Today in CS161

- ***Week #6 Introduction to Functions***
    - What is a function?
    - Why would you want to use a function?
    - How do you define functions?
    - How do you call functions?
- **Write Programs using Functions**
    - Black Jack *(a simple version)*
- **Graphics**
    - User interaction for the tic tac toe program

Functions: What are they?

- Have you ever written code and said
  - Haven’t I already done this before?
- Think about your algorithms..
  - How many times do you open a door or get into your car
  - It is done many times in different contexts
  - These could be functions. Create them once and then reuse them as many times as you need!

Functions: What are they?

- We can write our own functions in C++
- These functions can be called from your main program or from other functions
- A C++ function consists of a grouping of statements to perform a certain task
- This means that all of the code necessary to get a task done doesn't have to be in your main program
- You can begin execution of a function by calling the function

Functions: What are they?

- **When we write algorithms, we should divide our programs into a series of major tasks...**
  - where each major task is a function, called by the main program
- We can group together statements that perform a distinct task and give the overall action a name.
- This is accomplished by writing a C++ function.

Functions: What are they?

- For example, tasks such as driving a car, or cooking breakfast are every day functions that we use.
- The exact details of driving a car or cooking are hidden in the actual process, but even though you don't know the details -- just from the statement "driving a car" you know what that involves and what I am talking about. I don't need to have to tell you that first I get out my keys, then unlock the car door, then get inside, then.....

Functions: What are they?

- The same thing applies to functions in C++.
- A function has a name assigned to it and contains a sequence of statements that you want executed every time you invoke the function from your main program!
- Data is passed from one function to another by using arguments (in parens after the function name).
- When no arguments are used, the function names are followed by: "()".

Functions: Defining Them...

- The syntax of a function is very much like that of a main program.
- We start with a function header:

```
data_type function_name()
{
        <variable definitions>
        <executable statements>
}
```

Functions: Defining Them...

- A function must always be declared before it can be used
- This means that we must put a one-line function declaration at the beginning of our programs which allow all other functions and the main program to access it.
- This is called a **function prototype** (or **function declaration**)
- The function itself can be defined anywhere within the program.

Functions: Using Them...

- When you want to use a function, it needs to be CALLED or INVOKED from your main program or from another function.
- If you never call a function, it will never be used.
- For those of you doing graphics, you have used functions already:

```
setcolor(WHITE);
linexy(x1,y1, x2,y2);
```

Functions: Calling setcolor...

- When we called the setcolor function in graphics.h, we are temporarily suspending execution of our main program (or calling routine) and executing a function called `setcolor` that someone else has written.
- It takes one value as an argument (`an integer color number, where 15 is white`), called **actual arguments** and returns nothing

Let’s try writing a function!

- Let’s write a general algorithm for the game of blackjack.

  keeping it simple, we will have one user play against the computer…
  - Welcome the user and explain the game
  - Deal two cards to the player
  - Allow the player to ask for another card (“hit”) until they are satisfied
  - Player shows their hand
  - Dealer shows their hand
  - Do they want to play again or end the game?

Sub tasks for welcome the user

- For the welcome function we want to explain:
  - There will be only 1 player, playing against the computer
  - The deck will have an infinite number of cards
  - The goal of the game is to assemble a hand whose value is as close to 21 as possible without going over. Whoever gets closest to 21 without going over wins
  - Aces are worth 1 point or 11 points, which is up to the user. Face cards (Jack, Queen, King) are worth 10 points. All other cards are worth the number of points equal to their value.
  - The player gets two cards. After that they can ask to be “hit” with another card or decide to stay with the current hand. They can receive as many cards as they want, until they are ready to stop or over 21.
  - The dealer is forced to stay on any hand worth 17 or more.

Let’s try writing a function!

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

//  Program written by Karla Fant for CS161
//  This program will allow one user to play a simple game of
//  Blackjack against the computer

void welcome ();      //This is where the rules will be described

int main()
{
    welcome();        //calling the function to display the rules
    cin.get();
    return 0;
}
```

Let’s try writing a function!

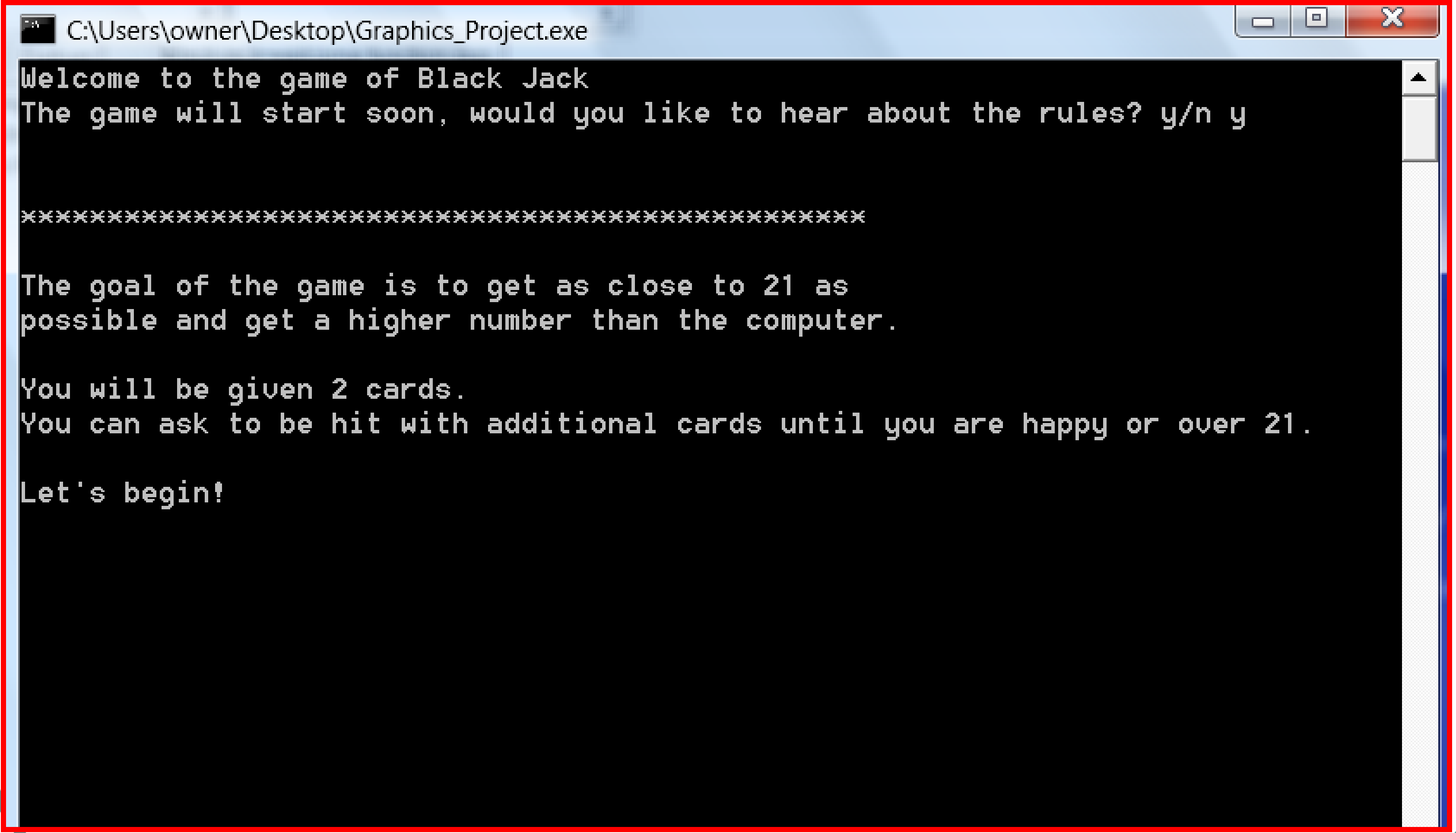
```
//Explaining the rules of the game
void welcome()
{
    char rules='y';     //do you want to see the rules?
    cout <<"Welcome to the game of Black Jack" <<endl;
    cout <<"The game will start soon, would you like"
        <<" to hear about the rules? y/n ";
    cin >> rules;    cin.get();
```

…continued on the next page…

Let’s try writing a function!

```
    if (rules == 'y')          //let's display the rules
    {       cout <<endl <<endl
                <<"************************************************"
                <<endl <<endl;
            cout <<"The goal of the game is to get as close to 21 as"
                <<endl
                <<"possible and get a higher number than the computer."
                <<endl <<endl
                <<"You will be given 2 cards."
                <<endl <<"You can ask"
                <<" to be hit with additional cards until you are"
                <<" happy or over 21. "
                <<endl <<endl;
    }
    cout <<"Let's begin!" <<endl <<endl;
}
```

Welcome to Black Jack…



```
Welcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n y


*************************************************

The goal of the game is to get as close to 21 as
possible and get a higher number than the computer.

You will be given 2 cards.
You can ask to be hit with additional cards until you are happy or over 21.

Let's begin!
```

Let’s try writing a function!

- Notice that in this example we use a function prototype for our function declarations.
- They are very similar to the function header except that they must be terminated by a semicolon....just like any other declaration in C++.

Why write functions?

- You might ask, why go through the trouble to write a program that does no more than the original, shorter version?
- One reason is that functions can be used as prefabricated parts for the easy construction of more complicated programs.
- Another reason is that a function - once created - can be called any number of times without writing its code again.

Why write functions?

- As our programs get more complicated, it is really important that you clearly understand the order in which statements are executed.
- The main program runs first, executing its statements, one after another.
- Even though the functions are declared before the main program (and may also be defined before the main program), they are not executed until they are called.
- They can be called as many times as you wish

Why write functions?

- By giving the task a name, we make it easier to refer to.
- Code that calls clearly named functions is easier to understand than code in which all tasks are described in the main program.
- Programs that use functions are easier to design because of the way they "divide and conquer" the whole problem.

Why write functions?

- By having a function perform the task, we can perform the task many times in the same program by simply invoking the function repeatedly.
- The code for the task need not be reproduced every time we need it.
- A function can be saved in a library of useful routines and plugged into any program that needs it.

Why write functions?

- Once a function is written and properly tested, we can use the function without any further concern for its validity.
- We can therefore stop thinking about how the function does something and start thinking of what it does.
- It becomes an abstract object in itself - to be used and referred to.

Some details about functions:

- Each function declaration can contain declarations for its own...this includes constants, variables.
- These are considered to be LOCAL to the function and can be referenced only within the function in which they are defined

```
data_type some_function()
{
        data_type variable;                 //local variable
 }
```

Why write functions?

- Functions enable us to implement our program in logically independent sections n the same way that we develop the solution algorithm.
- Our main approach could be…

```
welcome();
the user…
    Deal two cards
    Continue to Deal a card (hit) until satisfied
the dealer…
    Deal two cards
    Continue to Deal a card (hit) until reach 17
Select_winner();
```

Dealing cards

- So, what does it mean to "Deal cards"?
  - First we need to know what a card is
  - A card can be Ace,1-10, Jack, Queen, and King
  - Using the modulus operator we can take a random number and mod it by the number of different cards you can get (14). If we get a 0 it will be an Ace, if we get an 11 it will be a Jack, 12 will be a Queen, and 13 a King
  - We are NOT talking about points right now…just how to figure out what a card is.

Dealing cards:

```
void welcome ();      //This is where the rules will be described
int deal_card();      //get a card and find out the points...

int main()
{
   int user_points;  //to hold the number of points the player has
   srand(time(0));
   welcome();        //calling the function to display the rules
   user_points = deal_card();     //Deal a card

   cout <<"You have " <<user_points <<" points" <<endl;

   cin.get();
   return 0;
}
```

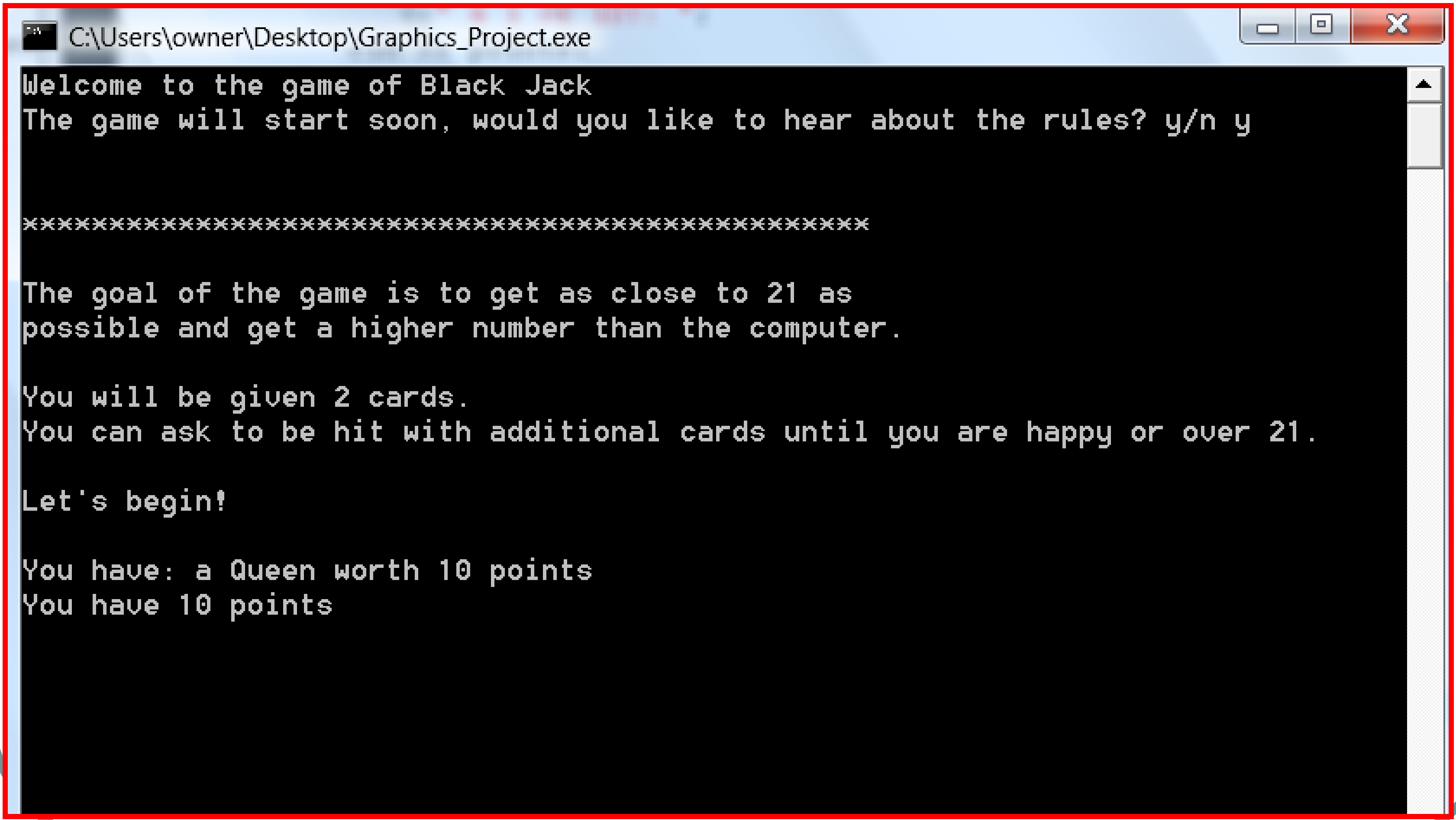
Dealing cards:

```
//Deal a card - one of 14 cards, 0 is Ace, 11 is Jack, 12 Queen, 13 King
int deal_card()
{
   int card = 0;     //Find the numeric value of a card
   int points = 0;

   card = rand() % 14; //Make sure the card is within range 0-13
   if (0 == card)          //ACE!!
   {
       do //find out the value they want to apply to the ace
       {
         cout <<"You have: an Ace " <<endl <<"Do you want to count it as"
           <<" a 1 or 10?: ";
         cin >> points;   cin.get();
       } while (points != 1 && points != 10);  //it has to be correct!
```

Dealing cards:

```
} else if (11 == card)   //Jack!
   {
          cout <<"You have: a Jack worth 10 points" <<endl;
          points = 10;
   } else if (12 == card) //Queen!
   {
          cout <<"You have: a Queen worth 10 points" <<endl;
          points = 10;
   } else if (13 == card)  //King!
   {
          cout <<"You have: a King worth 10 points" <<endl;
          points = 10;
   } else
   {
          cout <<"You were given a face card: " <<card <<endl;
          points = card;
   }
   return points;
}
```

Getting the first card…



C:\Users\owner\Desktop\Graphics_Project.exe

```
Welcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n y


*************************************************

The goal of the game is to get as close to 21 as
possible and get a higher number than the computer.

You will be given 2 cards.
You can ask to be hit with additional cards until you are happy or over 21.

Let's begin!

You have: a Queen worth 10 points
You have 10 points
```

Some details about functions:

- To have a function return a value - you simply say "`return expression`".
- The expression may or may not be in parens.
- Or, if you just want to return without actually returning a value, just say `return;` (note: return(); is illegal).
- If you normally reach the end of a function (the function's closing "}"), its just like saying `return;` and no value is returned.

Some details about functions:

- For functions that don't return anything, you should preface the declaration with the word "void".
- When using void, it is illegal to have your return statement(s) try to return a value
- Also notice, that the type of a function must be specified in both the function declaration and in the function definition.

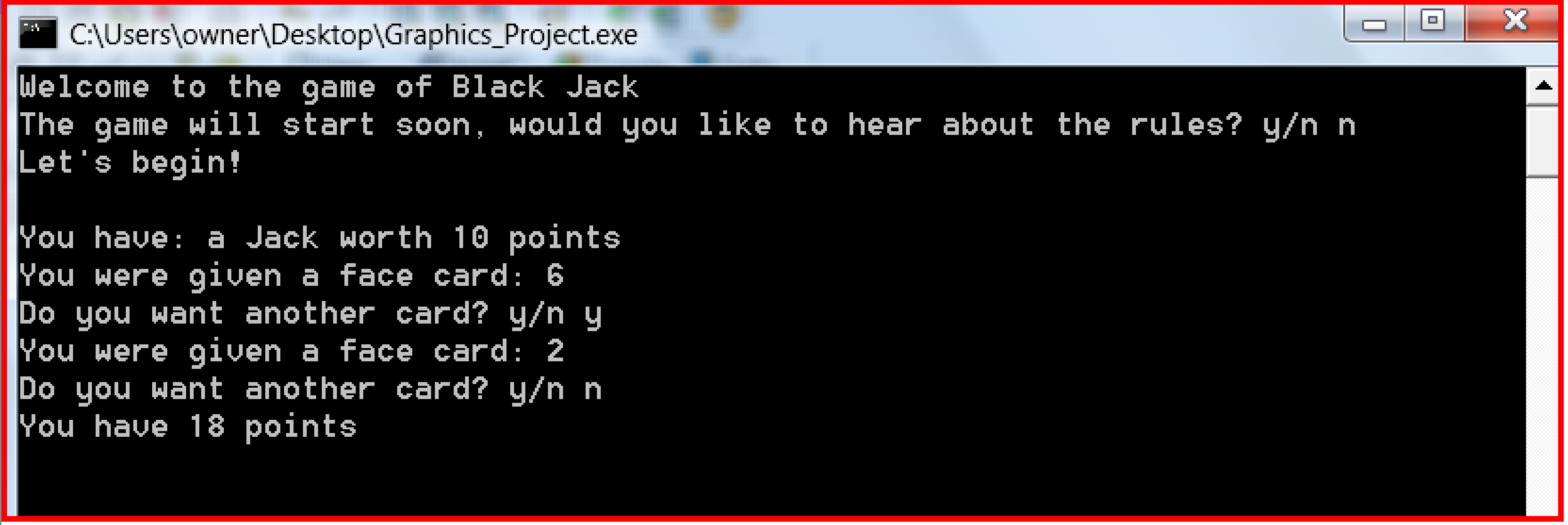
Dealing more cards

- We have seen how to deal one card, but…
  - We need to start off with the player getting two cards and
  - Then getting "hit" as many times as they want.
  - We don't need to change the deal_card function but rather just call it a few more times.
  - Our "algorithm" will be:
  - For the player
    - Deal a card storing the points
    - Deal another card and add those points to our running total
    - Ask the user if they want to be "hit"
      - If so, deal another card and add those points to the running total
      - Continue to do so until the user is happy or until the points are over 21

New main program….

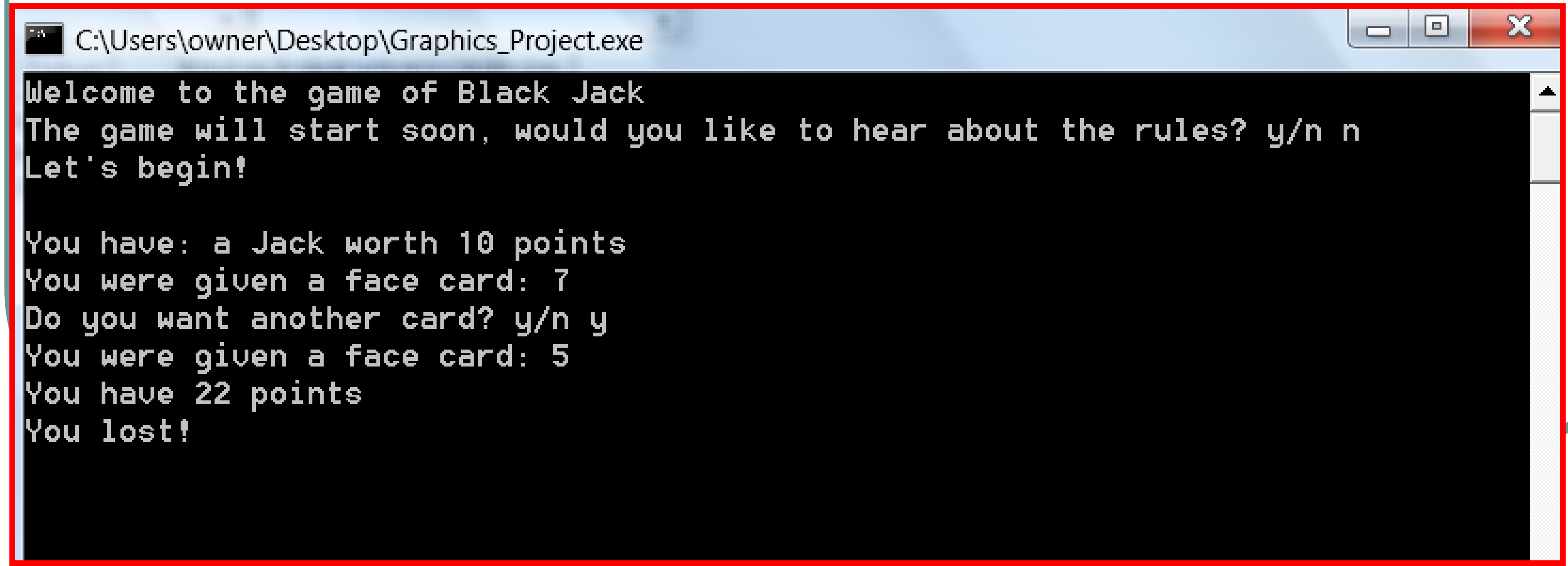
```
int user_points;  //to hold the number of points the player has
char hit;         //do they want to be hit?
srand(time(0));
welcome();        //calling the function to display the rules

//deal the players hand
user_points = deal_card();     //Deal a card
user_points += deal_card();    //Deal the second card
do
{    //Does the user want a “hit”?
          cout <<"Do you want another card? y/n ";
          cin >>hit;     cin.get();
          if ('y' == hit || 'Y' == hit)
                    user_points += deal_card();
} while ((hit == 'y' || hit == 'Y') && user_points < 21);

cout <<"You have " <<user_points <<" points" <<endl;
if (user_points > 21)
     cout <<"You lost! " <<endl;
```

Player gets some cards…



C:\Users\owner\Desktop\Graphics_Project.exe

```
Welcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n n
Let's begin!

You have: a Jack worth 10 points
You were given a face card: 6
Do you want another card? y/n y
You were given a face card: 2
Do you want another card? y/n n
You have 18 points
```



C:\Users\owner\Desktop\Graphics_Project.exe

```
Welcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n n
Let's begin!

You have: a Jack worth 10 points
You were given a face card: 7
Do you want another card? y/n y
You were given a face card: 5
You have 22 points
You lost!
```

Another way to write it…

- This is a good start
- But, think about how we could use functions to make the main simpler?
  - Does the main really need to know about what a hit is?
  - Does the main really need to always check upper and lower case?
- Once we tackle this, we can start applying this to the computer as the opposition.
  - Keeping our main very simple
  - Watch….

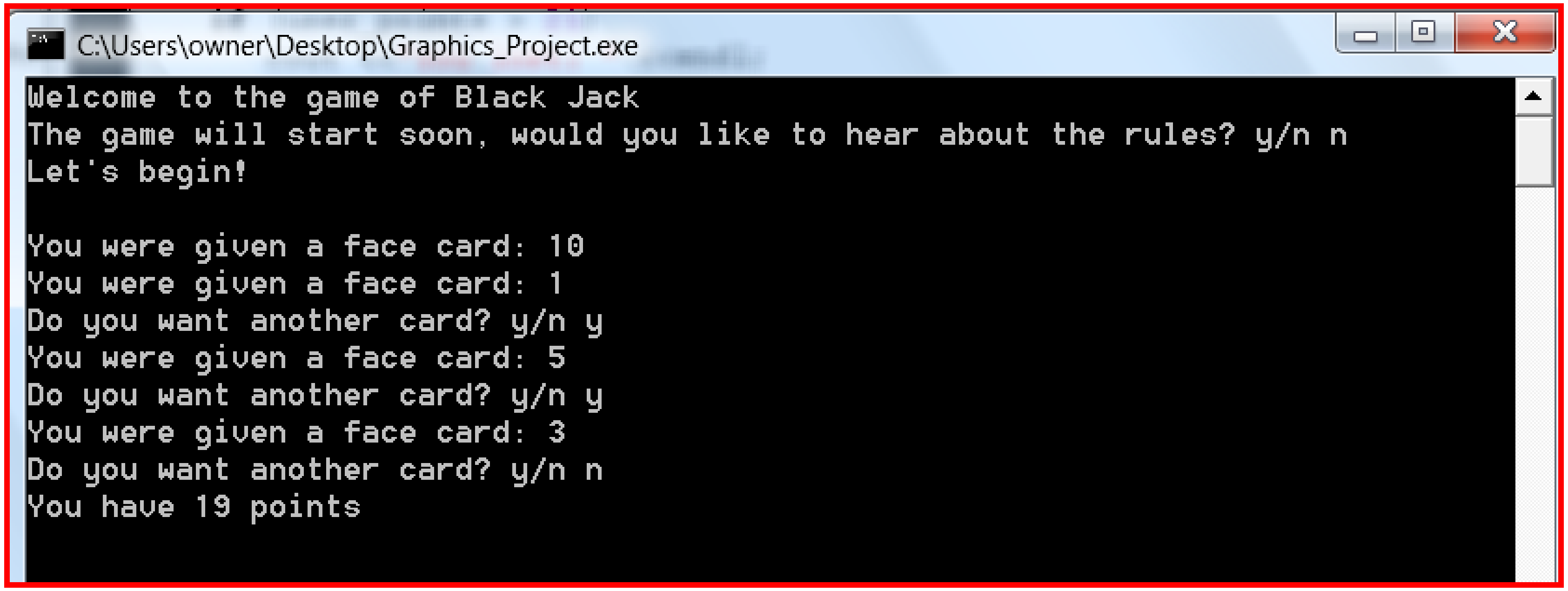
New main program….

```
void welcome ();      //This is where the rules will be described
int deal_card();      //get a card and find out the points...
int more_cards();     //returns the points accumulated

int main()
{
   int user_points;  //to hold the number of points the player has
   srand(time(0));
   welcome();        //calling the function to display the rules

   //deal the players hand
   user_points = deal_card();     //Deal a card
   user_points += deal_card();    //Deal the second card
   user_points += more_cards();   //keep getting cards

   cout <<"You have " <<user_points <<" points" <<endl;
   if (user_points > 21)
     cout <<"You lost! " <<endl;

   cin.get();
   return 0;
```

More cards….as a function

```
//Continue to deal cards until the user is happy or the value is too great
int more_cards()
{
    char hit;
    int points = 0;
    do
    {
                cout <<"Do you want another card? y/n ";
                cin >>hit;     cin.get();
                if ('y' == hit || 'Y' == hit)
                        points += deal_card();
    } while ((hit == 'y' || hit == 'Y') );

    return points;
}
```

Player gets some cards…



C:\Users\owner\Desktop\Graphics_Project.exe

```
Welcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n n
Let's begin!

You were given a face card: 10
You were given a face card: 1
Do you want another card? y/n y
You were given a face card: 5
Do you want another card? y/n y
You were given a face card: 3
Do you want another card? y/n n
You have 19 points
```

Today in CS161

- ***More Functions***
  - **Write Programs using Functions**
    - Black Jack
      - *(add the dealer)*
  - **Arguments**
    - Now let's write a function that takes arguments
  - **Graphics**
    - User interaction for the tic tac toe program

Adding in the dealer….main()

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

//  Program written by Karla Fant for CS161
//  This program will allow one user to play a simple game of
//  Blackjack against the computer

void welcome ();      //This is where the rules will be described
int deal_card();      //get a card and find out the points...
int more_cards();     //returns the points accumulated, prompts user

int main()
{
    int user_points;  //to hold the number of points the player has
    int dealer_points;     //to hold the dealer's points
    srand(time(0));
    welcome();        //calling the function to display the rules
```

Main() continued….*the player*

```
//deal the players hand
cout <<"IT IS NOW THE PLAYERS TURN:" <<endl;
user_points = deal_card();     //Deal a card
user_points += deal_card();    //Deal the second card
user_points += more_cards(); //keep getting cards until 21

cout <<"You have " <<user_points <<" points" <<endl;
if (user_points > 21)
   cout <<"You lost! " <<endl;
else if (user_points == 21)
     cout <<"You won!" <<endl;
else
{
```

Main() continued….*the dealer*

```
 //deal the dealer's hand
  cout <<endl <<endl <<"IT IS NOW THE DEALERs TURN: " <<endl;
  dealer_points = deal_card();
  dealer_points += deal_card();

  while (dealer_points < 17)
      dealer_points += deal_card();

  if (user_points > dealer_points || dealer_points > 21)
     cout <<"You beat the Dealer!! Great job" <<endl;

  else if (user_points == dealer_points) //push
     cout <<"A push...you get your money back!" <<endl;
  else
      cout <<"Better luck next time - Dealer Rules!" <<endl;
}
cin.get();
return 0;
```

The Functions….*deal_card()*

```
//Deal a card - one of 14 cards, 0 is Ace, 11 is Jack, 12 Queen, 13 King
int deal_card()
{
   int card = 0;     //Find the numeric value of a card
   int points = 0;

   card = rand() % 14;
   if (0 == card)     //ACE!!
   {
        do //find out the value they want to apply to the ace
        {
          cout <<"Dealt: an Ace " <<endl <<"Do you want to count it as"
            <<" a 1 or 11?: ";
          cin >> points;     cin.get();
        } while (points != 1 && points != 11);  //it has to be correct!
    }
```

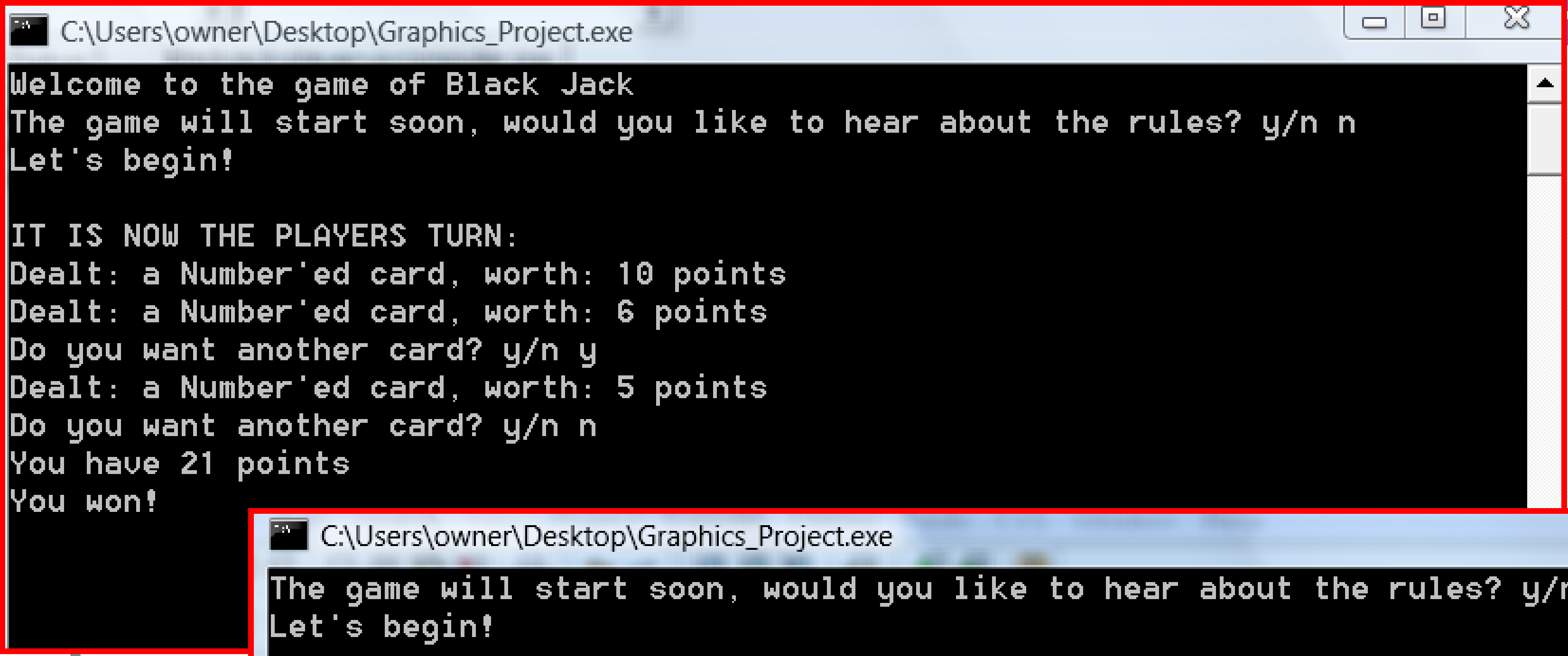
The Functions….*deal_card()*

```
      else if (11 == card)//Jack!
      {
            cout <<"Dealt: a Jack worth 10 points" <<endl;
            points = 10;

      } else if (12 == card) //Queen!
      {
            cout <<"Dealt: a Queen worth 10 points" <<endl;
            points = 10;

      } else if (13 == card)  //King!
      {
            cout <<"Dealt: a King worth 10 points" <<endl;
            points = 10;
      } else   //a numbered card instead…
      {
            cout <<"Dealt: a Numbered card, worth: " <<card <<" points" <<endl;
            points = card;
      }
      return points;
  }
```

The Functions….*more_cards()*

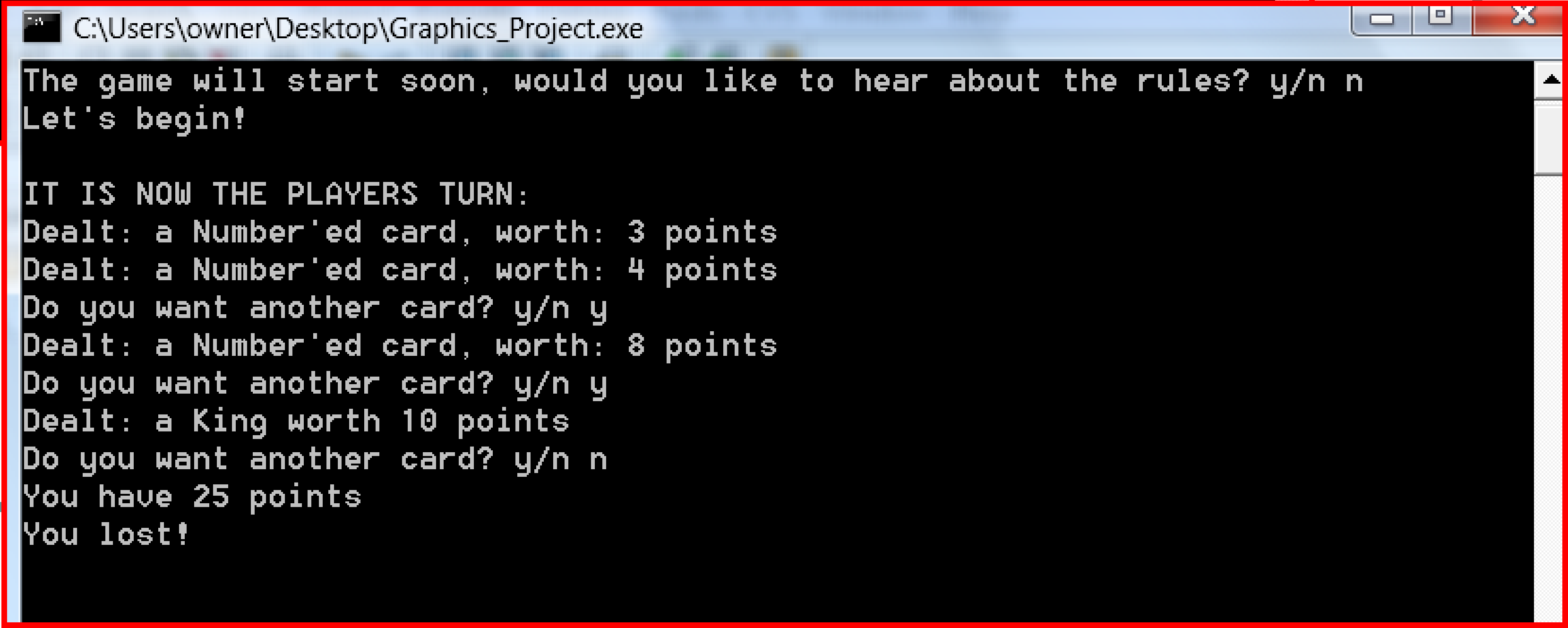
```
//Continue to deal cards until the user is happy or the value is too
    great
int more_cards()
{
    char hit;
    int points = 0;
    do
    {
            cout <<"Do you want another card? y/n ";
            cin >>hit;     cin.get();
            if ('y' == hit || 'Y' == hit)
                    points += deal_card();
    } while ((hit == 'y' || hit == 'Y'));
    return points;
}
```

Playing against the Dealer…



C:\Users\owner\Desktop\Graphics_Project.exe

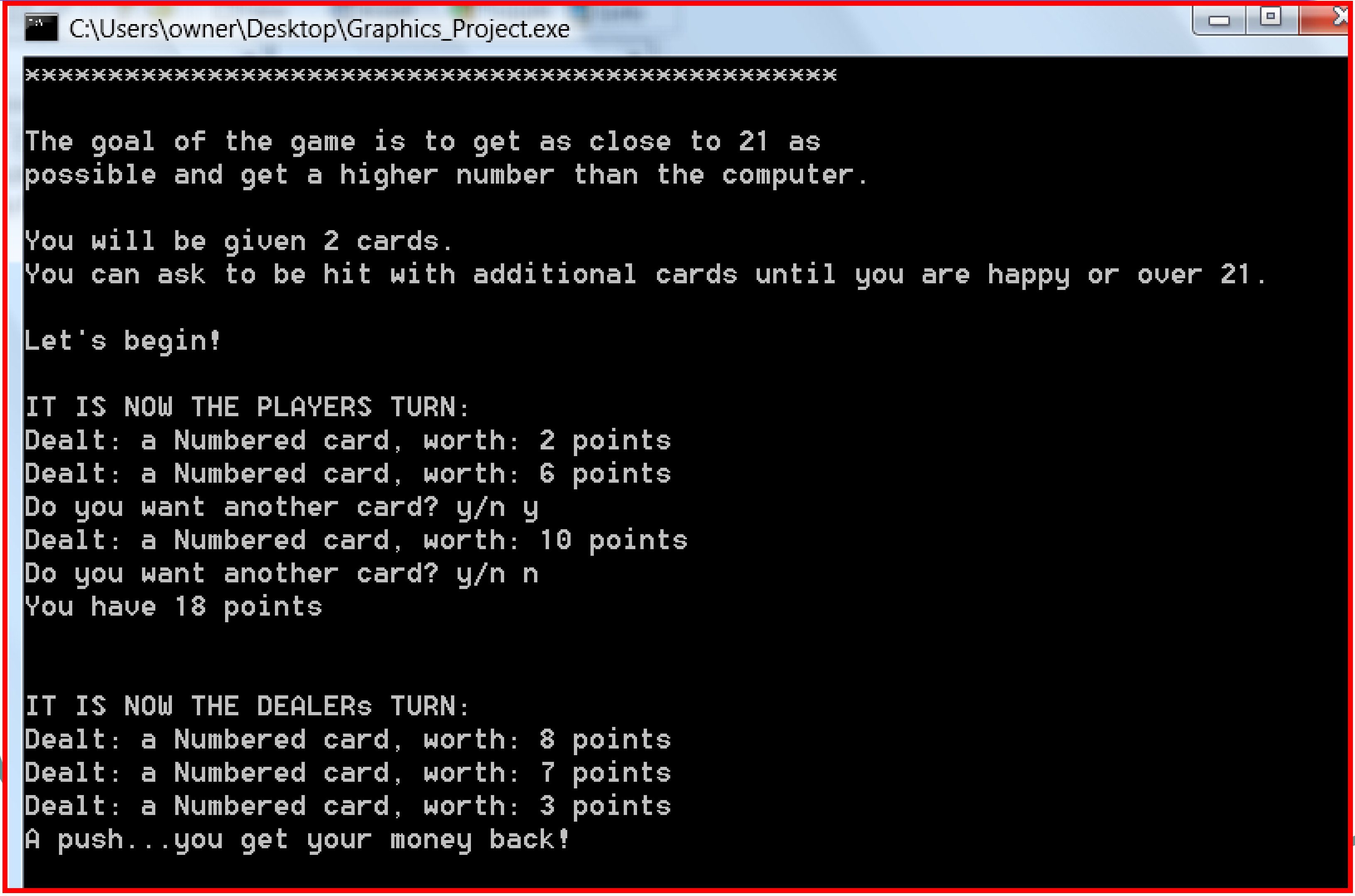
```
Welcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n n
Let's begin!

IT IS NOW THE PLAYERS TURN:
Dealt: a Number'ed card, worth: 10 points
Dealt: a Number'ed card, worth: 6 points
Do you want another card? y/n y
Dealt: a Number'ed card, worth: 5 points
Do you want another card? y/n n
You have 21 points
You won!
```



C:\Users\owner\Desktop\Graphics_Project.exe

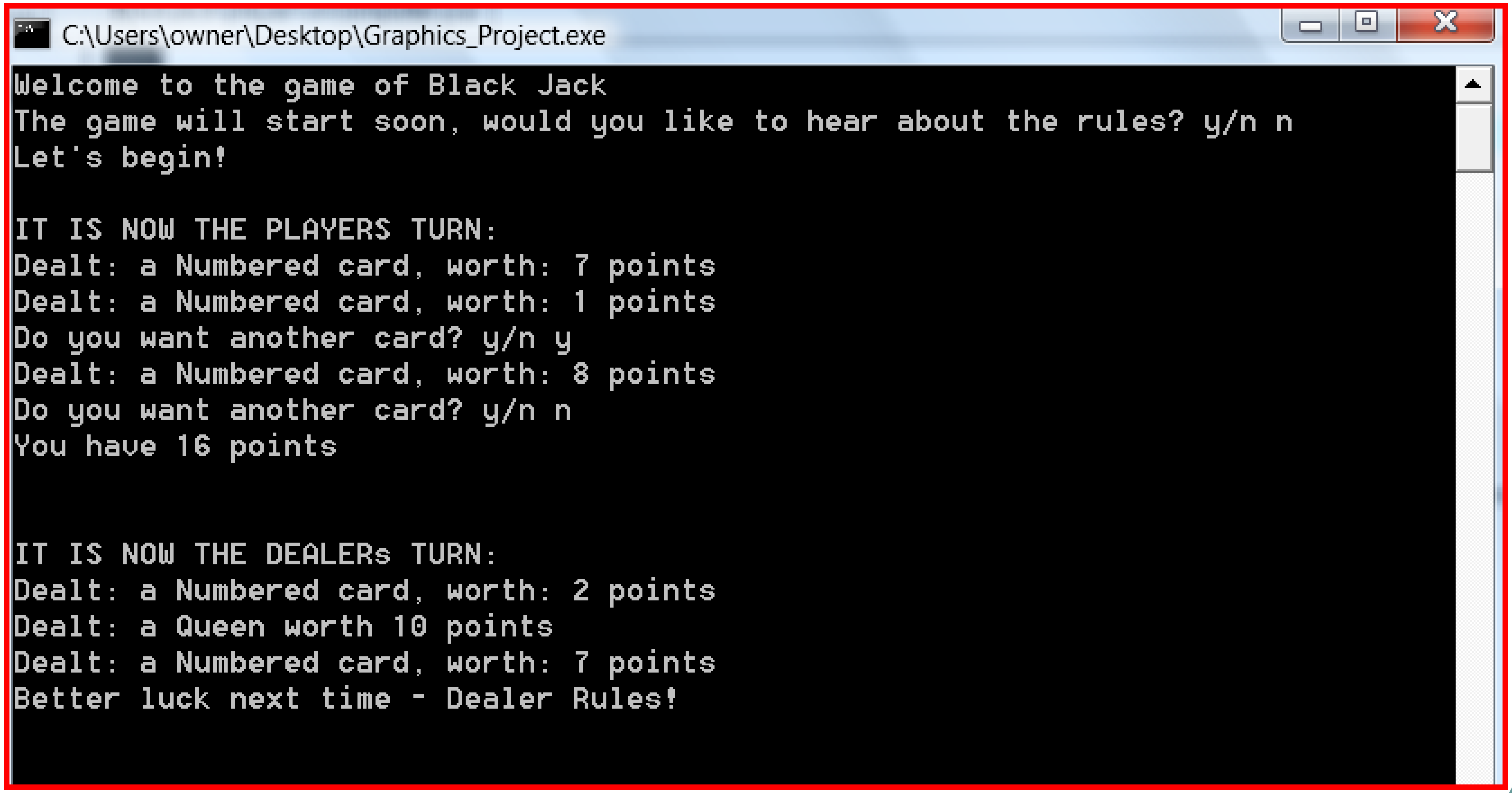
```
The game will start soon, would you like to hear about the rules? y/n n
Let's begin!

IT IS NOW THE PLAYERS TURN:
Dealt: a Number'ed card, worth: 3 points
Dealt: a Number'ed card, worth: 4 points
Do you want another card? y/n y
Dealt: a Number'ed card, worth: 8 points
Do you want another card? y/n y
Dealt: a King worth 10 points
Do you want another card? y/n n
You have 25 points
You lost!
```

Playing against the Dealer…



```
**************************************************

The goal of the game is to get as close to 21 as
possible and get a higher number than the computer.

You will be given 2 cards.
You can ask to be hit with additional cards until you are happy or over 21.

Let's begin!

IT IS NOW THE PLAYERS TURN:
Dealt: a Numbered card, worth: 2 points
Dealt: a Numbered card, worth: 6 points
Do you want another card? y/n y
Dealt: a Numbered card, worth: 10 points
Do you want another card? y/n n
You have 18 points


IT IS NOW THE DEALERs TURN:
Dealt: a Numbered card, worth: 8 points
Dealt: a Numbered card, worth: 7 points
Dealt: a Numbered card, worth: 3 points
A push...you get your money back!
```

Playing against the Dealer…



C:\Users\owner\Desktop\Graphics_Project.exe

```
Welcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n n
Let's begin!

IT IS NOW THE PLAYERS TURN:
Dealt: a Numbered card, worth: 7 points
Dealt: a Numbered card, worth: 1 points
Do you want another card? y/n y
Dealt: a Numbered card, worth: 8 points
Do you want another card? y/n n
You have 16 points


IT IS NOW THE DEALERs TURN:
Dealt: a Numbered card, worth: 2 points
Dealt: a Queen worth 10 points
Dealt: a Numbered card, worth: 7 points
Better luck next time - Dealer Rules!
```

Another way to write it…

- This is a good start
- But, could we get any simpler?
  - How about we break it down further
  - While the user wants to play
    - Call a function to deal for the player
    - Call a function to deal for the computer
    - Find out if the user wants to play again
- By doing this we could go the next step easily to add more players!

New main program….

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

//  Program written by Karla Fant for CS161
//  This program will allow one user to play a simple game of
//  Blackjack against the computer

//Here are the functions we will be using this time…break it down!

void welcome ();                //This is where the rules will be described
int deal_card();                //get a card and find out the points...
int more_cards();                //returns the points accumulated, prompts user
int player();                   //handles the player's hand and returns points
int dealer();                    //handles the dealer's hand and returns points
void ending_message();          //Let them know the game is over
```

New main program….

```
//Now begins main…let's start trying to clean up the main…this will be
    a process
int main()
{

//These are the variables we will be using in main

    int user_points;           //to hold the number of points the player has
    int dealer_points;         //to hold the dealer's points
    char again;                //do you want to play again?
    srand(time(0));
    welcome();                 //calling the function to display the rules
```

New main program….

```
do              //let's allow the user to play as much as they want
   {
      user_points = player();
      if (user_points > 21)
         cout <<"You lost! " <<endl;
      else if (user_points == 21)
         cout <<"You won!" <<endl;
      else
      {
         dealer_points = dealer();
         if (user_points > dealer_points || dealer_points > 21)
            cout <<"You beat the Dealer!! Great job" <<endl;
         else if (user_points == dealer_points)
                       cout <<"A Push – try again next time" <<endl;
             else
            cout <<"Better luck next time - Dealer Rules!" <<endl;
      }
      cout <<endl <<"Would you like to play again? y/n ";
      cin >>again;        cin.get();
   } while (again == 'y' || again == 'Y');
   ending_message();        cin.get();
   return 0;
```

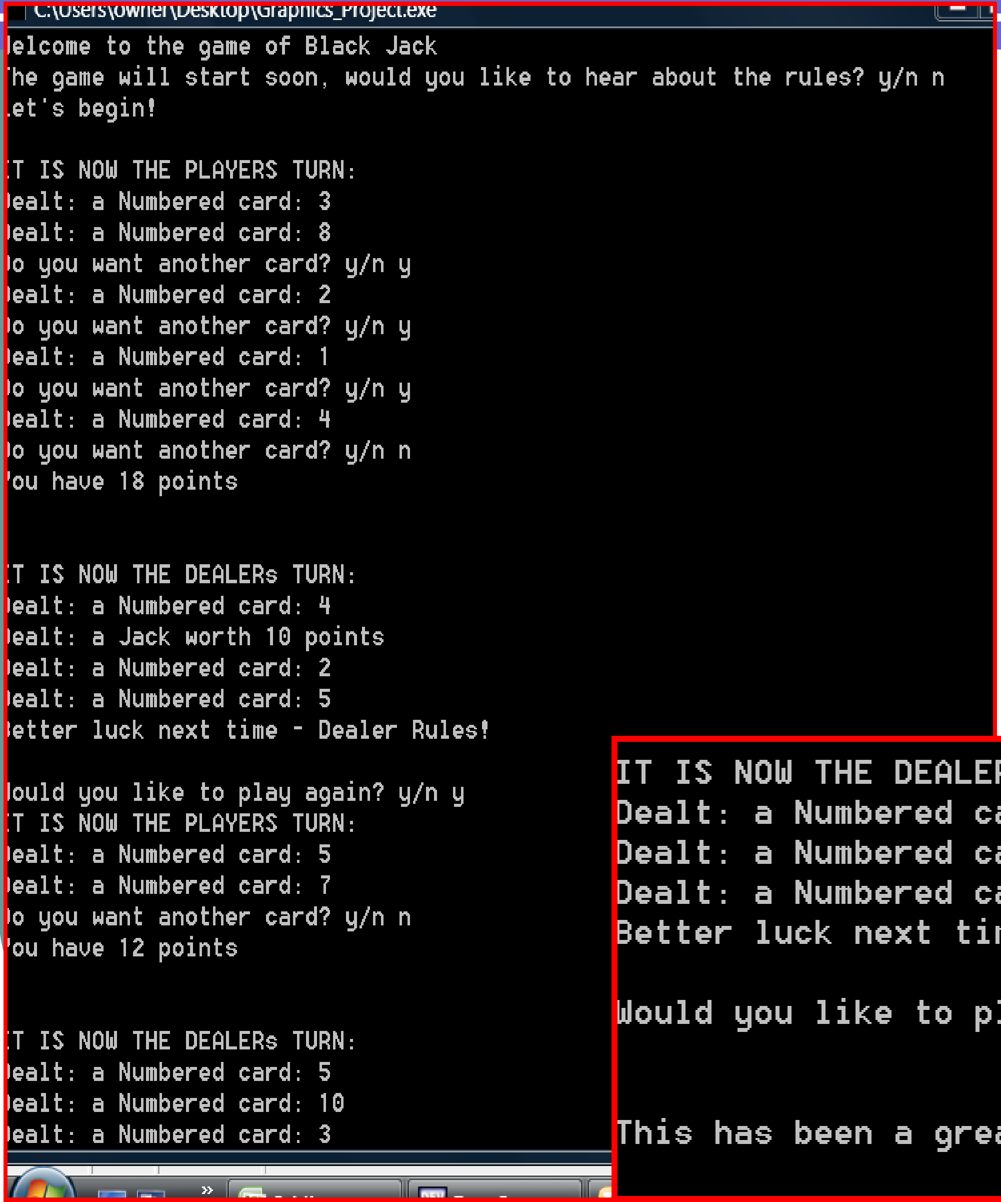
Using functions to simplify

```
//display a termination message so the user knows the program is over.
void ending_message()
{
    cout <<endl <<endl <<"This has been a great game - Until next time!"
        <<endl;
}

//deal the players hand
int player()
{

   int points;    //accumulate points for this player

   cout <<"IT IS NOW THE PLAYERS TURN:" <<endl;
   points = deal_card();     //Deal a card
   points += deal_card();    //Deal the second card
   points += more_cards(); //keep getting cards until 21
   cout <<"You have " <<points <<" points" <<endl;

   return points;
}
```

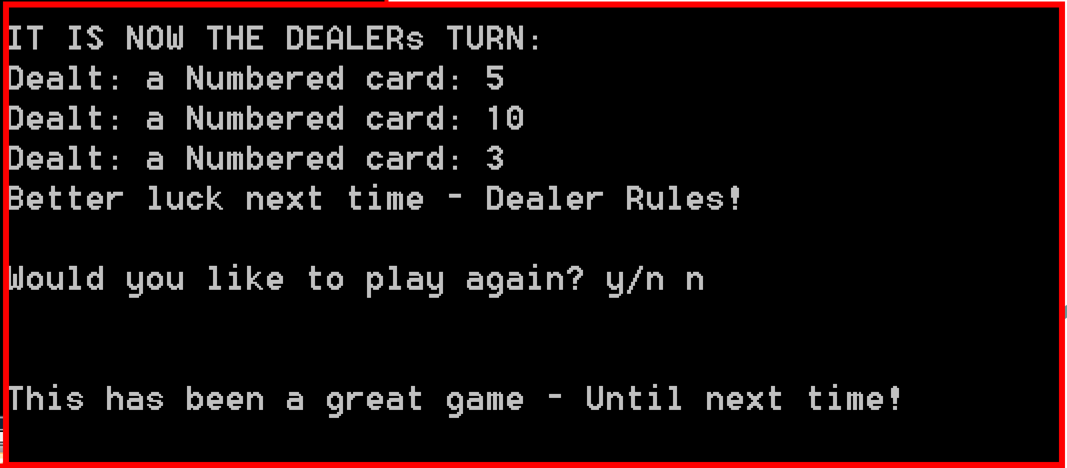
Using functions to simplify

```
//deal the dealer's hand
int dealer()
{
      int points;

      cout <<endl <<endl <<"IT IS NOW THE DEALERs TURN: " <<endl;
      points = deal_card();
      points += deal_card();

        //dealers must accept more cards if their total is less than 17
      while (points < 17)
          points += deal_card();

      return points;
}
```

Playing again…simplifying…



```
C:\Users\owner\Desktop\Graphics_Project.exe
Jelcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n n
.et's begin!

IT IS NOW THE PLAYERS TURN:
Dealt: a Numbered card: 3
Dealt: a Numbered card: 8
Do you want another card? y/n y
Dealt: a Numbered card: 2
Do you want another card? y/n y
Dealt: a Numbered card: 1
Do you want another card? y/n y
Dealt: a Numbered card: 4
Do you want another card? y/n n
'ou have 18 points


IT IS NOW THE DEALERs TURN:
Dealt: a Numbered card: 4
Dealt: a Jack worth 10 points
Dealt: a Numbered card: 2
Dealt: a Numbered card: 5
Better luck next time - Dealer Rules!

Jould you like to play again? y/n y
IT IS NOW THE PLAYERS TURN:
Dealt: a Numbered card: 5
Dealt: a Numbered card: 7
Do you want another card? y/n n
'ou have 12 points


IT IS NOW THE DEALERs TURN:
Dealt: a Numbered card: 5
Dealt: a Numbered card: 10
Dealt: a Numbered card: 3
```



```
IT IS NOW THE DEALERs TURN:
Dealt: a Numbered card: 5
Dealt: a Numbered card: 10
Dealt: a Numbered card: 3
Better luck next time - Dealer Rules!

Would you like to play again? y/n n


This has been a great game - Until next time!
```

Functions: What are arguments?

- If we want to send information to a function when we call it, we can use arguments
- For example, when we supplied one item within the parentheses for the setcolor function – this is an argument that is being passed to the function setcolor!
- We can define functions with no arguments, or with many arguments

Functions: What are arguments?

- **If we go back to our blackjack game**
  - We could make our main() simpler by making a winner’s function that takes the players points and the dealers points and displays the result based on what is passed.
  - So, we want to write a function with two arguments – both integers.
  - void winner(int player, int dealer);
  - The purpose of the function will be to display the appropriate message based on who has the best hand!

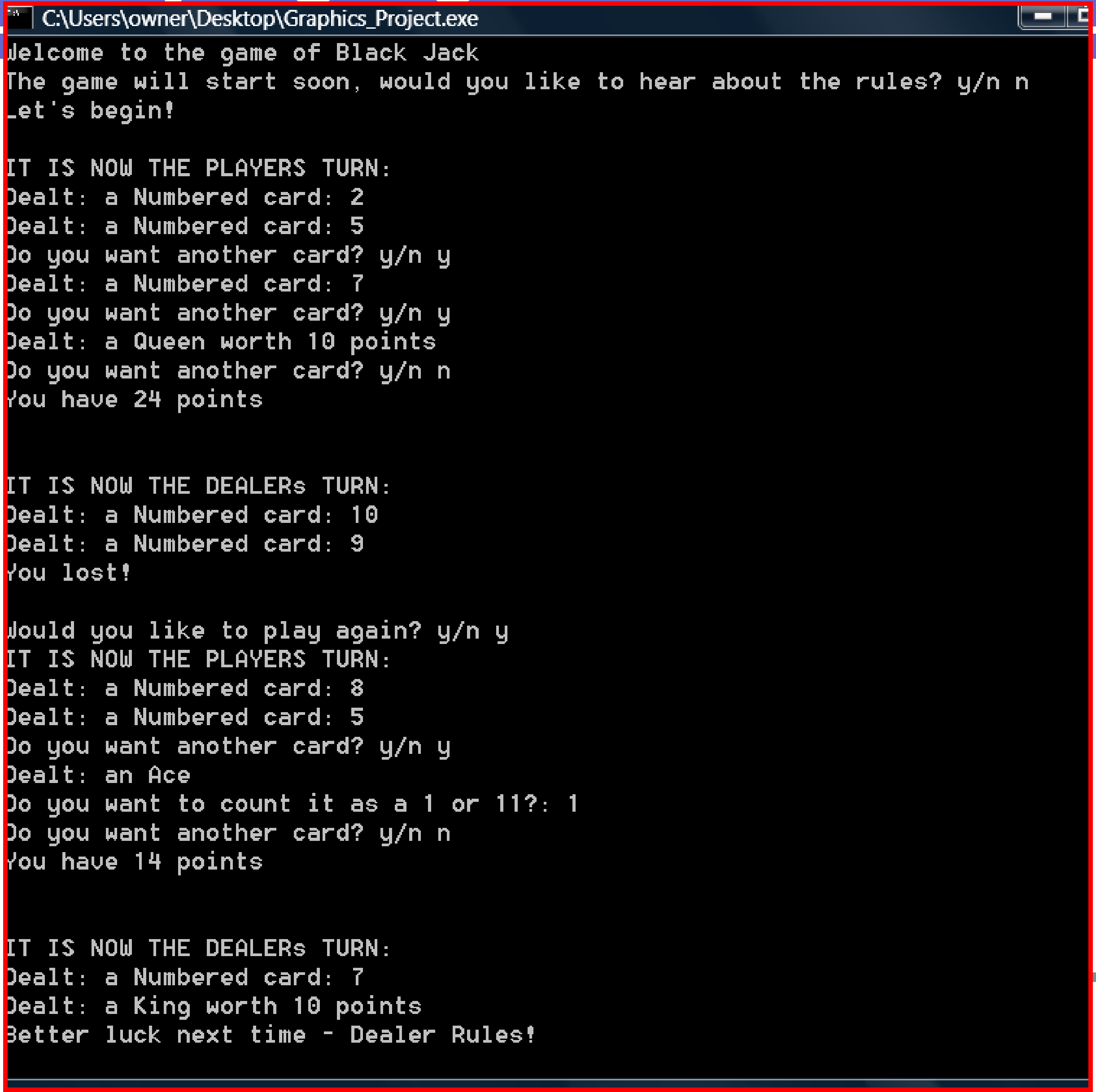
New main program….

```
void winner(int player, int dealer);   //display the winning message
int main()
{
    int user_points;                   //to hold the number of points the player has
    int dealer_points;                 //to hold the dealer's points
    char again;                        //do you want to play again?
    srand(time(0));
    welcome();                         //calling the function to display the rules

    do              //let's allow the user to play as much as they want
    {
        user_points = player();
        dealer_points = dealer();
        winner(user_points, dealer_points);

        cout <<endl   <<"Would you like to play again? y/n ";
        cin >>again;                  cin.get();
    } while (again == 'y' || again == 'Y');
    ending_message();            cin.get();
    return 0;
}
```

The winner function….

```
//Display a message as to which player wins
//Return true if the player doesn't win
void winner(int player, int dealer)
{
    //First let's see if the player is the clear winner
    if (player > 21)
      cout <<"You lost! " <<endl;
    else if (player == 21)
        cout <<"You won!" <<endl;

    //now let's see how the dealer did!
    else if (player > dealer || dealer > 21)
        cout <<"You beat the Dealer!! Great job" <<endl;
    else if (player == dealer)
        cout <<"Push - try again next time" <<endl;
    else
        cout <<"Better luck next time - Dealer Rules!" <<endl;
}
```

Playing again…with functions…



C:\Users\owner\Desktop\Graphics_Project.exe

```
Jelcome to the game of Black Jack
The game will start soon, would you like to hear about the rules? y/n n
Let's begin!

IT IS NOW THE PLAYERS TURN:
Dealt: a Numbered card: 2
Dealt: a Numbered card: 5
Do you want another card? y/n y
Dealt: a Numbered card: 7
Do you want another card? y/n y
Dealt: a Queen worth 10 points
Do you want another card? y/n n
You have 24 points


IT IS NOW THE DEALERs TURN:
Dealt: a Numbered card: 10
Dealt: a Numbered card: 9
You lost!

Would you like to play again? y/n y
IT IS NOW THE PLAYERS TURN:
Dealt: a Numbered card: 8
Dealt: a Numbered card: 5
Do you want another card? y/n y
Dealt: an Ace
Do you want to count it as a 1 or 11?: 1
Do you want another card? y/n n
You have 14 points


IT IS NOW THE DEALERs TURN:
Dealt: a Numbered card: 7
Dealt: a King worth 10 points
Better luck next time - Dealer Rules!
```

Functions: What are arguments?

- **When you call winner,**
  - There are two arguments
  - Both are integers
  - A copy is made of both arguments from the function call
  - And sent to the function as the initial value for that function's arguments
  - Inside the function, the arguments act as local variables.

Functions: What are arguments?

- When the function call is executed,
  - the actual arguments are conceptually copied into a storage area local to the called function.
  - If you then alter the value of a formal argument, only the local copy of the argument is altered.
  - The actual argument never gets changed in the calling routine.

Functions: What are arguments?

- C++ checks to make sure that the number and type of actual arguments sent into a function when it is invoked match the number and type of the formal arguments defined for the function.
- The return type for the function is checked to ensure that the value returned by the function is correctly used in an expression or assignment to a variable.

Functions: What are arguments?

- Technically, what we are doing with the arguments to setcolor, or winner is called PASS BY VALUE
    - the value of the arguments in the function call cannot be modified by the function.
    - This allows us to use these functions, giving literals and constants as arguments without having conflicts.
    - This is the default way of doing things in C++.

Functions: More possibilities

- **Can we even further simplify?**
  - We could make a play_again function that would find out if the user wants to play again and make sure to capitalize their answer so that your main won't have to worry about it.

    bool play_again();
  - Revise the deal_card function to take an argument. If the argument indicates that it is the dealers turn, then the ace is either a 1 or an 11 depending on which is calculated to be "best" and not put the dealer over 21.

    int deal_card(int dealer=-1);

```
bool play_again();                              //Do you want to play again?
int deal_card(int dealer=-1);                   //get a card and find out the points...

int main()
{
    int user_points;                  //to hold the number of points the player has
    int dealer_points;                //to hold the dealer's points

    srand(time(0));
    welcome();                        //calling the function to display the rules

    do                                //let's allow the user to play as much as they want
    {
        user_points = player();
        dealer_points = dealer();
        winner(user_points, dealer_points);
    } while (play_again());

    ending_message();          cin.get();
    return 0;
}
```

The play_again function….

```
//Do you want to play again
bool play_again()
{
    char again;
    cout <<endl   <<"Would you like to play again? y/n ";
    cin >>again;          cin.get();
    return (again == 'y' || again == 'Y');
}
```

The deal_card changes….

```
//Deal a card - one of 14 cards, 0 is Ace, 11 is Jack, 12 Queen, 13 King
//if the argument is a -1, we are dealing to the player, otherwise dealer
int deal_card(int dealer)
{
   int card = 0;     //Find the numeric value of a card
   int points = 0;

   card = rand() % 14;
   if (0 == card)     //ACE!!
   {
        if (dealer == -1) //player...prompt for what to do
          do //find out the value they want to apply to the ace
          {

            cout <<"Dealt: an Ace " <<endl <<"Do you want to count it as"
                 <<" a 1 or 11?: ";
            cin >> points;     cin.get();
          } while (points != 1 && points != 11);  //it has to be correct!
```

The deal_card changes….

```
else //dealer
{
    if (dealer + 11 > 21) //will 11 put it over the top?
        points = 1;
    else
        points = 11;
    cout <<"Dealt: an Ace with " << points <<" point";
    if (points == 11)
        cout <<"s";        //points should be plural…
    cout <<endl;
}
else //same code for other cards….as before
```

Functions: Value vs. Reference

- **Call by value brings values into a function (as the initial value of formal arguments)**
  - that the function can access but not permanently change the original <u>actual</u> args
- Call by reference can bring information into the function or pass information to the rest of the program;
  - the function can access the values and can permanently change the <u>actual</u> arguments!

Functions: Value vs. Reference

- Call by value is useful for:

  - passing information to a function

  - allows us to use expressions instead of variables in a function call

  - value arguments are restrained to be modified only within the called function; they do not affect the calling function.

  - can't be used to pass information back, except through a returned value

Functions: Value vs. Reference

- Call by reference is useful for:

  - allowing functions to modify the value of an argument, permanently

  - requires that you use variables as your actual arguments since their value may be altered by the called function;

  - you can't use constants or literals in the function call!

Example of call by reference:

```
void convert (float inches, float & mils);
int main() {
            float in;  //local variable to hold # inches
            float mm;           //local variable for the result
            cout << “Enter the number of inches: “;
            cin >> in;
            convert (in, mm);           //function call
            cout << in << “ inches converts to “ << mm << “mm”;
            return 0;
    }
void convert (float inches, float & mils) {
            mils = 25.4 * inches;
    }
```

Example of call by reference:

```
void swap (int & a, int & b);
int main() {
    int i = 7, j = -3;
    cout << "i and j start off being equal to :" << i
            << " & " << j << '\n';
    swap(i,j);
    cout << "i and j end up being equal to    :" << i
            << " & " << j << '\n';
    return 0;
}
void swap(int & c,int & d) {
    int temp = d;
    d = c;
    c = temp;
}
```

What kind of args to use?

- Use a call by reference if:

  1) The function is supposed to provide information to some other part of the program. Like returning a result and returning it to the main.

  2) They are OUT or both IN and OUT arguments.

  3) In reality, use them WHENEVER you don’t want a duplicate copy of the arg...

What kind of args to use?

- Use a call by value:

  1) The argument is only to give information to the function - not get it back

  2) They are considered to only be IN parameters. And can't get information back OUT!

  3) You want to use an expression or a constant in function call.

  4) In reality, use them only if you need a complete and duplicate copy of the data