MA122 - Computer Programming and Applications

Indian Institute of Space Science and Technology

January 18, 2017

Lecture 3

1 Function 1

2 User-defined functions

3 Dealing with Data

4 int

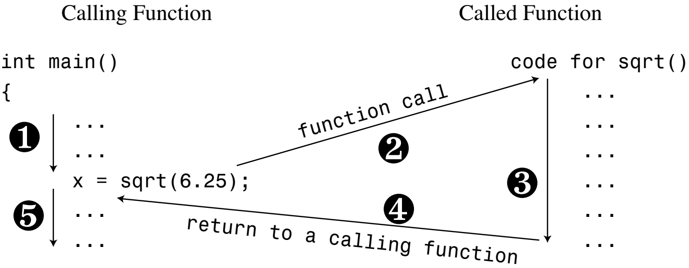
5 char

6 escape sequences

sqrt

```cpp
// sqrt.cpp -- using the sqrt() function
#include <iostream>
#include <cmath> // or math.h
int main()
{
    using namespace std;
    double area;
    cout << "Enter the floor area, in square feet, of your home: ";
    cin >> area;
    double side;
    side = sqrt(area);
    cout << " Thats  the equivalent of a square " << side
    << " feet to the side." << endl;
    cout << "How fascinating!" << endl;
    return 0;
}
```

explanation



Calling Function

Called Function

```
int main()
{
  ❶  ...
     ...
     x = sqrt(6.25);
  ❺  ...
     ...
```

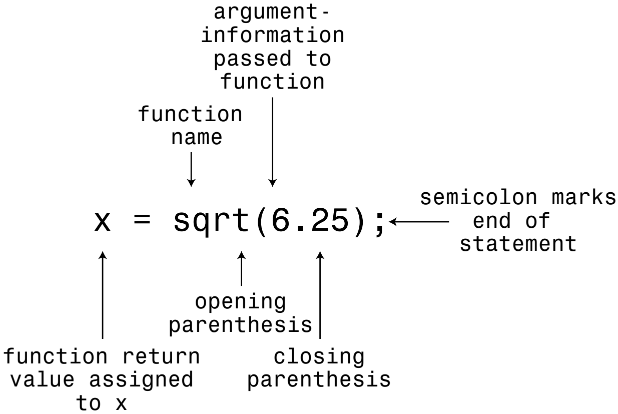
function call ❷

```
code for sqrt()
     ...
     ...
     ...
  ❸  ...
     ...
     ...
```

❹ return to a calling function

Figure 2.6 Calling a function.

more details



argument-information passed to function

function name

```
x = sqrt(6.25);
```

semicolon marks end of statement

opening parenthesis

function return value assigned to x

closing parenthesis

Figure 2.7 Function call syntax.

Lecture 3

1 Function 1

2 User-defined functions

3 Dealing with Data

4 int

5 char

6 escape sequences

my first function

```
// ourfunc.cpp -- defining your own function
#include <iostream>
void apples(int);//function prototype for apple()

int main()
{
    using namespace std;
    apples(3); //call the apple() function
    cout << "Pick an integer: ";
    int count;
    cin >> count;
    apples(count);// call it again
    cout << "Done!"<<endl;
    return 0;
}
```

my first function continued...

```cpp

void apples(int n)
{
    using namespace std;

    cout << "my first function: " << n << " apples."
        << endl;

    // void functions dont need return statements

}
```

function continued...

```
// convert.cpp -- converts stone to pounds
#include <iostream>
int stonetolb(int); // function prototype
int main()
{

    using namespace std;
    int stone;
    cout << "Enter the weight in stone: ";
    cin >> stone;
    int pounds = stonetolb(stone);
    cout << stone << " stone = ";
    cout << pounds << " pounds." << endl;
    return 0;
}
```

function continued...

```


int stonetolb(int sts)
{
        return 14 * sts;
}
```

Lecture 3

1 Function 1

2 User-defined functions

3 Dealing with Data

4 int

5 char

6 escape sequences

Integer Types

Integer Types

1. A **short** integer is at least 16 bits wide

Integer Types

1. A **short** integer is at least 16 bits wide
2. An **int** integer is at least as big as **short**.

Integer Types

1. A **short** integer is at least 16 bits wide
2. An **int** integer is at least as big as **short**.
3. A **long** integer is at least 32 bits wide and at least as big as **int**.

Integer Types

1. A **short** integer is at least 16 bits wide
2. An **int** integer is at least as big as **short**.
3. A **long** integer is at least 32 bits wide and at least as big as **int**.
4. A **long long** integer is at least 64 bits wide and at least as big as **long**.

Integer continued...

```
#include <iostream>
#include <climits> // use limits.h for older systems
int main()
{
using namespace std;
int n_int = INT_MAX; // initialize n_int to max int value
short n_short = SHRT_MAX; // symbols defined in climits file
long n_long = LONG_MAX;
long long n_llong = LLONG_MAX;

// sizeof operator yields size of type or of variable
cout << "int is " << sizeof (int) << " bytes." << endl;
cout << "short is " << sizeof n_short << " bytes." << endl;
```

Integer continued...

```
cout << "long is " << sizeof n_long << " bytes." <<
    endl;
cout << "long long is " << sizeof n_llong << " bytes."
     << endl;
cout << endl;
cout << "Maximum values:" << endl;
cout << "int: " << n_int << endl;
cout << "short: " << n_short << endl;

cout << "long: " << n_long << endl;
cout << "long long: " << n_llong << endl << endl;
cout << "Minimum int value = " << INT_MIN << endl;
cout << "Bits per byte = " << CHAR_BIT << endl;
return 0;}
```

Climits

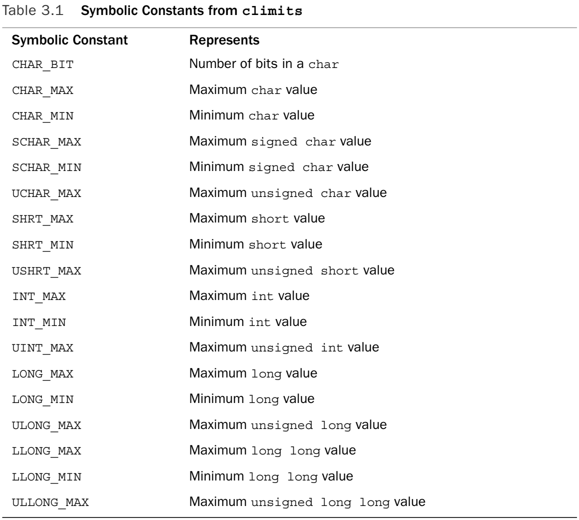


Table 3.1 **Symbolic Constants from climits**

Symbolic Constant	Represents
CHAR_BIT	Number of bits in a char
CHAR_MAX	Maximum char value
CHAR_MIN	Minimum char value
SCHAR_MAX	Maximum signed char value
SCHAR_MIN	Minimum signed char value
UCHAR_MAX	Maximum unsigned char value
SHRT_MAX	Maximum short value
SHRT_MIN	Minimum short value
USHRT_MAX	Maximum unsigned short value
INT_MAX	Maximum int value
INT_MIN	Minimum int value
UINT_MAX	Maximum unsigned int value
LONG_MAX	Maximum long value
LONG_MIN	Minimum long value
ULONG_MAX	Maximum unsigned long value
LLONG_MAX	Maximum long long value
LLONG_MIN	Minimum long long value
ULLONG_MAX	Maximum unsigned long long value

Lecture 3

1 Function 1

2 User-defined functions

3 Dealing with Data

4 int

5 char

6 escape sequences

Initialization

```
// Initialization
#include <iostream>
#include <cmath> // or math.h
int main()
{
    using namespace std;
    int uncles = 5;        // initialize uncles to 5
  int aunts = uncles;    // initialize aunts to 5
  int chairs = aunts + uncles + 4;
  // initialize chairs to 14

  int owls = 101; // traditional C initialization
  int wrens(432); // alternative C++ syntax

  int hamburgers = {24}; // C++98
  int emus{7};     //C++11
  return 0;}
```

Unsigned

```
// listing3pt2.cpp -- exceeding some integer limits
#include <iostream>
#define ZERO 0 // makes ZERO symbol for 0 value
#include <climits> // defines INT_MAX as largest int
int main()
{
  using namespace std;
  short sam = SHRT_MAX; // initialize a variable to max value

  unsigned short sue = sam;// okay if variable sam already defined

  cout << "Sam has " << sam << " dollars and Sue has " <<sue;

  cout << " dollars deposited." << endl
     << "Add $1 to each account." << endl << "Now ";
```

Unsigned

```
sam=sam+1;
sue = sue + 1;
cout << "Sam has " << sam << " dollars and Sue has "<< sue;
cout << " dollars deposited.\nPoor Sam!" << endl;
sam = ZERO;
sue = ZERO;
cout << "Sam has " << sam << " dollars and Sue has "<< sue;
cout << "Take $1 from each account." << endl << "Now ";
sam = sam - 1;
sue = sue - 1;
cout << "Sam has " << sam << " dollars and Sue has "<< sue;
cout << " dollars deposited." << endl << "Lucky Sue!" << endl;
return 0;}
```

Lecture 3

1. Function 1
2. User-defined functions
3. Dealing with Data
4. int
5. char
6. escape sequences

The char type: Characters and Small Integers

```cpp
// chartype.cpp -- the char type
#include <iostream>
int main( )
{
  using namespace std;

  char ch; // declare a char variable
  cout << "Enter a character: " << endl;

  cin >> ch;
  cout << "Hola! ";

  cout << "Thank you for the " << ch << " character."
      << endl;

  return 0;}
```

more on char

```
#include <iostream>
int main()
{
  using namespace std;
  char ch='M'; //assign ASCII code for M to ch
  int i =ch; //store same code

  cout << "The ASCII code for " << ch << " is " << i << endl;
  cout << "Add one to the character code:" <<endl;

  ch=ch+1;
  i=ch;
  cout << "The ASCII code for " << ch << " is " << i << endl;

  return 0; }
```

Lecture 3

1. Function 1
2. User-defined functions
3. Dealing with Data
4. int
5. char
6. escape sequences

escape sequences

Table 3.2 **C++ Escape Sequence Codes**

Character Name	**ASCII Symbol**	**C++ Code**	**ASCII Decimal Code**	**ASCII Hex Code**
Newline	NL (LF)	`\n`	10	0xA
Horizontal tab	HT	`\t`	9	0x9
Vertical tab	VT	`\v`	11	0xB
Backspace	BS	`\b`	8	0x8
Carriage return	CR	`\r`	13	0xD
Alert	BEL	`\a`	7	0x7
Backslash	\	`\\`	92	0x5C
Question mark	?	`\?`	63	0x3F
Single quote	’	`\'`	39	0x27
Double quote	”	`\"`	34	0x22

escape sequences

```cpp
// bondini.cpp -- using escape sequences
#include <iostream>
int main()
{

  using namespace std;

  cout << "\aOperation \"HyperHype\" is now activated!\n";
  cout << "Enter your agent code:________\b\b\b\b\b\b\b\b";

  long code;
  cin >> code;

  cout << "\aYou entered " << code << "...\n";
  cout << "\aCode verified! Proceed with Plan Z3!\n";
  return 0; }
```