**Evitando que leitura de uma string seja pulada com std::cin.ignore()**

A função cin.ignore () é usada para ignorar ou limpar um ou mais caracteres do buffer de entrada(no nosso caso o buffer do teclado).

Às vezes, precisamos limpar este  buffer do teclado para evitar que o programa PULE a leitura da próxima entrada

Veja no código da aula os comentários adicionais que fiz sobre isso

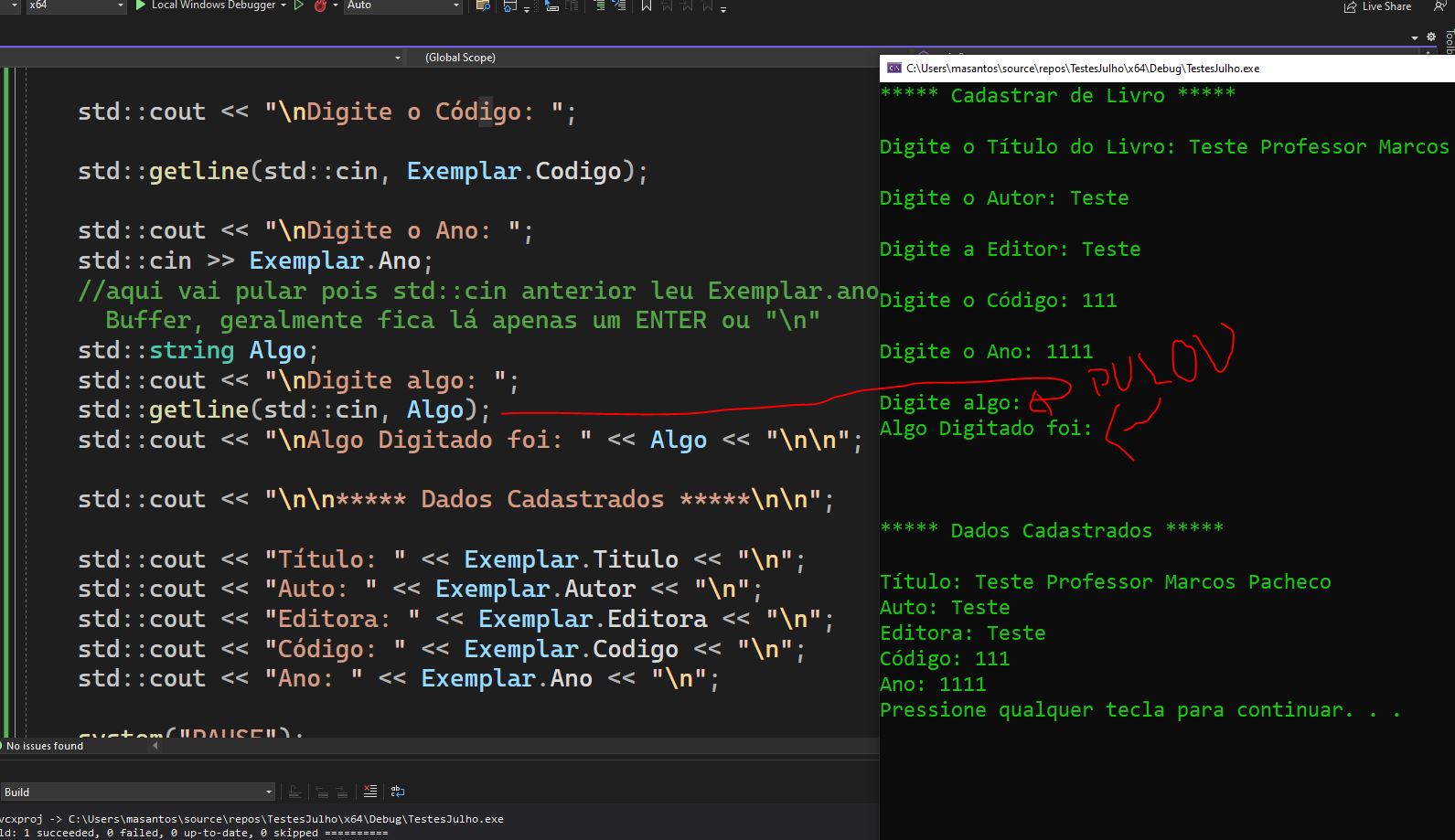
1. std::cout << "\*\*\*Cadastro de Livro\*\*\*" << "\n";
2. std::cout << "Digite ID do Livro: ";
3. std::cin >> livro.ID;
4. //Aqui temos uma entrada cin e desta forma o conteúdo digitado no teclado ficará armazenado em uma região de memória chamado buffer do teclado. Ocorre que se formos ler uma string em seguida estes valores que estão no buffer de entrada ou buffer do teclado serão armazenados na variável livro.Titulo e com isso o fluxo de execução do código "PULA" a leitura da entrada para a esta variável livro.Titulo.
5. /\*Logo toda vez que usarmos std::cin e formos ler um string em seguida será necessário limpar este buffer de teclado para evitar este problema\*/
6. std::cout << "\nDigite o Titulo do livro: ";
7. std::cin.ignore(); // Por isso aqui temos o cin.ignore pois ele vai limpar o buffer e evitar portanto, que este conteúdo do buffer seja colocado na variável livro.Titulo e esta entrada seja PULADA
8. std::getline(std::cin, livro.Titulo);

Observe o seguinte código:

1. #include <iostream>
2. #include <locale.h>
3. #include <string>
5. struct Livro
6. {
7. std::string Titulo;
8. std::string Autor;
9. std::string Editora;
10. std::string Codigo;
11. int Ano;
12. Livro()
13. {
14. Titulo = "Sem Título";
15. Autor = "Sem Autor";
16. Editora = "Sem Editora";
17. Codigo = "Sem Código";
18. Ano = 0;
20. }
21. };

24. int main()
25. {
26. setlocale(LC\_ALL, "portuguese");
27. Livro Exemplar;
29. std::cout << "\*\*\*\*\* Cadastrar de Livro \*\*\*\*\*\n\n";
31. std::cout << "Digite o Título do Livro: ";
32. std::getline(std::cin, Exemplar.Titulo);
34. std::cout << "\nDigite o Autor: ";
36. std::getline(std::cin, Exemplar.Autor);
38. std::cout << "\nDigite a Editor: ";
39. std::getline(std::cin, Exemplar.Editora);
41. std::cout << "\nDigite o Código: ";
43. std::getline(std::cin, Exemplar.Codigo);
45. std::cout << "\nDigite o Ano: ";
46. std::cin >> Exemplar.Ano;
47. //aqui vai pular pois std::cin anterior leu Exemplar.ano e deixou armazenado entradas no Buffer, geralmente fica lá apenas um ENTER ou "\n"
48. std::string Algo;
49. std::cout << "\nDigite algo: ";
50. std::getline(std::cin, Algo);
51. std::cout<< "\nAlgo Digitado foi: " << Algo << "\n\n";
53. std::cout << "\n\n\*\*\*\*\* Dados Cadastrados \*\*\*\*\*\n\n";
55. std::cout << "Título: " << Exemplar.Titulo << "\n";
56. std::cout << "Auto: " << Exemplar.Autor << "\n";
57. std::cout << "Editora: " << Exemplar.Editora << "\n";
58. std::cout << "Código: " << Exemplar.Codigo << "\n";
59. std::cout << "Ano: " << Exemplar.Ano << "\n";
61. system("PAUSE");
62. return 0;
63. }

Veja que ao executar o código o programa pula a leitura da entrada pois um ENTER ficou preso no buffer do teclado e este enter "\n"será colocado como entrada da variável algo



Para resolver isso precisamos colocar um std::cin.ignore(); após a entrada std::cin para que ele limpe este ENTER ou seja limpe o buffer

1. std::cout << "\nDigite o Ano: ";
2. std::cin >> Exemplar.Ano;
3. std::cin.ignore();// <-------------------------COLOQUE AQUI PARA LIMPAR BUFFER E CONSEGUIR LER VARIÁVEL Algo

Veja que agora não pulou

