Using a Function

- Define the function before it is called.

- However, many programmers prefer to see main() earlier to have a global view.
  - Declare the function using a statement called a function prototype.

```
void print_stars()
{
   cout << "*********"
        << endl;
}

int main()
{
   print_stars();
   cout << "Test" << endl;
   print_stars();
}
```

Function Prototypes (P.256)

- It contains the same information as appears in the function header, with the addition of a semicolon (;).
  - `double power(double value, int index);`
  - `void print_stars();`
- You can even omit the names of the parameters
  - `double power(double, int);`
  - However, it is better to use meaningful name in a prototype to increase readability.

Example of Using Function Prototypes

```
void print_stars();

int main()
{
   print_stars();
   cout << "Test" << endl;
   print_stars();
}

void print_stars()
{
   cout << "*********"
        << endl;
}
```

Using a Function

- Ex5_01.cpp on P.257

- Note the 3 ways to call this function:
  - Passing constants as arguments
    - power(5.0, 3)
  - Passing expressions (variables) as arguments
    - power(3.0, index)
  - Using a function as an argument
    - power(x, power(2.0, 2.0))

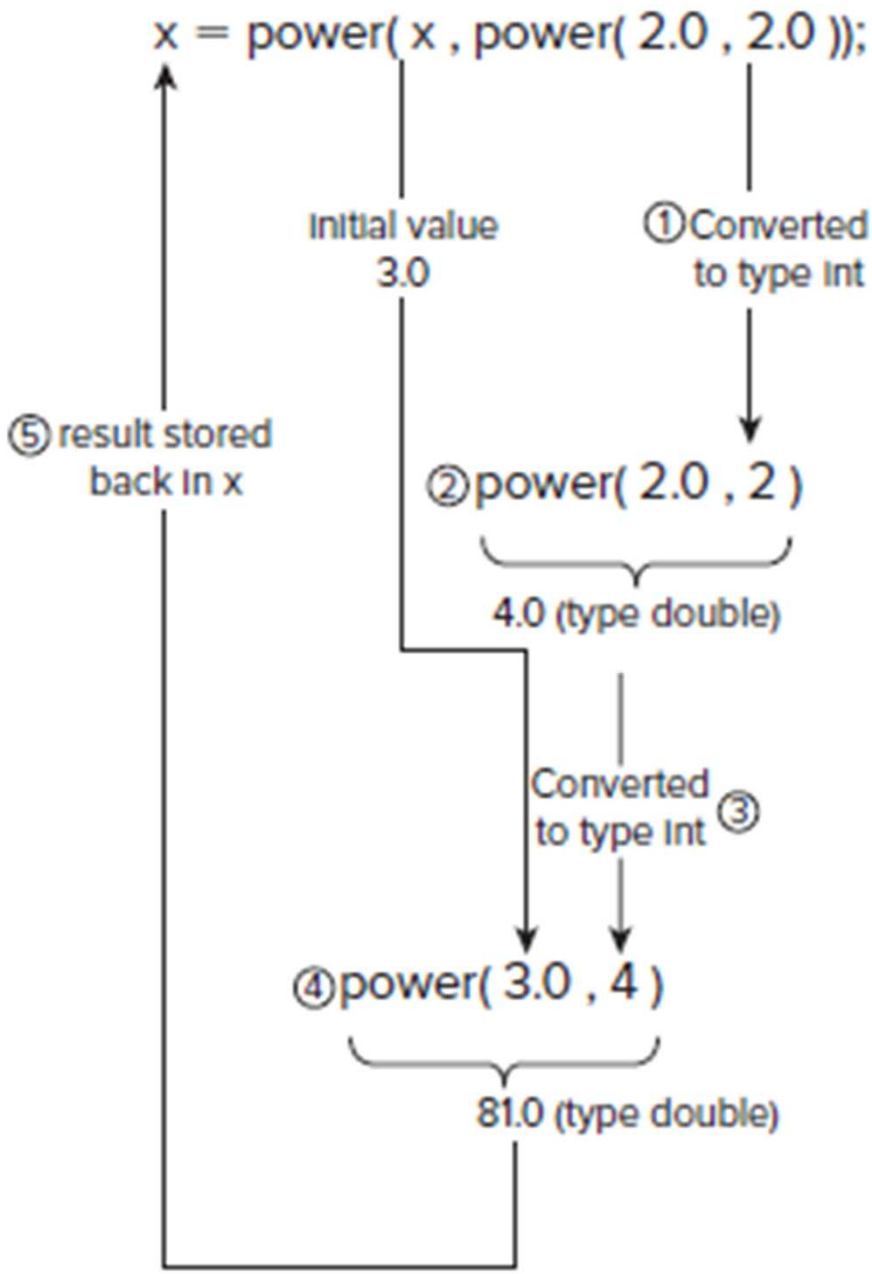
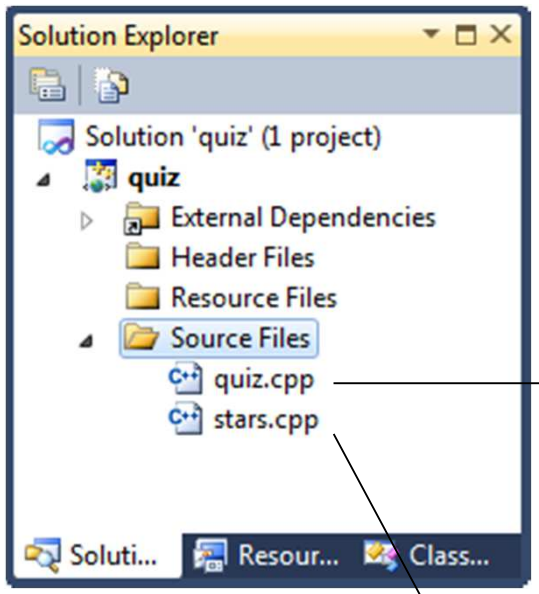

x = power(x , power(2.0 , 2.0));
Initial value
3.0
①Converted
to type int
⑤ result stored
back in x
②power(2.0 , 2)
4.0 (type double)
Converted
to type int ③
④power(3.0 , 4)
81.0 (type double)

FIGURE 5-2

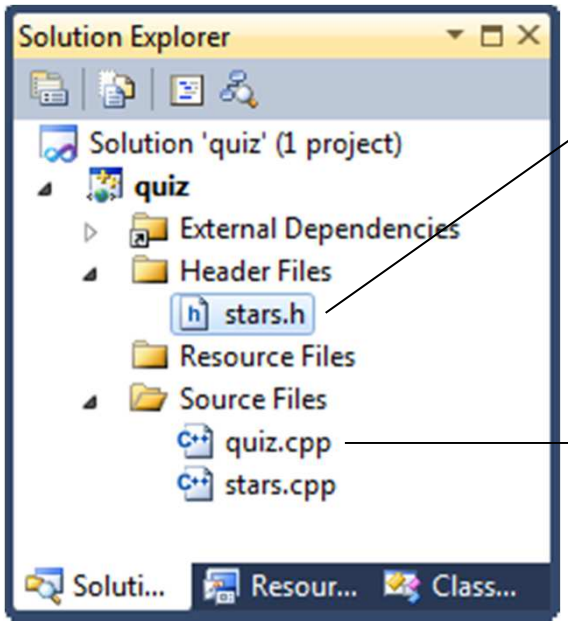
You May Separate Your Code into Several Files




```
#include <iostream>
using std::cout;

void print_stars(); // function prototype

int main()
{
        cout << "Good morning.\n";
        print_stars();
        cout << "Good afternoon.\n";
}
```

```
#include <iostream>

void print_stars()
{
        std::cout << "******\n";
}
```

Define function prototypes in a header file




```
void print_stars();
```

```
#include <iostream>
using std::cout;

#include "stars.h"

int main()
{
    cout << "Good morning.\n";
    print_stars();
    cout << "Good afternoon.\n";
}
```

Passing Arguments to a Function

- There are two mechanisms in C++ to pass arguments to functions
  - Pass-by-value
  - Pass-by-reference

Pass-by-value

```
int index = 2;
double value = 10.0;
double result = power(value, index);
```

- Copied of arguments are stored in a temporary location in memory.
- This mechanism protect your caller arguments from being accidentally modified by a rogue function.

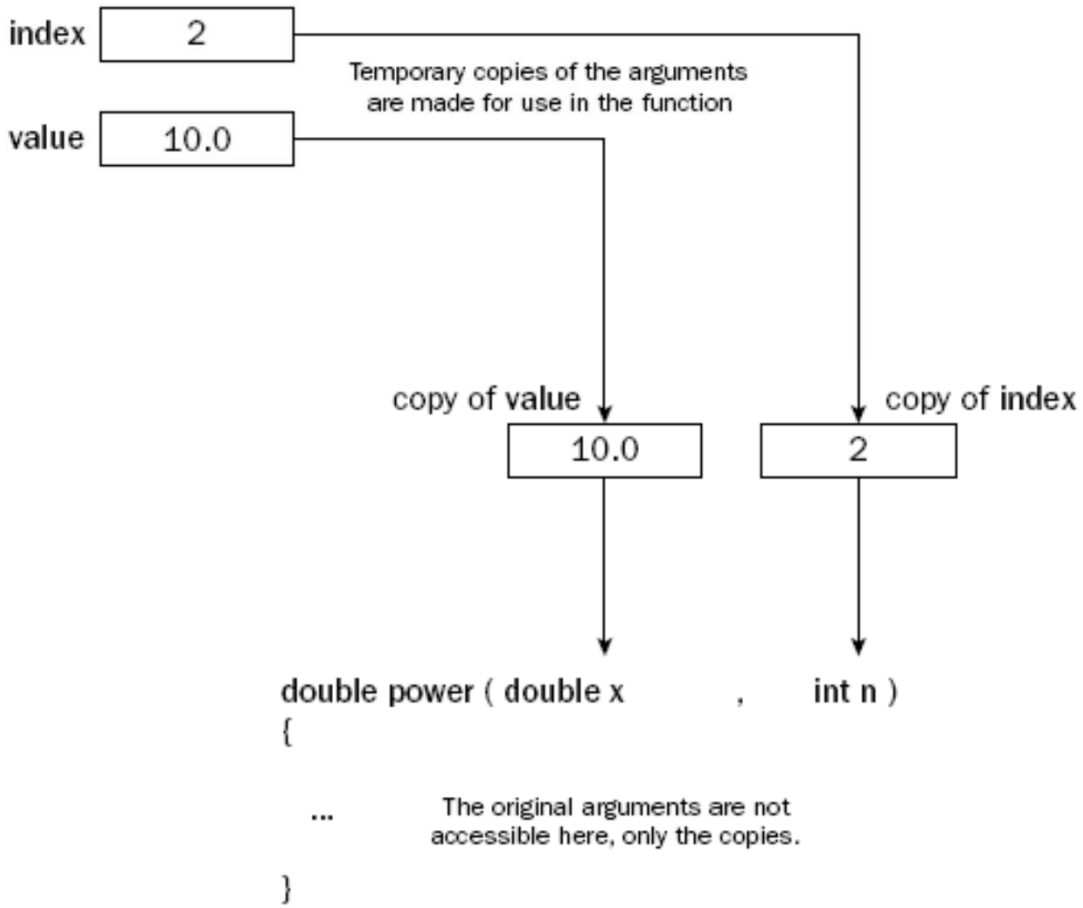


Figure 5-3

Ex5_02.cpp on P.261

```
int main(void)
{  int num = 3;
   cout << endl
        << "incr10(num) = " << incr10(num)
        << endl << "num = " << num;
   cout << endl;
   return 0;
}

int incr10(int num)
{ num += 10;
   cout << "In the function, num = "
        << num << endl;
   return num;
}
```

Example: Playing Cards

- Playing cards have four suits:
  - Spade (♠)
  - Heart (♥)
  - Diamond (♦)
  - Club (♣)
- There are 13 ranks in a suit:
  - 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A

Use 0..51 to Represent 52 Cards

```
   2  3  4  5  6  7  8  9  T  J  Q  K  A
 ================================================
♣  0  1  2  3  4  5  6  7  8  9 10 11 12
♦ 13 14 15 16 17 18 19 20 21 22 23 24 25
♥ 26 27 28 29 30 31 32 33 34 35 36 37 38
♠ 39 40 41 42 43 44 45 46 47 48 49 50 51
```

ex5_cards.cpp

Suit	♥	♦	♣	♠
ASCII code	3	4	5	6

Randomly Get 5 Cards

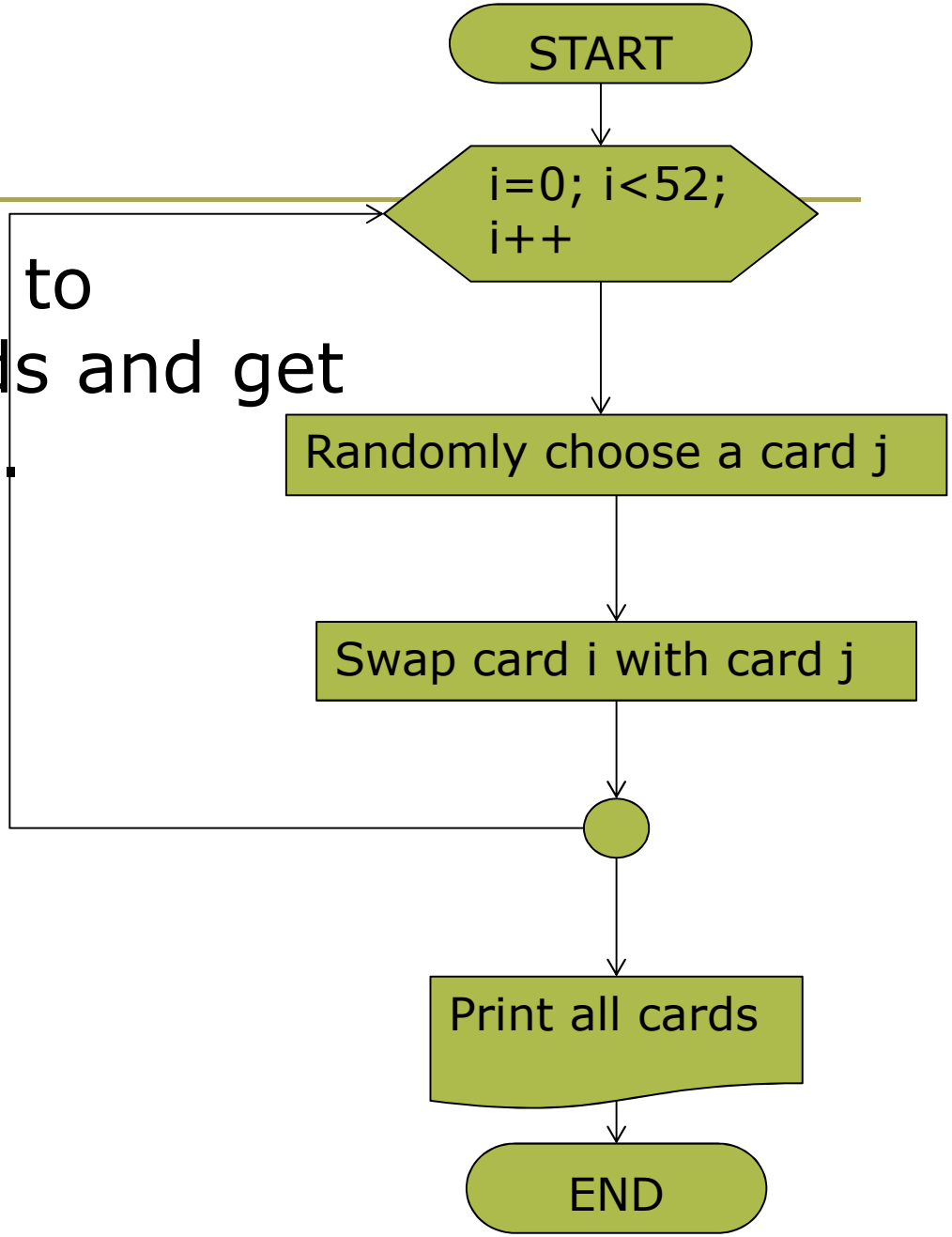
- An intuitive method is to randomly choose a number between 0..51, and repeat this action for 5 times

```
for (int i = 0; i < 5; i++ )
    cout << rand() % 52 << endl;
```

- print_card.cpp
- print_card_with_seed.cpp
  - Sooner or later, you will see duplicate cards!

Shuffle (1)

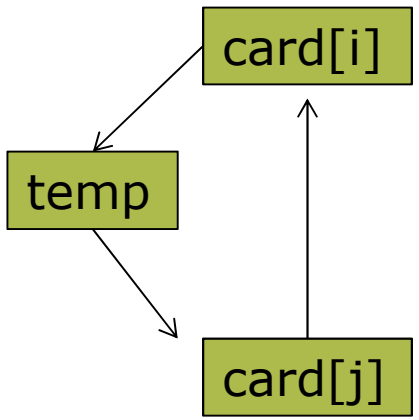
- The correct way is to "shuffle" your cards and get the first 5 of them.



Shuffle (2)

- The correct way is to "shuffle" your cards and get the first 5 of them.

```
for (i=0; i<52; i++)
{
    j = rand() % 52;
     temp=card[i];
    card[i]=card[j];
    card[j]=temp;
}
```




```
1 2 3 4 5 6 7 8 9
^           +
7 2 3 4 5 6 1 8 9
  @
7 2 3 4 5 6 1 8 9
    ^ +
7 2 4 3 5 6 1 8 9
      @
7 2 4 3 5 6 1 8 9
        @
7 2 4 3 5 6 1 8 9
        + ^
7 2 4 3 6 5 1 8 9
  +           ^
7 1 4 3 6 5 2 8 9
              ^ +
7 1 4 3 6 5 2 9 8
              + ^
7 1 4 3 6 5 2 8 9
```

- Let's see a shorter example: rand_swap.cpp

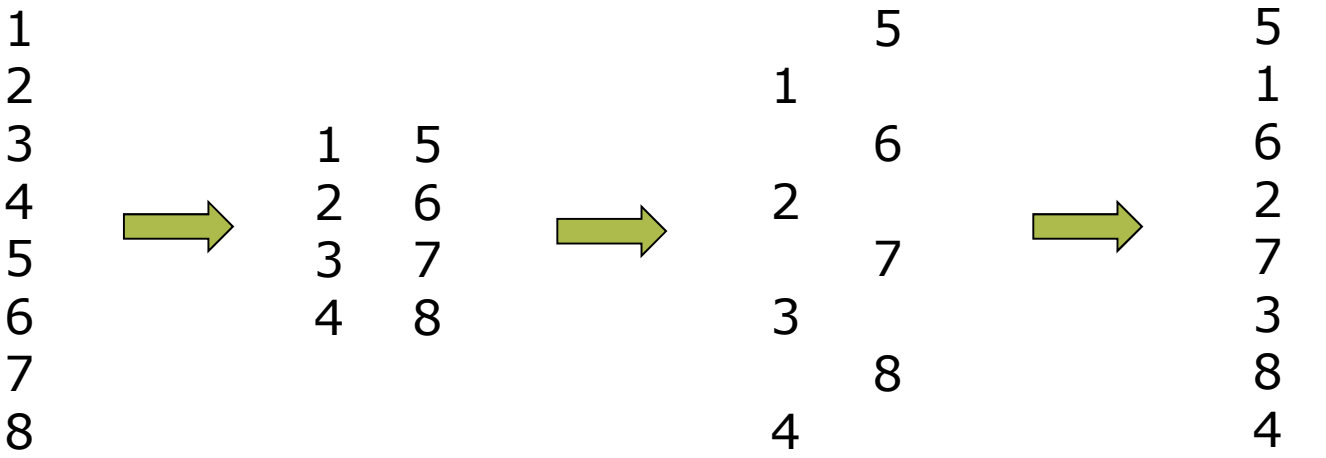
card_shuffle.cpp

```
int main()
{
    int cards[52];
    initialize(cards, 52);
    shuffle(cards, 52);
    get_five(cards);
    return 0;
}
```

Let’s inspect the details after we learn how to pass arrays to a function.

Homework: Riffle Shuffle

- See these professional ways to shuffle a deck of cards:
  - The Riffle Shuffle for Playing Card Shuffling
  - The Riffle Shuffle
  - Slow Motion of Riffle Shuffle



riffle_shuffle()

- Implement the Riffle Shuffle as a function, whose input is an array (length may be shorter than 52) of integers. The function will shuffle the sequence of array elements according to Riffle Shuffle.
- Write a simple `main()` function to call the function, and display a deck of cards before and after the shuffle.