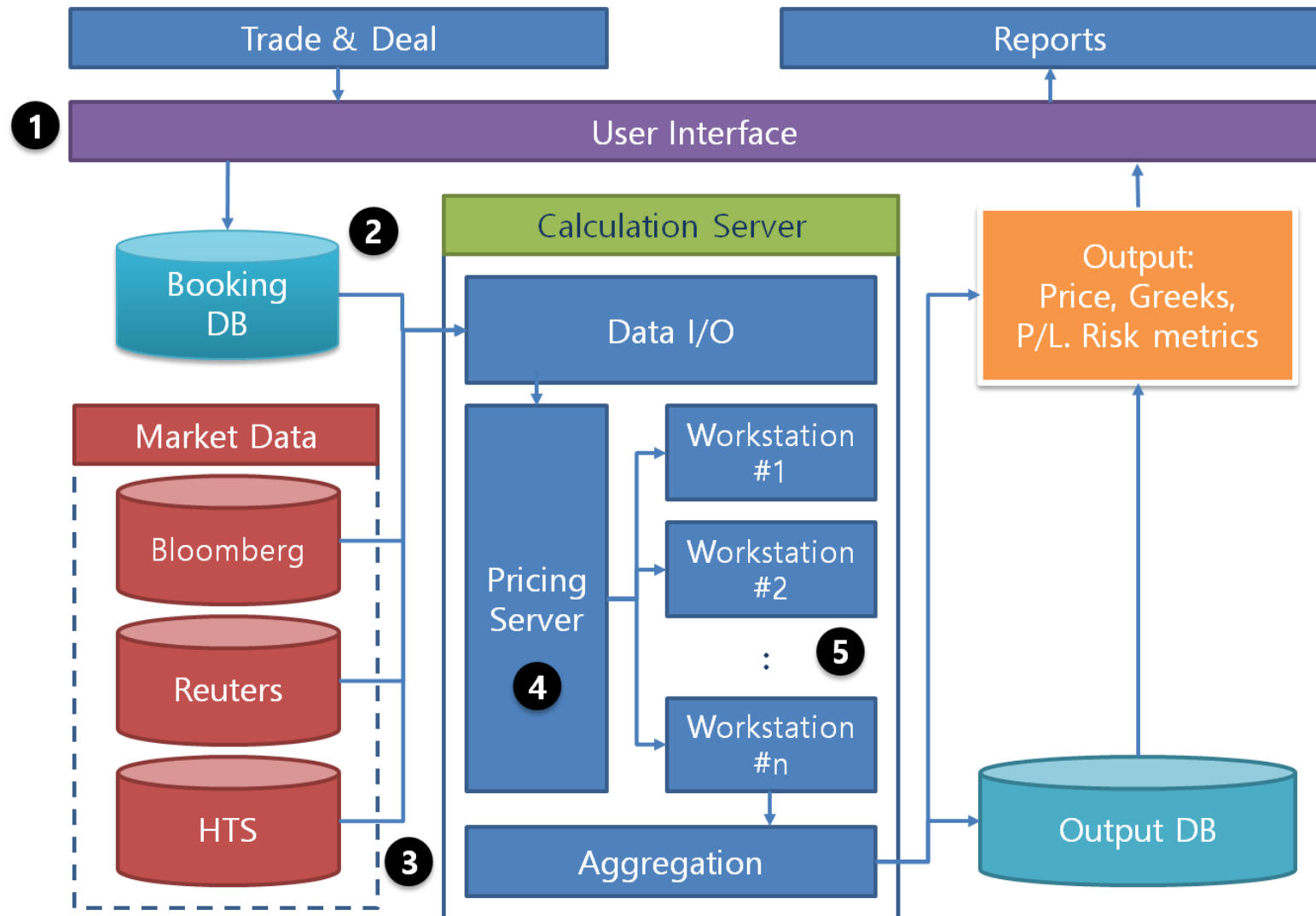


C++ 기초 문법

금융공학 프로그래밍

Programming for Financial Engineering



Getting Started with C++

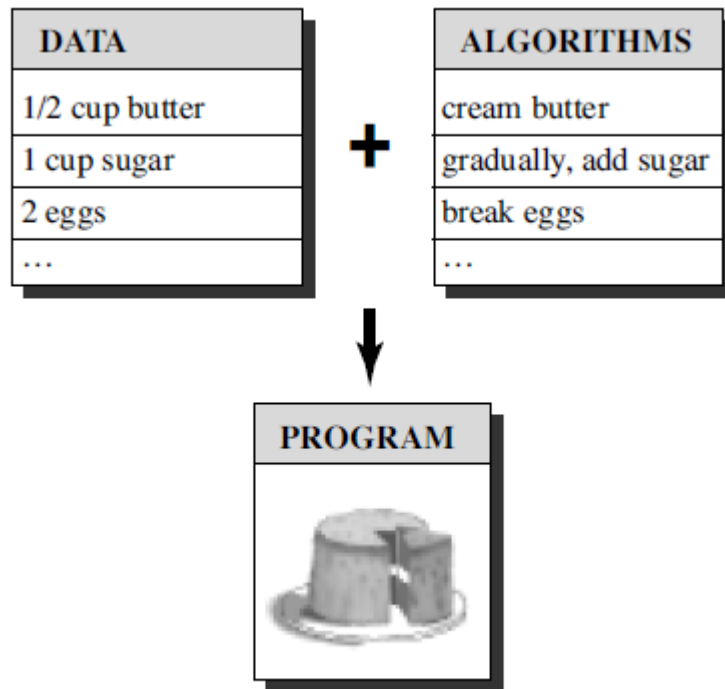


Figure 1.1 Data + algorithms = program.

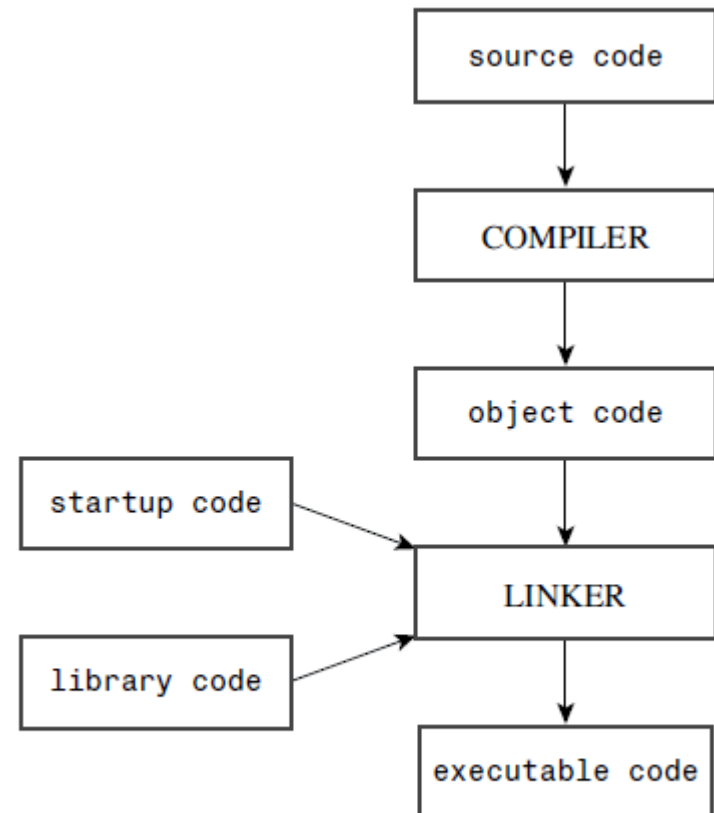


Figure 1.3 Programming steps.

Setting Out to C++

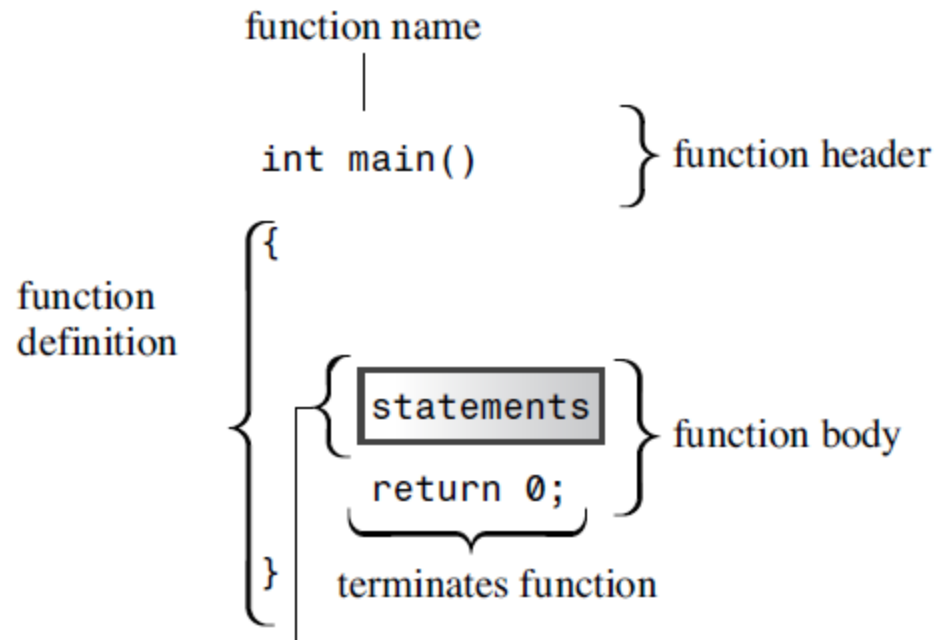
Listing 2.1 `myfirst.cpp`

```
// myfirst.cpp -- displays a message

#include <iostream>                // a PREPROCESSOR directive
int main()                        // function header
{                                // start of function body
    using namespace std;         // make definitions visible
    cout << "Come up and C++ me some time."; // message
    cout << endl;               // start a new line
    cout << "You won't regret it!" << endl; // more output
    return 0;                   // terminate main()
}                                // end of function body
```

- ❖ 함수 Function
- ❖ 주석 Comments
- ❖ 전처리기 Preprocessor
- ❖ 헤더파일 Header
- ❖ 이름공간 Namespaces
- ❖ 객체 Object
- ❖ 연산자 Operator
- ❖ 문자열 String
- ❖ 토큰 Tokens
- ❖ 화이트스페이스 White Space

Setting Out to C++



Statements are C++ expressions terminated by a semicolon.

Figure 2.1 The `main()` function.

Listing 2.2 carrots.cpp

```
// carrots.cpp -- food processing program
// uses and displays a variable

#include <iostream>

int main()
{
    using namespace std;

    int carrots;           // declare an integer variable

    carrots = 25;          // assign a value to the variable
    cout << "I have ";
    cout << carrots;       // display the value of the variable
    cout << " carrots.";
    cout << endl;
    carrots = carrots - 1; // modify the variable
    cout << "Crunch, crunch. Now I have " << carrots << " carrots." << endl;
    return 0;
}
```

❖ 선언 Declare

❖ 변수 Variable

❖ 대입 Assign

❖ 수정 Modify

입력과 출력 **cin & cout**

Listing 2.3 **getinfo.cpp**

```
// getinfo.cpp -- input and output
#include <iostream>

int main()
{
    using namespace std;

    int carrots;

    cout << "How many carrots do you have?" << endl;
    cin >> carrots;           // C++ input
    cout << "Here are two more. ";
    carrots = carrots + 2;
    // the next line concatenates output
    cout << "Now you have " << carrots << " carrots." << endl;
    return 0;
}
```

함수 Function

❖ 함수 호출

- `X = sqrt(6.25);`

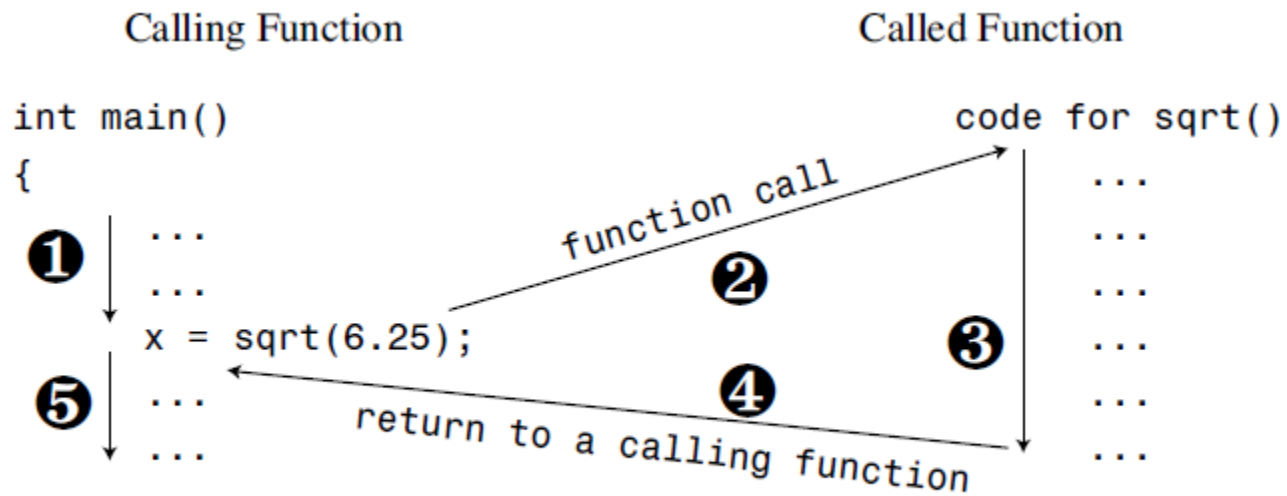


Figure 2.6 Calling a function.

Function 호출 문법

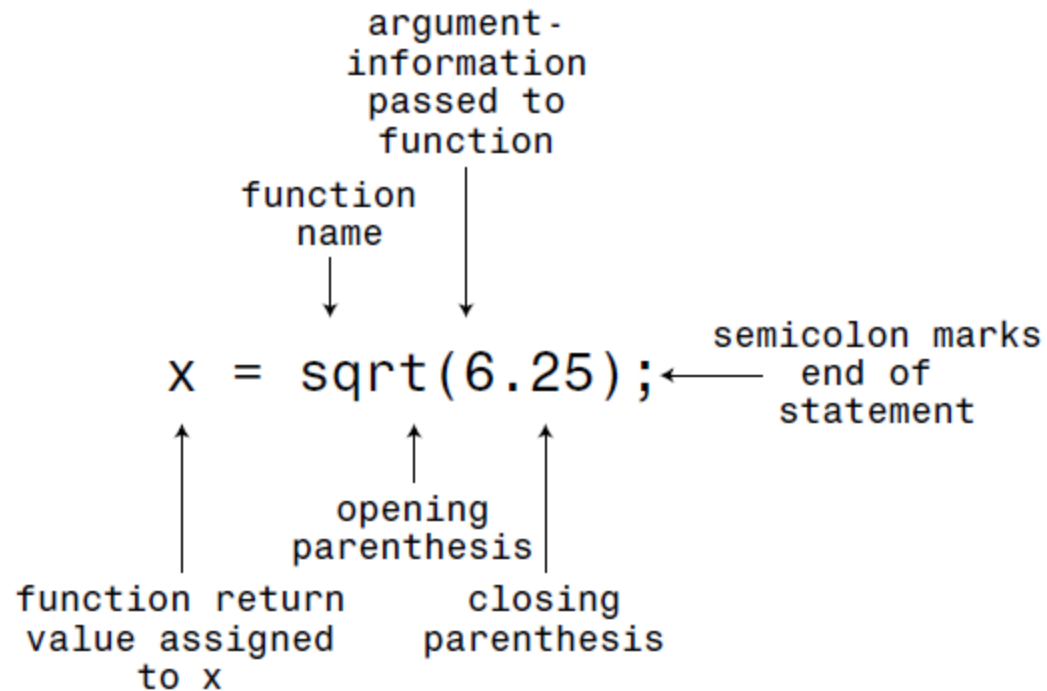


Figure 2.7 Function call syntax.

사용자 정의 함수

Listing 2.5 `ourfunc.cpp`

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

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

```
type functionname(argumentlist)
{
    statements
}
```

```
void simon(int n)    // define the simon() function
{
    using namespace std;
    cout << "Simon says touch your toes " << n << " times." << endl;
}                    // void functions don't need return statements
```

리턴 타입이 있는 함수

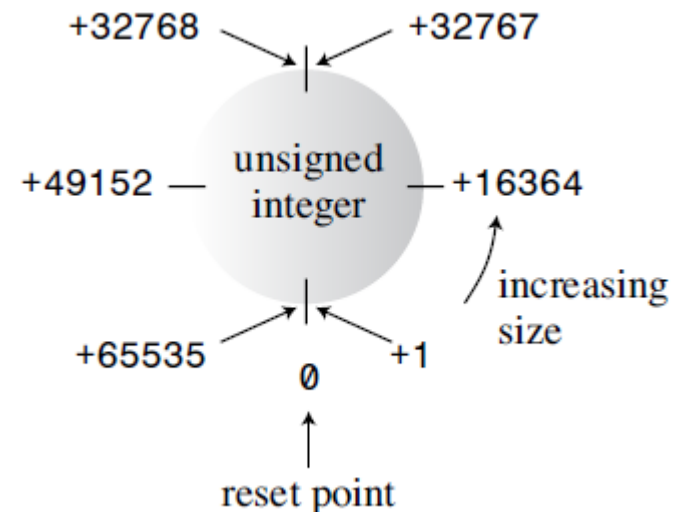
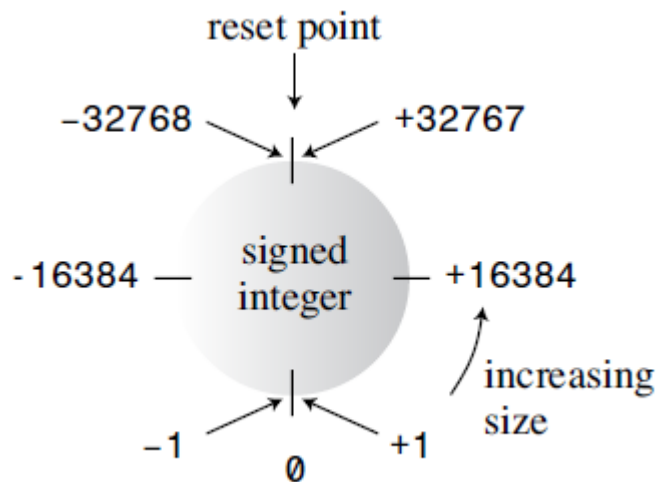
Listing 2.6 convert.cpp

```
// 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;
}

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

Integer Types

- A short integer is at least 16 bits wide.
- An int integer is at least as big as short.
- A long integer is at least 32 bits wide and at least as big as int.
- A long long integer is at least 64 bits wide and at least as big as long.



Floating Point Numbers

- ❖ float
- ❖ double
- ❖ long double

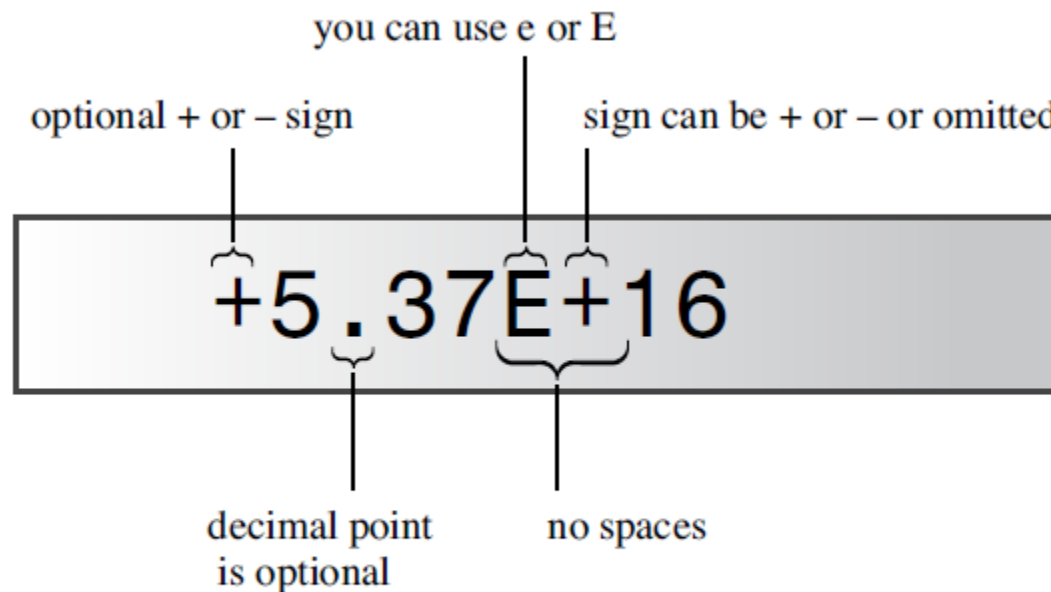
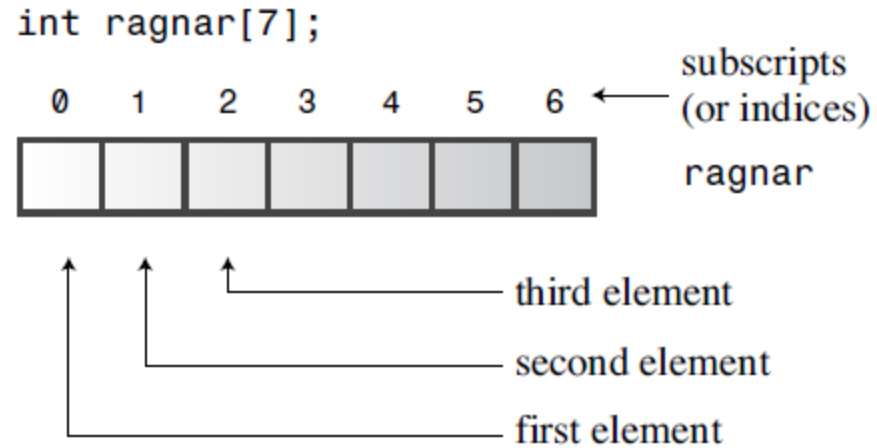


Figure 3.3 E notation.

배열



ragnar is an array holding seven values,
each of which is a type `int` variable

Figure 4.1 Creating an array.

배열 초기화

```
int cards[4] = {3, 6, 8, 10};    // okay
int hand[4];                     // okay
hand[4] = {5, 6, 7, 9};          // not allowed
hand = cards;                    // not allowed
```

```
float hotelTips[5] = {5.0, 2.5};
```

```
long totals[500] = {0};
```

```
short things[] = {1, 5, 3, 8};
```

구조체

the struct keyword the tag becomes the name for the new type

struct inflatable

opening and closing braces { char name[20]; float volume; double price; }

structure members

terminates the structure declaration

Figure 4.6 Parts of a structure description.


```

int main()
{
    using namespace std;
    inflatable guest =
    {
        "Glorious Gloria", // name value
        1.88,               // volume value
        29.99               // price value
    }; // guest is a structure variable of type inflatable
// It's initialized to the indicated values
    inflatable pal =
    {
        "Audacious Arthur",
        3.12,
        32.99
    }; // pal is a second variable of type inflatable
// NOTE: some implementations require using
// static inflatable guest =

    cout << "Expand your guest list with " << guest.name;
    cout << " and " << pal.name << "!\n";
// pal.name is the name member of the pal variable
    cout << "You can have both for $";
    cout << guest.price + pal.price << "!\n";
    return 0;
}

```

구조체 배열

```
inflatable guests[2] =           // initializing an array of structs
{
    {"Bambi", 0.5, 21.99},        // first structure in array
    {"Godzilla", 2000, 565.99}    // next structure in array
};
```

Q & A
