Texto

O conteúdo gerado por IA pode estar incorreto.

Todo programa deve iniciar com #include <iostream>, caso possua lógica de quase qualquer tipo, visto que esta é a p4rincipal lib.

String não é um tipo primitivo, e portanto é inexistente de forma puramente nativa, assim é necessário importar a lib criada para isso, anteriormente era possível criar strings definindo seu cumprimento, como se fosse um array mas, isso é mais complexo de manter, com esta lib, não é necessário especificar quantidade caracteres.

Todas as funções além da main devem ser definidas ou declaradas antes da main, caso não haja interesse em declarar-las antes ao menos sua definição junto dos argumentos devem ser definidos na porção superior, uma alternativa é criar um arquivo .hpp, que guardará a declaração das funções, que serão definidas no arquivo .cpp com mesmo nome, assim o arquivo .hpp será importado no topo da aplicação, criando a abastração do código e mantendo as regras da linguagem.

Todas as variáveis devem ser declaradas com seus respectivos tipos, como string não é um tipo primitivo deve ser declarada com std::string, mantendo este padrão mesmo quando formos definir uma string como argumento de função.

Std::cout é utilizado para dar saída de texto no programa, como um console.log

O operador `<<` também pode ser usado como operador de deslocamento de bits para a esquerda (bit shift), não é um operador lógico.

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Estes arquivos estão ligados sendo a extensão hpp, um arquivo de cabeçalho e visa apenas declarar as funções de um determinado grupamento lógico. Cpp é o arquivo onde as funções de hpp serão implementadas.

Ifndef significa “se ainda não definido”, logo em seguida o arquivo é definido, isso ocorre para não haverem múltiplas inclusões do mesmo arquivo.

Texto

O conteúdo gerado por IA pode estar incorreto.

Std::cout = emitir uma mensagem na tela

Std::cin = é a estrutura usada para obter informações do usuário, sendo que irá capturar apenas os dados que vierem até a primeira divisão sendo considerados espaços, quebras de linhas etc.

Std::getline = considera tudo até a quebra de linha, porém caso já tenha havido uma quebra de linha será considerada esta quebra na leitura atual, fazendo com que se obtenha uma linha vazia isso ocorre porque o getline pega uma única linha, e já existe um \n, ele interpreta o fim de uma linha e não obtém o restante das informações, o buffer é como um arquivo de texto que recebe o valor do input antes de repassar ao código. Quando se tem o número 23 obtido através de um std::cin, temos isso no buffer:



O std:: cin pega os valores 2 e 3 mas, mantém o \n, quando o getline inicia a leitura encontra uma quebra de linha que é o seu indicador para parar a coleta de informações. Para corrigir isso é necessário usar:

std::getline(std::cin >> std::ws, variavel);

onde std::ws (White space) faz a limpeza de espaços em branco e quebras de linha antes do início da coleta de informação, variável é uma variável que irá receber o valor do input.

g++ main.cpp -o meu\_programa

caso hajam mais arquivos de extensão cpp devem ser colocados logo após o main e após o -o, nome do programa.