I concentrated on using the procedural programming paradigm, which is defined by a series of functions or procedures, when creating the Blackjack game in C. My code's functions all carry out distinct, well-defined tasks, which is in line with the fundamental ideas of procedural programming. For instance, I created functions like shuffle\_deck to mix the cards, and draw\_card to pick the top card from the deck. This approach ensures that my code is organized and each part does one thing well. I also used structures, such as Player, Card, and Deck, in order to handle and portray the game data. In procedural programming, this is a standard procedure where data is first organised and subsequently processed using different methods.

Additionally, I used global variables to control the state of the game. The ability of functions to share and alter the global state in response to programme changes is a fundamental component of procedural programming. For example, variables like player\_sum and computer\_sum are used to keep track of the scores during the game.My approach essentially follows the procedural programming model by utilising functions to manage the game's flow, organising data, and controlling the state of the game through global variables. This method, which is common in procedural programming, guarantees that my code is clear and sequential, making it easy to read and efficient as well.

Finally, my C implementation demonstrates the effectiveness and comprehensibility of procedural programming. I've managed to design a codebase that is simple to learn and maintain by breaking up the game into a number of distinct functions and using global variables to control the game state. Procedural programming's definite structure helps with the development process now and lays a solid basis for any future improvements or debugging endeavours.