## Value-Returning Functions

- A call to a value-returning function is part of an expression.
- A value-returning function is used when there is only one result returned by a function and that result is to be used directly in an expression.
- A value-returning function returns one value, not through a parameter but by means of a `return` statement.





## Value-Returning Function Example

```
int Power (/* in */ int x,     // Base number
           /* in */ int n)    // Power to raise base to
{
   int result;   // Holds intermediate powers of x
   result = 1;
   while (n > 0)
   {
      result = result * x;
      n--;
   }
   return result;
}
```

Usage

```
   y = Power(5, 3)
```





## Another Value-Returning Function Example

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

int Min (int val1, int val2, int val3);

int main()
{
   int num1;
   int num2;
   int num3;
   int min_num;
```





## Another Value-Returning Function Example

```
  cout << "Please enter an integer value for num1: "
       << endl;
  cin >> num1;
  cout << "Please enter an integer value for num2: "
       << endl;
  cin >> num2;
  cout << "Please enter an integer value for num3: "
       << endl;
  cin >> num3;

  min_num = Min(num1, num2, num3);

  cout << "The minimum number of the three entered is "
       << min_num << endl;
}
```





## Another Value-Returning Function Example

```
int Min (int val1, int val2, int val3)
{
   int result;

   result = val1;

   if (val2 < result)
       result = val2;
   if (val3 < result)
       result = val3;

   return result;
}
```





## Revisiting the Power Function

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

float power (int base, int exponent);

int main()
{
   float result;
   int base;
   int exponent;

   cout << "Please enter integer base value " << endl;
   cin >> base;
```

## Revisiting the Power Function

```
   cout << "Please enter integer exponent value "
        << endl;
   cin >> exponent;

   result = power(base, exponent);
   cout << base << " raised to the " << exponent
        << " power is " << result << endl;
}
```





## Revisiting the Power Function

```
float power (int base, int exponent)
{
   int i =1;
   float result = 1.0;

   while (i <= abs(exponent))
   {
      result = result * base;
      i++;
   }
   if (exponent < 0)
      result = 1.0/result;
   return result;
}
```





## Problem-Solving Case Study (Starship Weight and Balance)

- Problem: Write a program that determines the weight and center of gravity of a new plane, based on the number of crew members and passengers as well as the weight of the baggage, closet contents, and fuel.
- Input: Number of crew members, number of passengers, weight of closet contents, baggage weight, fuel in gallons.
- Output: Total weight, center of gravity.





## Starship Weight and Balance (continued)

- Discussion: Sum all of the weights, using the standard average weight of a person of 170 pounds, and the fact that a gallon of fuel weighs 6.7 pounds.
  - totalWeight = emptyWeight + (crew + passengers) * 170 + baggage + closet + fuel * 6.7
  - centerofGravity = (emptyMoment + crewMoment + passengerMoment + cargoMoment + fuelMoment)/totalWeight





## Main (Level 0)

Get data

Set totalWt = EMPTY_WEIGHT + (passengers + crew) * 170 + baggage + closet + fuel * 6.7

Set centerof Gravity = (CrewMoment(crew) + PassengerMoment(passengers) + CargoMoment(closet, baggage) + FuelMoment(fuel) + EMPTY _MOMENT)/totalWt

Print totalWt, centerof Gravity

Print warning





## Level 1 Functional Decomposition

Get Data (Out: crew, passengers, closet, baggage, fuel)

Prompt for numer of crew, number of passengers, weight in closet and baggage compartments, and gallons of fuel

Read crew, passengers, closet, baggage, fuel

Echo print the input

CrewMoment (In: crew) Out Function Value

Return crew * 170 * 143

Cargo Moment (In: closet, baggage) Out: Function value

Return closet * 182 + baggage * 386

## Level 1 Functional Decomposition (continued)

Passenger Moment (In: passengers)
- Set moment = 0.0
- IF passengers > 6
  - Add (passengers – 6) * 170 * 341 to moment
  - Set passengers = 6
- IF passengers > 4
  - Add (passengers – 4) * 170 * 295 to moment
  - Set passengers = 4
- IF passengers > 2
  - Add (passengers – 2) * 170 * 219 to moment
  - Set passengers = 2
- IF passengers > 0
  - Add passengers * 170 * 265 to moment
- Return moment





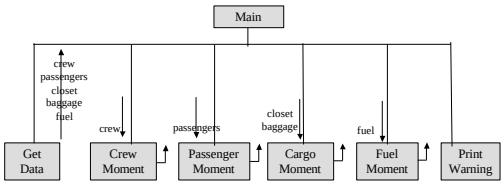
## Level 1 Functional Decomposition (concluded)

Fuel Moment (In: fuel) Out: Function value
- Set moment = 0.0
- Set fuelWt = fuel * 6.7
- IF fuel < 60
  - Set fuelDistance = fuel * 314.6
- ELSE IF fuel < 361
  - Set fuelDistance = 305.8 + (-0.01233 * (fuel – 60))
- ELSE IF fuel < 521
  - Set fuelDistance = 303.0 + (0.12500 * (fuel – 361))
- ELSE IF fuel < 521
  - Set fuelDistance = 323.0 + (-0.04444 * (fuel – 521))
- Return fuelDistance * fuelWt





## Module Structure Chart








## Case Study Implementation (abridged)

```
const float PERSON_WT = 170.0;        // Average person weighs
                                      //   170 lbs.
const float LBS_PER_GAL = 6.7;        // Jet-A weighs 6.7 lbs.
                                      //   per gal.
const float EMPTY_WEIGHT = 9887.0;    // Standard empty weight
const float EMPTY_MOMENT = 3153953.0; // Standard empty moment

float CargoMoment( int, int );
float CrewMoment( int );
float FuelMoment( int );
void  GetData( int&, int&, int&, int&, int& );
float PassengerMoment( int );
void  PrintWarning();
```





## Case Study Implementation

```
int main()
{
    int   crew;          // Number of crew on board (1 or 2)
    int   passengers;    // Number of passengers (0 through 8)
    int   closet;        // Weight in closet (160 lbs. Max.)
    int   baggage;       // Weight of baggage (525 lbs. max.)
    int   fuel;          // Gallons of fuel (10 - 565 gals.)
    float totalWt;       // Total weight of loaded Starship
    float centerOfGravity; // Center of gravity of loaded Ship
    GetData(crew, passengers, closet, baggage, fuel);
    totalWt = EMPTY_WEIGHT + float(passengers + crew) *
        PERSON_WT + float(baggage + closet) + float(fuel) *
        LBS_PER_GAL;
    centerOfGravity = (CrewMoment(crew) + PassengerMoment
        (passengers) + CargoMoment(closet, baggage) +
        FuelMoment(fuel) + EMPTY_MOMENT) / totalWt;
```





## Case Study Implementation

```
void GetData( /*out*/ int& crew,
              /*out*/ int& passengers,
              /*out*/ int& closet,
              /*out*/ int& baggage,
             /*out*/ int& fuel      )
{
    cout << "Enter the number of crew members." << endl;
    cin >> crew;
    cout << "Enter the number of passengers." << endl;
    cin >> passengers;
    cout << "Enter the closet cargo weight, in pounds."
         << endl;
    cin >> closet;
    cout << "Enter the weight, in pounds, of cargo in
             the aft baggage compartment." << endl;
    cin >> baggage;
    cout << "Enter the number of U.S. gallons of fuel"
         << endl << " loaded.";
    cin >> fuel;
}
```

## Case Study Implementation

```
float CrewMoment( /*in*/ int crew )  // Number of crew members

// Computes the crew moment arm in inch-pounds from the number of
// crew members.  Global constant PERSON_WT is used as the weight
// of each crew member
// Precondition:
//    crew == 1  OR  crew == 2
// Postcondition:
//    Function value == Crew moment arm, based on the crew
//    parameter

{
    const float CREW_DISTANCE = 143.0;  // Distance to crew seats
                                        //   from front

    return float(crew) * PERSON_WT * CREW_DISTANCE;
}
```





## Case Study Implementation

```
float PassengerMoment( /*in*/ int passengers )
{
    const float ROW1_DIST = 219.0;  // Distance to row 1 seats
    const float ROW2_DIST = 265.0;  // Distance to row 2 seats
    const float ROW3_DIST = 295.0;  // Distance to row 3 seats
    const float ROW4_DIST = 341.0;  // Distance to row 4 seats
    float moment = 0.0;             // Running total of moment
    if (passengers > 6)             // For passengers 7 and 8
    {
        moment = moment + float (passengers - 6) *
                 PERSON_WT * ROW4_DIST;
        passengers = 6;                     // 6 remain
    }
    if (passengers > 4)              // For passengers 5 and 6
        ⋮
    return moment;
}
```





## Case Study Implementation

```
float CargoMoment( /*in*/ int closet,     // Weight in closet
                   /*in*/ int baggage )   // Weight of baggage

// Computes the total moment arm for cargo loaded into the
// front closet and aft baggage compartment
// Precondition:
//     0 <= closet <= 160  &&  0 <= baggage <= 525
// Postcondition:
//     Function value == Cargo moment arm, based on the closet
//     and baggage parameters

{
    const float CLOSET_DIST = 182.0;    // Distance to closet
    const float BAGGAGE_DIST = 386.0;   // Distance to baggage
    return float(closet) * CLOSET_DIST +
           float(baggage) * BAGGAGE_DIST;
}
```





## Case Study Implementation

```
float FuelMoment( /* in */ int fuel )    // Fuel in gallons
{
    float fuelWt;           // Weight of fuel in pounds
    float fuelDistance;     // Distance from front of plane

    fuelWt = float(fuel) * LBS_PER_GAL;
    if (fuel < 60)
        fuelDistance = float(fuel) * 314.6;
    else if (fuel < 361)
        fuelDistance = 305.8 + (-0.01233 * float(fuel - 60));
    else if (fuel < 521)
        fuelDistance = 303.0 + ( 0.12500 * float(fuel - 361));
    else
        fuelDistance = 323.0 + (-0.04444 * float(fuel - 521));
    return fuelDistance * fuelWt;
}
```





## Testing & Debugging – Stubs

- A stub is has the same name and interface as a function that actually would be called by the part of the program being tested, but it is usually much simpler
- Example

```
void GetYear (/*inout*/ ifstream& dataIn,   // Input file
              /*out*/ string& year)  //Four digits of year

// Stub for GetYear function in the ConvertDates program

{
  cout << "Get Year was called here. Returning \"1949\"."
       << endl;
  year = "1948";
}
```





## Testing & Debugging - Drivers

- A driver is a simple main function which cals a function being tested and permits direct control of testing.
- Example

```
float FuelMoment (int);
int main()
{
  int testVal;
  cout << "Fuel moment for gallons from 10 – 565 in steps"
       << "of 15:" << endl;
  testVal = 10;
  while (testVal <= 565)
  {
     cout << FuelMoment(testVal) << endl;
     testVal = testVal + 15;
  }
}
```