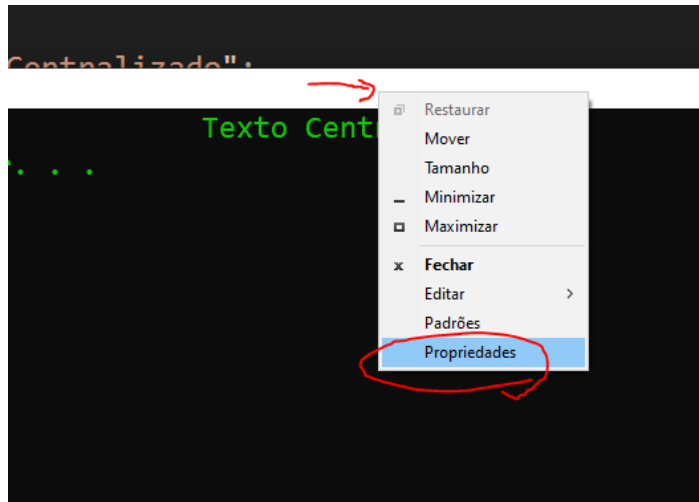
int main()
{
    string str = "Texto Centralizado";
    int console_width = 120; // Largura do seu console.
    int len = str.length(); // Tamanho da String
    cout << setw((console_width / 2) + (len / 2)) << str << endl;
    //ou seja o mesmo que 120/2 = 60 ou setw(60);
    system("PAUSE");
}
```

Observe que `setw((console_width / 2) + (len / 2))` pega o tamanho(largura) da tela que é 120 e dividi por dois. Logo será $120/2 = 60$. Porém ainda temos que considerar o tamanho do texto que tem 18 contando com o caractere espaço. Então para o texto ficar centralizado vai ficar metade na esquerda do centro e metade na direita do centro da tela. Assim temos que somar ao `setw` o tamanho da tela/2 mais o tamanho do texto/2.

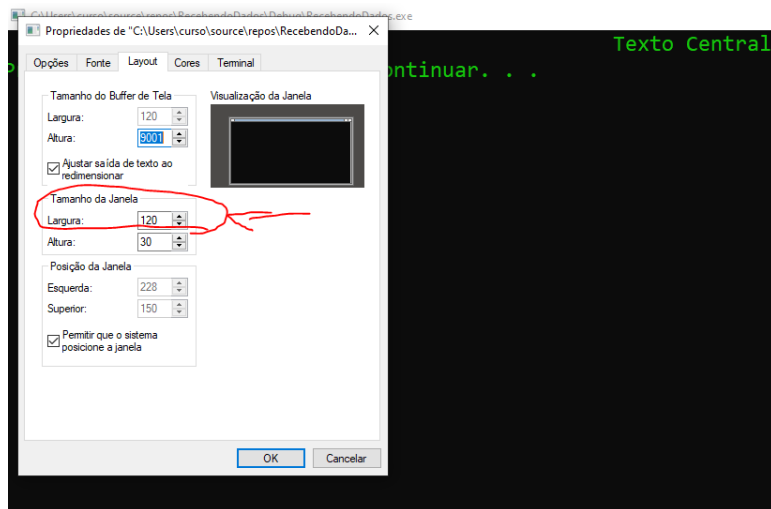


Neste caso o texto será centralizado, mas estamos usando o que chamados de NUMERO MÁGICO para a largura do console e isso não é recomendado. Mesmo porque a largura de cada console pode variar, no geral é 40 mais no meu coloquei 120. Para funcionar você precisa configurar o tamanho do console como 120.

Para ver ou alterar a largura do console clique com botão direito do mouse sobre a barra branca e escolha propriedades

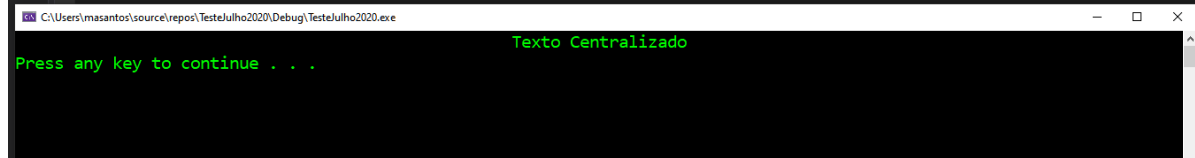


Vá até a aba layout e verifique que podemos ver ou alterar a largura e até altura do terminal. A largura será usada no cálculo para definir a posição do texto centralizado na tela



Execute o código e veja a saída

```
1  #include <string>
2  #include <iomanip>
3  #include <iostream>
4
5  using namespace std;
6
7  int main()
8  {
9      string str = "Texto Centralizado";
10     int console_width = 120; // largura do seu console.
11     int len = str.length(); // Tamanho da String
12     cout << setw((console_width / 2) + (len / 2)) << str << endl;
13     //ou seja o mesmo que 120/2 = 60 + 18/2 ou setw(69);
14     system("PAUSE");
15 }
```



Mas e se você quiser ao executar o programa já modificar automaticamente o tamanho do console windows (CMD) ou prompt de comandos ? Você pode modificar as linhas e colunas da tela com o comando system assim:

```
system("MODE CON: COLS=80 LINES=40");
```

```
#include <iostream>
#include <string>
#include <iomanip>
```

```
using namespace std;
```

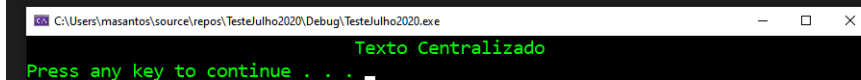
```
int main()
{
    system("MODE CON: COLS=80 LINES=40"); // Já coloca a janela com 80 de largura e 40 altura
    string str = "Texto Centralizado";
    int console_width = 80; // Largura do seu console.
    int len = str.length(); // Tamanho da String
    cout << setw((console_width / 2) + len / 2) << str << endl;
    system("PAUSE");
}
```

Execute o código acima e veja o resultado

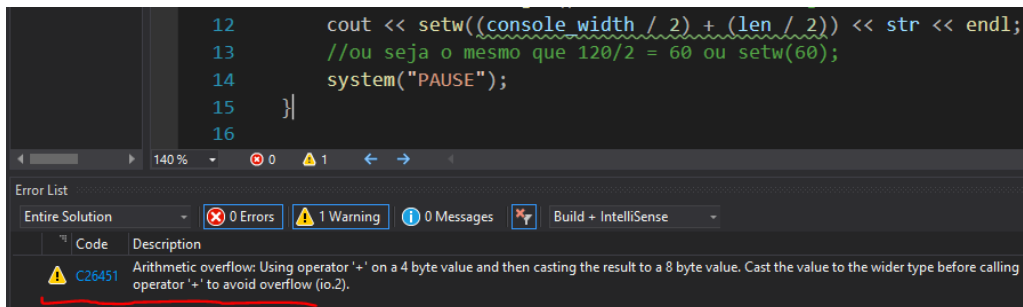
```
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

int main()
{
    system("MODE CON: COLS=80 LINES=40"); // Já coloca a janela com 80 de largura e 40 altura
    string str = "Texto Centralizado";
    int console_width = 80; // Largura do seu console.
    int len = str.length(); // Tamanho da String
    cout << setw((console_width / 2) + (len / 2)) << str << endl;
    system("PAUSE");
}
```



Obs: Caso você receba a informação de aviso(Warning) do compilador



faça o seguinte:

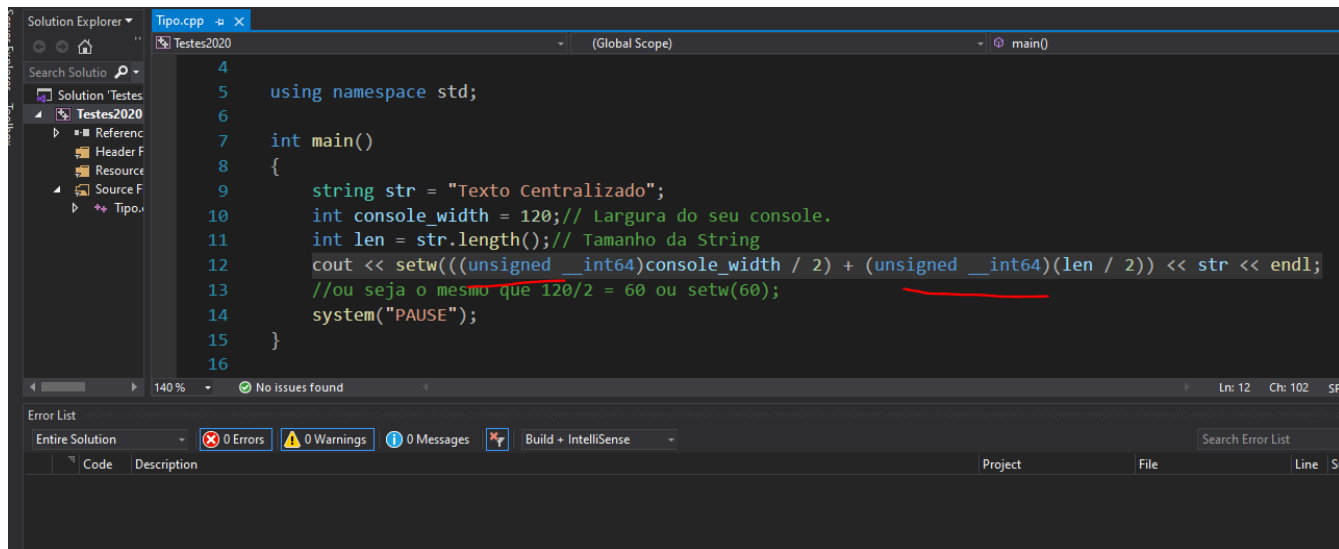
isso é apenas um aviso que indica que os valores limites das variáveis podem ser ultrapassados pois são do tipo inteiro simples. Você pode seguir com o programa.

Mas se quiser que o erro desapareça você pode fazer um CAST para que os valores sejam de um tipo mais robusto com precisão maior de valores

faça o seguinte coloque desta forma a linha do cálculo e compile

```
cout << setw(((unsigned __int64)console_width / 2) + (unsigned __int64)(len / 2)) << str << endl;
```

Veja que agora o aviso não aparece mais



Veja o que informa a Microsoft sobre o tipo `__int64`, basicamente é um tipo para valores altíssimos de variáveis

Link para o artigo: <https://docs.microsoft.com/pt-br/cpp/cpp/int8-int16-int32-int64?view=msvc-160>

Como podemos obter automaticamente o tamanho da tela do console?

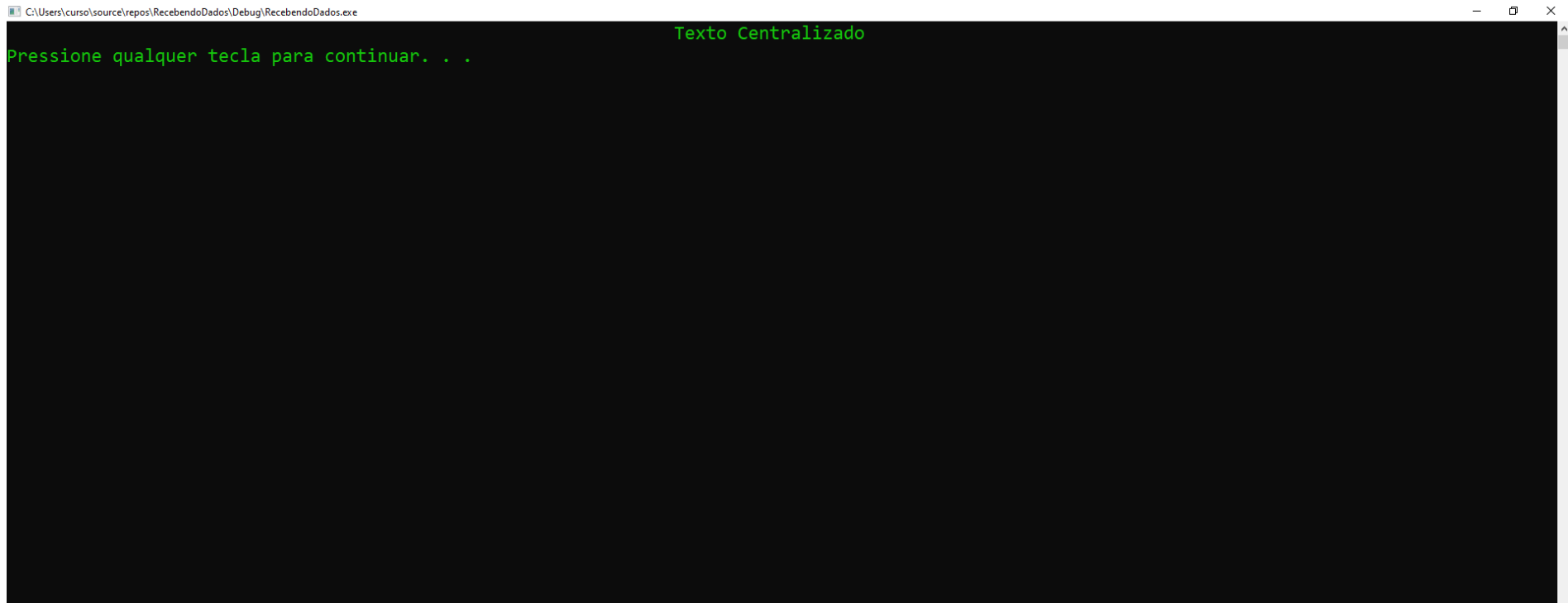
Você pode usar a API do windows.h

obs: **API**(Interface de Programação de Aplicativos) é um conjunto de funções e padrões de programação disponibilizadas por uma plataforma de software(neste caso a plataforma windows). Ou seja, os programadores e programadoras do sistema operacional windows nos fornecem diversas funções, tipos de dados, etc que podemos usar para usar recursos do sistema operacional windows.

Veja

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

```
int main()
{
    HANDLE screen = GetStdHandle(STD_OUTPUT_HANDLE); // Aqui você obtém a janela de Console e coloca em screen um endereço para manipular(HANDLE) esta janela
    COORD max_size = GetLargestConsoleWindowSize(screen);
    //max_size terá as coordenadas X (Largura) e Y (Altura)
    //Então max_size.X será a Largura e max_size.Y será a altura
    //Precisamos apenas da Largura
    string str = "Texto Centralizado";
    int LarguraConsole = max_size.X; // Largura do seu console obtida automaticamente.
    int Tam = str.length(); // Tamanho da String
    if (Tam % 2 == 0) str += " ";
    cout << setw((LarguraConsole / 2) + Tam / 2) << right << str << endl;
    system("PAUSE");
}
```

```
C:\Users\curso\source\repos\RecebendoDados\Debug\RecebendoDados.exe
Texto Centralizado
Pressione qualquer tecla para continuar. . .
```

Há ainda o que você deseja que é colocar o texto em local específico da tela

Na linguagem C existe um função de nome gotoxy mas em C++ podemos criar a própria função usando a API do windows.h

```
#include <iostream>
#include <windows.h>
```

```
COORD GotoXY(int x, int y)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD pos;

    pos.X = x;
    pos.Y = y;

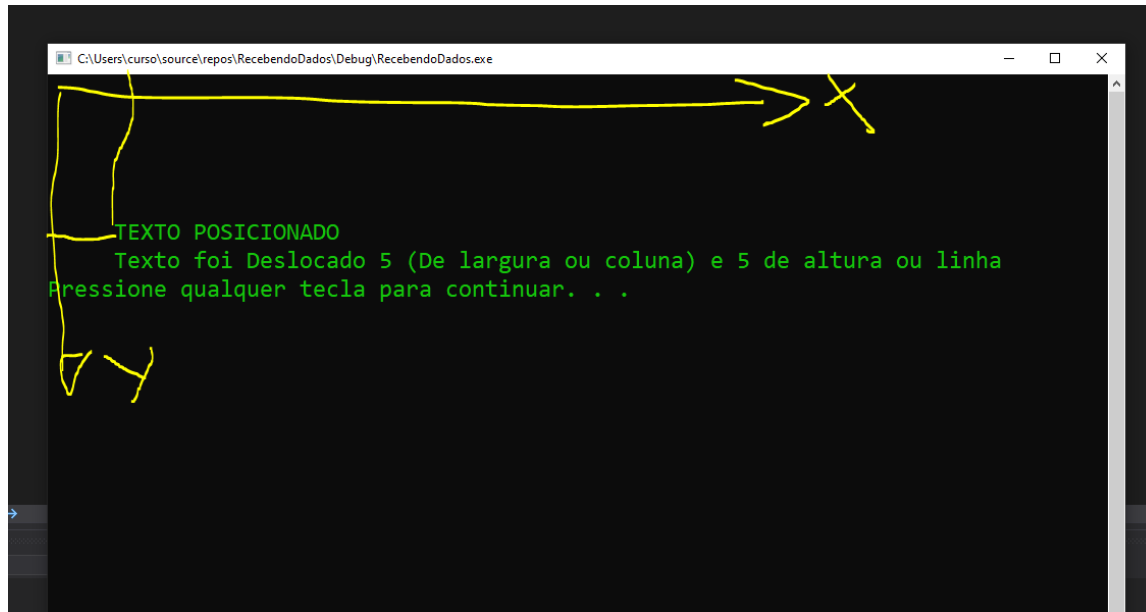
    SetConsoleCursorPosition(hConsole, pos);

    return pos;
}
```

```
int main()
{
    COORD Posicao;
    Posicao = GotoXY(5, 5);
}
```

```
}  
std::cout << "TEXTO POSICIONADO";  
GotoXY(5, 6);  
std::cout << "Texto foi Deslocado " << Posicao.X << " (De largura ou coluna) e " << Posicao.Y << " de altura ou linha " << "\n";  
system("PAUSE");  
return 0;  
}
```

Veja a saída



Você ainda pode usar bibliotecas feitas por terceiros e que simplificam o uso de gotoxy

vá até o site

<https://tapiov.net/rlutil/>

Clique em .zip para fazer o download de um arquivo zipado

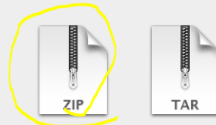
rlutil

Copyright (C) 2010 Tapio Vierros

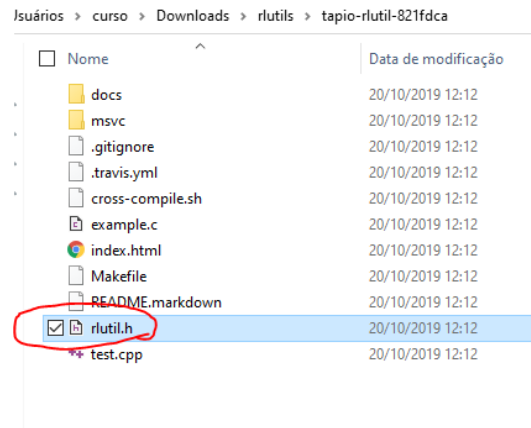
RLUTIL é uma coleção simples de utilitários para ajudar na criação de jogos roguelike em modo de console multiplataforma com C++ e C. Pelo menos essa era a ideia original. Na realidade, na maioria das vezes, contém apenas funções para posicionar e colorir o texto, além de ler a entrada do teclado.

COMEÇO RÁPIDO

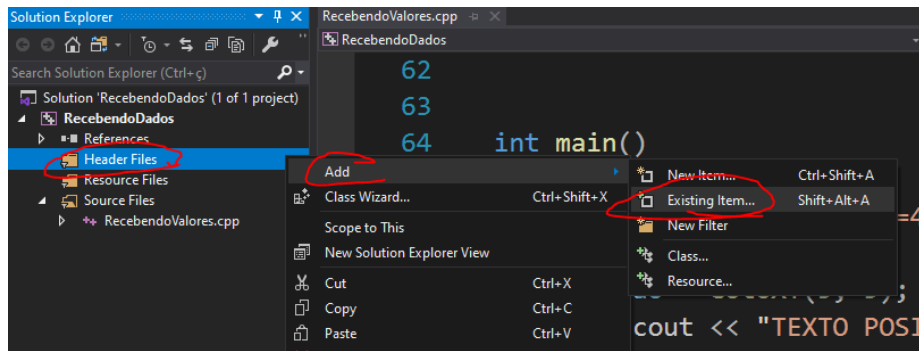
Apenas `#include "rlutil.h"` no seu código fonte. Ele detecta automaticamente se você estiver usando C++ ou C e ajusta o código de acordo (por exemplo, `std::string` vs. `char*`).



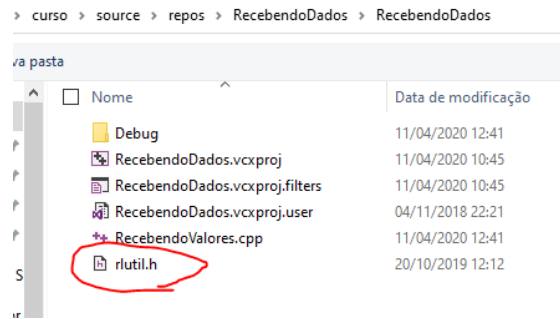
Descompacte e copie o arquivo `rlutils.h` para dentro da pasta de seu projeto



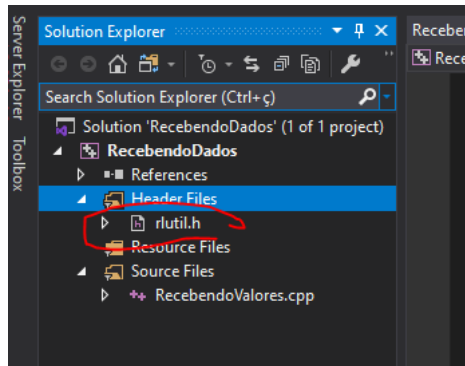
Agora dentro do visual studio clique com botão direito do mouse em header files e escolha ADD->Existing item



Escolha o arquivo que você copiou e clique no botão ADD



Veja que aparece agora



Então no código basta incluir entre aspas duplas este arquivo "rlutil.h". Em aspas duplas pois está no mesmo diretório

```
#include <iostream>
#include "rlutil.h"
```

Veja o código bem mais simplificado

```
#include <iostream>
#include "rlutil.h"

int main()
{
    gotoxy(15,5);
    std::cout << "TEXTO POSICIONADO";
    gotoxy(15,6);
    system("PAUSE");
    return 0;
}
```

E saída

```

RecebendoValores.cpp  X
RecebendoDados  (Global Scope)
1  #include <iostream>
2  #include "rlutil.h"
3
4  int main()
5  {
6      gotoxy(15,5);
7      std::cout << "TEXTO POSICIONADO";
8      gotoxy(15,6);
9      system("PAUSE");
10     return 0;
11 }

```

C:\Users\curso\source\repos\RecebendoDados\Debug\RecebendoDados.exe

```

TEXTO POSICIONADO
Pressione qualquer tecla para continuar. . .

```

Se você abrir em um editor de texto tipo notepad++ verá que usam o mesmo da API do windows.h que citei mais acima

```

RLUTIL_INLINE void cls(void) {
    #if defined(_WIN32) && !defined(RLUTIL_USE_ANSI)
        // Based on https://msdn.microsoft.com/en-us/library/windows/desktop/ms682022%28v=vs.85%29.asp
        const HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        const COORD coordScreen = {0, 0};
        DWORD cCharsWritten;
        CONSOLE_SCREEN_BUFFER_INFO csbi;

        GetConsoleScreenBufferInfo(hConsole, &csbi);
        const DWORD dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
        FillConsoleOutputCharacter(hConsole, (TCHAR)' ', dwConSize, coordScreen, &cCharsWritten);

        GetConsoleScreenBufferInfo(hConsole, &csbi);
        FillConsoleOutputAttribute(hConsole, csbi.wAttributes, dwConSize, coordScreen, &cCharsWritten);

        SetConsoleCursorPosition(hConsole, coordScreen);
    #else
        RLUTIL_PRINT(ANSI_CLS);
        RLUTIL_PRINT(ANSI_CURSOR_HOME);
    #endif
}

```

Abraço