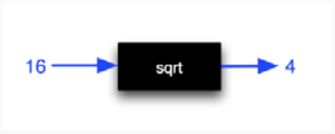
6. Functions I

Introduction to Using Functions (1)

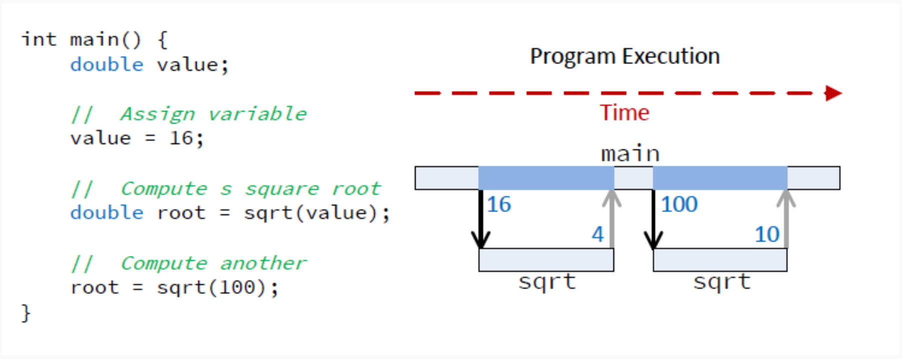
```
#include <iostream>
int main() {
   double input;
   std::cout << "Enter number: ";
   std::cin >> input;
   double diff;
   double root = 1.0;
   do {
      root = (root + input/root) / 2.0;
      std::cout << "root is " << root << '\n';
      diff = root * root - input;
   }
   while (diff > 0.0001 || diff < -0.0001);

   std::cout << "Square root of " << input << " = " << root << '\n';
}
```

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{A}{x_k}\right),\ x_0 = 1$$



Introduction to Using Functions (2)

```
#include <iostream>
#include <cmath>
int main() {
   double input;
   std::cout << "Enter number: ";
   std::cin >> input;
   double root = sqrt(input);
   std::cout << "Square root of " << input << " = " << root << '\n';
}
```



```
int main() {
    double value;

    //  Assign variable
    value = 16;

    //  Compute s square root
    double root = sqrt(value);

    //  Compute another
    root = sqrt(100);
}
```

Introduction to Using Functions (3)

```
std::cout << sqrt(16.0) << '\n';

std::cout << sqrt(x) << '\n';

std::cout << sqrt(2 * x - 5) << '\n';

double y = sqrt(x);

y = 2 * sqrt(x + 16) - 4;

y = sqrt(sqrt(256.0));
```

Introduction to Using Functions (4)

```
// Function name, parameter list, return type
// double sqrt(double)
// double sqrt(double x)

// Prototype declaration
double sqrt(double);
double sqrt(double x);

std::cout << ~~sqrt("16") ~~<< '\n';
std::cout << ~~sqrt(4.0, 7.0) ~~<< '\n';


#include <algorithm>

std::cout << "The larger of " << 4 << " and " << 7
   << " is " << max(4, 7) << '\n';
```

Introduction to Using Functions (5)

```
int rand(); // generating pseudorandom number
void exit(int); // terminating the program's execution

~~std::cout << exit(8) << '\n';~~
```

Standard Math Functions (1)

```
// <cmath>
double sqrt(double x);
double exp(double x);  // Natural Exponential
double log(double x);  // Natural logarithm
double log10(double x);
double sin(double x);  // x: radian
double cos(double x);
double tan(double x);
double pow(double x, double y);
double fabs(double x);


double asin(double x);
double acos(double x);
double atan(double x);
double atan2(double y, double x);
```

Standard Math Functions (2)

```
double sqrt(double);
float sqrt(float);
long double sqrt(long double);

// overloading or template
```

Maximum and Minimum

```
#include <iostream>
#include <algorithm>
int main() {
   int value1, value2;
   std::cout << "Please enter two integer values: ";
   std::cin >> value1 >> value2;
   std::cout << "max = " << std::max(value1, value2)
      << ", min = " << std::min(value1, value2) << '\n';
}
```

Clock Function

```
#include <iostream>
#include <ctime>
int main() {
   char letter;
   std::cout << "Enter a character: ";
   clock_t seconds = clock(); // Record starting time
   std::cin >> letter;
   clock_t other = clock();   // Record ending time
   std::cout << static_cast<double>(other - seconds)/CLOCKS_PER_SEC
      << " seconds\n";
}


// typedef long clock_t;
// #define CLOCKS_PER_SEC  ((clock_t)1000)
```

Character Functions

```
#include <iostream>
#include <cctype>
int main() {
   for (char lower = 'a'; lower <= 'z'; lower++) {
      char upper = toupper(lower);
      std::cout << lower << " => " << upper << '\n';
   }
}

// int toupper(int ch);
// int tolower(int ch);
// int isupper(int ch);
// int islower(int ch);
// int isalpha(int ch);
// int isdigit(int ch);
```

Random Numbers

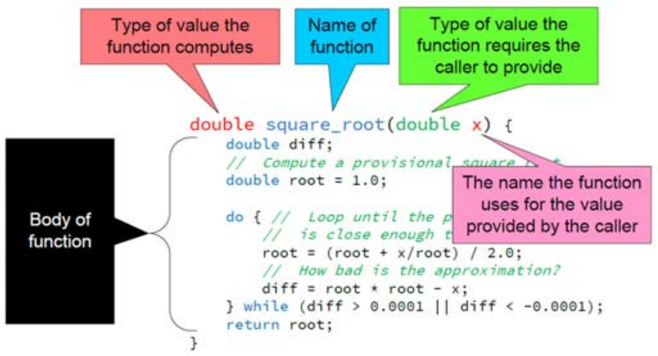
```
// void srand(unsigned)
// int rand()

#include <iostream>
#include <cstdlib>
int main() {
   srand(23);
   for (int i = 0; i < 100; i++) {
      int r = rand();
      std::cout << r << " ";
   }
   std::cout << '\n';
}


// srand(static_cast<unsigned>(time(0)));
// #define RAND_MAX 0x7fff
// int r = rand() % 100;
```

Function Basics (1)

```
#include <iostream>
#include <iomanip>
#include <cmath>
double square_root(double x) {
   double diff;
   double root = 1.0;
   do {
      root = (root + x/root) / 2.0;
      //std::cout << "root is " << root << '\n';
      diff = root * root - x;
   } while (diff > 0.0001 || diff < -0.0001);
   return root;
}
int main() {
   for (double d = 1.0; d <= 10.0; d += 0.5)
   std::cout << std::setw(7) << square_root(d) << " : " << sqrt(d)
      << '\n';
}
```

Function Basics (2)

- Function definition
  - The definition of a function specifies the function's return type and parameter types, and it provides the code that determines the function's behavior.
- Function invocation (by call)
  - A programmer uses a function via a function invocation. The main function invokes both our `square_root` function and the `sqrt` function. Every function has exactly one definition but may have many invocations.
- Function name
- Return type
- Parameter list
- Body


Type of value the function computes
Name of function
Type of value the function requires the caller to provide
The name the function uses for the value provided by the caller
Body of function

```
double square_root(double x) {
    double diff;
    //  Compute a provisional square
    double root = 1.0;

    do { //  Loop until the p
        //  is close enough t
        root = (root + x/root) / 2.0;
        //  How bad is the approximation?
        diff = root * root - x;
    } while (diff > 0.0001 || diff < -0.0001);
    return root;
}
```

Function Basics (3)

```
#include <iostream>
// Definition of the prompt function
void prompt() {
   std::cout << "Please enter an integer value: ";
}

int main() {
   int value1, value2, sum;
   std::cout << "This program adds together two integers.\n";
   prompt(); // Call the function
   std::cin >> value1;
   prompt(); // Call the function again
   std::cin >> value2;
   sum = value1 + value2;
   std::cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

Function Basics (4)

- 1. Program's execution → "main" function → the first line (declaration)
- 2. std::cout
- 3. Call "prompt" function → std::cout (prompt)
- 4. "prompt" is finished, return to "main" function
- 5. std::cin >> value1;
- 6. Call "prompt" function → std::cout (prompt)
- 7. "prompt" is finished, return to "main" function
- 8. std::cin >> value2;
- 9. sum = value1 + value2;
- 10. std::cout
- 11. Program's execution terminates.

Function Basics (5)

```
#include <iostream>
// Definition of the prompt function
int prompt() {
   int result;
   std::cout << "Please enter an integer value: ";
   std::cin >> result;
   return result;
}
int main() {
   int value1, value2, sum;
   std::cout << "This program adds together two integers.\n";
   value1 = prompt();
   value2 = prompt();
   sum = value1 + value2;
   std::cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

Function Basics (6)

```
#include <iostream>
int prompt(int n) {
   int result;
   std::cout << "Please enter integer #" << n << ": ";
   std::cin >> result;
   return result;
}
int main() {
   int value1, value2, sum;
   std::cout << "This program adds together two integers.\n";
   value1 = prompt(1); // Call the function
   value2 = prompt(2); // Call the function again
   sum = value1 + value2;
   std::cout << value1 << " + " << value2 << " = " << sum << '\n';
}
```

Using Functions

```
int gcd(int num1, int num2) {
   int min = (num1 < num2) ? num1 : num2;
   int largestFactor = 1;

   for (int i = 2; i <= min; i++)
      if (num1 % i == 0 && num2 % i == 0)
         largestFactor = i; // Found larger factor

   return largestFactor;
}

// std::cout << gcd(36, 24);
// x = gcd(x - 2, 24);
// x = gcd(x - 2, gcd(10, 8));

// Greatest common divisor
// Euclidean algorithm
```

Pass by Value (1)

```
#include <iostream>
void increment(int x) {
   std::cout << "Beginning execution of increment, x = "
      << x << '\n';
   x++; // Increment x
   std::cout << "Ending execution of increment, x = "
      << x << '\n';
}
int main() {
   int x = 5;
   std::cout << "Before increment, x = " << x << '\n';
   increment(x);
   std::cout << "After increment, x = " << x << '\n';
}
```

Pass by Value (2)

```
// Local variables
#include <iostream>
void increment() {
   int x = 15;
   std::cout << "x = " << x << '\n';
   x++;
   std::cout << "x = " << x << '\n';
}
int main() {
   int x = 5;
   std::cout << "x = " << x << '\n';
   increment();
   std::cout << "x = " << x << '\n';
   {
      int x = 10;
      std::cout << "x = " << x << '\n';
   }
   std::cout << "x = " << x << '\n';
}
```

Function Example

```
#include <iostream>
#include <cmath>
bool is_prime(int n) {
   bool result = true;
   double r = n, root = sqrt(r);

   for (int trial_factor = 2;
      result && trial_factor <= root; trial_factor++)
      result = (n % trial_factor != 0);

   return result;
}
```

Organizing Functions

```
#include <iostream>
int twice(int);        // Declare function named twice
int main() {
   std::cout << twice(5) << '\n';
}
int twice(int n) {     // Define function named twice
   return 2 * n;
}
--------------------------------------------------------------
#include <iostream>
int main() {
   int twice(int);     // twice function available anywhere in main
   std::cout << twice(5) << '\n';
}
int twice(int n) {     // Define function twice
   return 2 * n;
}
```

Commenting Functions

```
/*
*       distance(x1, y1, x2, y2)
*       Computes the distance between two geometric points
*       x1 is the x coordinate of the first point
*       y1 is the y coordinate of the first point
*       x2 is the x coordinate of the second point
*       y2 is the y coordinate of the second point
*       Returns the distance between (x1,y1) and (x2,y2)
*       Author: Joe Algori (joe@eng-sys.net)
*       Last modified: 2010-01-06
*       Adapted from a formula published at
*       http://en.wikipedia.org/wiki/Distance
*/
double distance(double x1, double y1, double x2, double y2) {
   …
}
```

Custom Functions vs. Standard Functions

- Standard routines are typically tuned to be very efficient; it takes a great deal of effort to make custom code efficient.
- Standard routines are well-documented; extra work is required to document custom code, and writing good documentation is hard work.