Array / Vector / Reference / Normal Distribution

금융공학 프로그래밍

배열

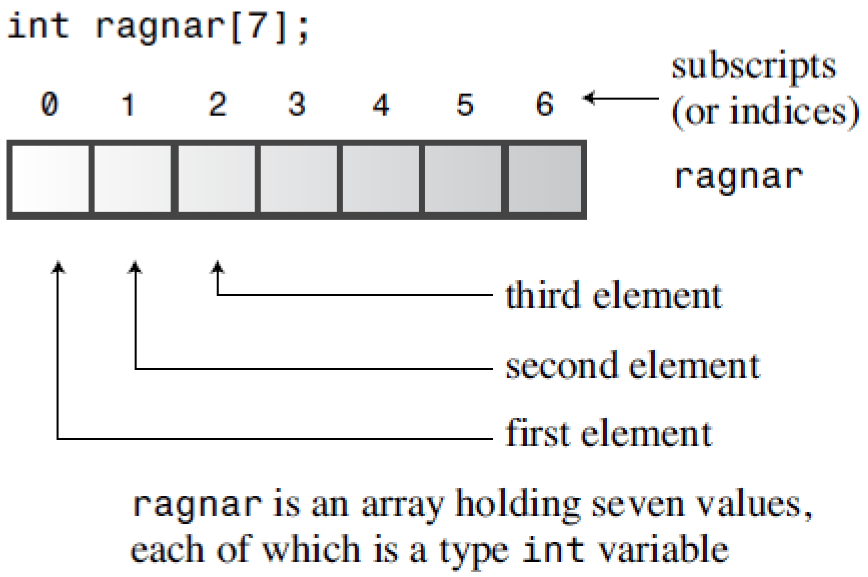


Figure 4.1 Creating an array.

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 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
    };
```

std::vector

```
#include <vector>
...
using namespace std;
vector<int> vi;          // create a zero-size array of int
int n;
cin >> n;
vector<double> vd(n);   // create an array of n doubles
```

```
vector<double> scores;   // create an empty vector
double temp;
while (cin >> temp && temp >= 0)
    scores.push_back(temp);
cout << "You entered " << scores.size() << " scores.\n";
```

std::vector

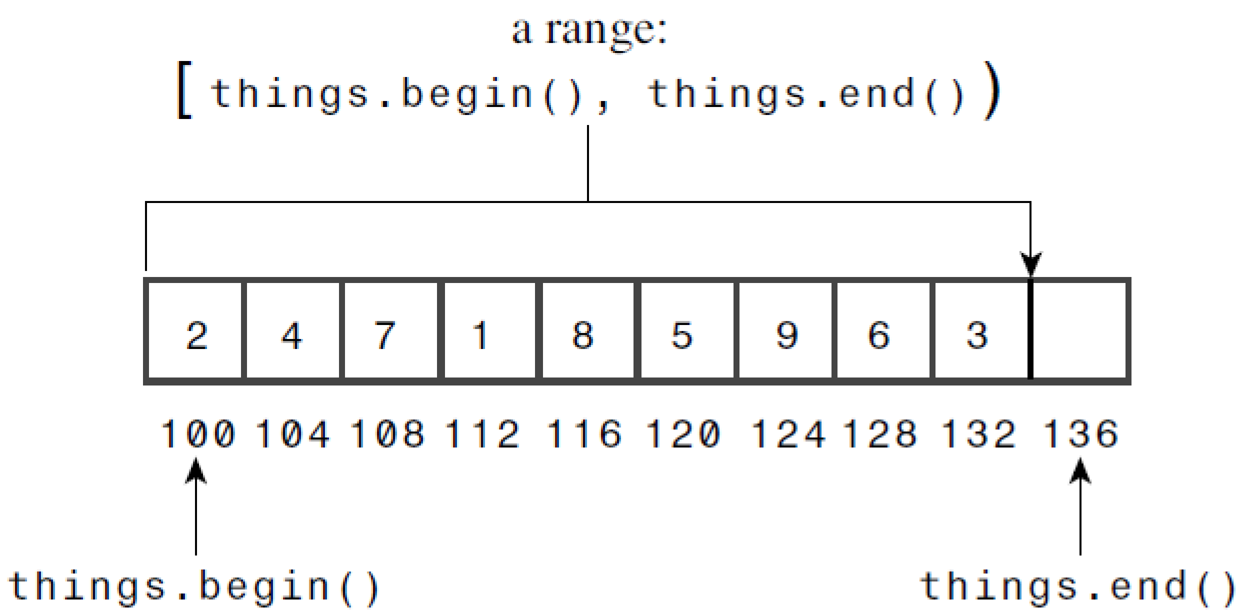


Figure 16.3 The STL range concept.

```
scores.erase(scores.begin(), scores.begin() + 2);

vector<int> old_v;
vector<int> new_v;
...
old_v.insert(old_v.begin(), new_v.begin() + 1, new_v.end());
```

Array as Arguments

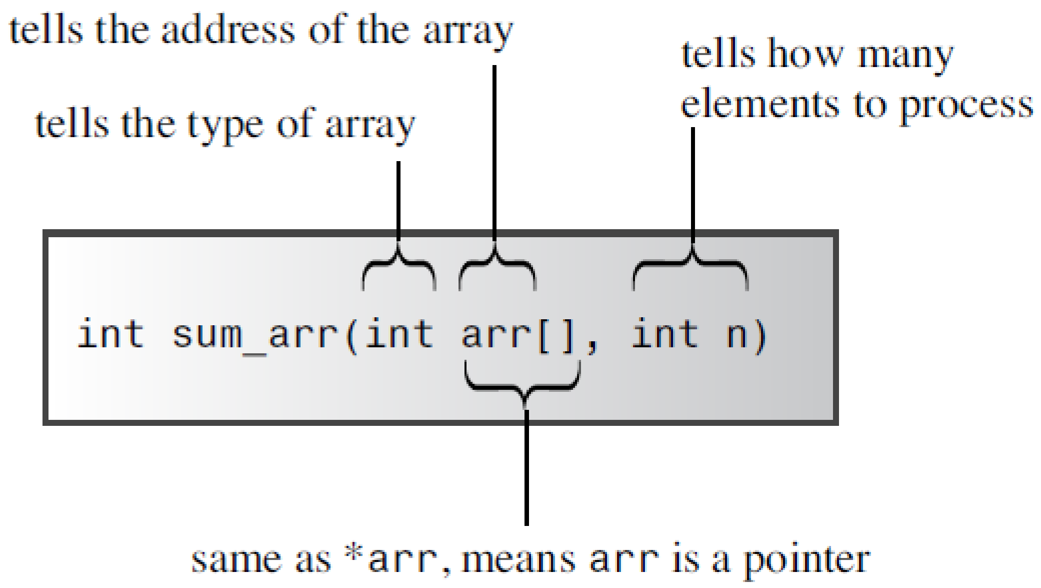

tells the address of the array
tells how many elements to process
tells the type of array
int sum_arr(int arr[], int n)
same as *arr, means arr is a pointer

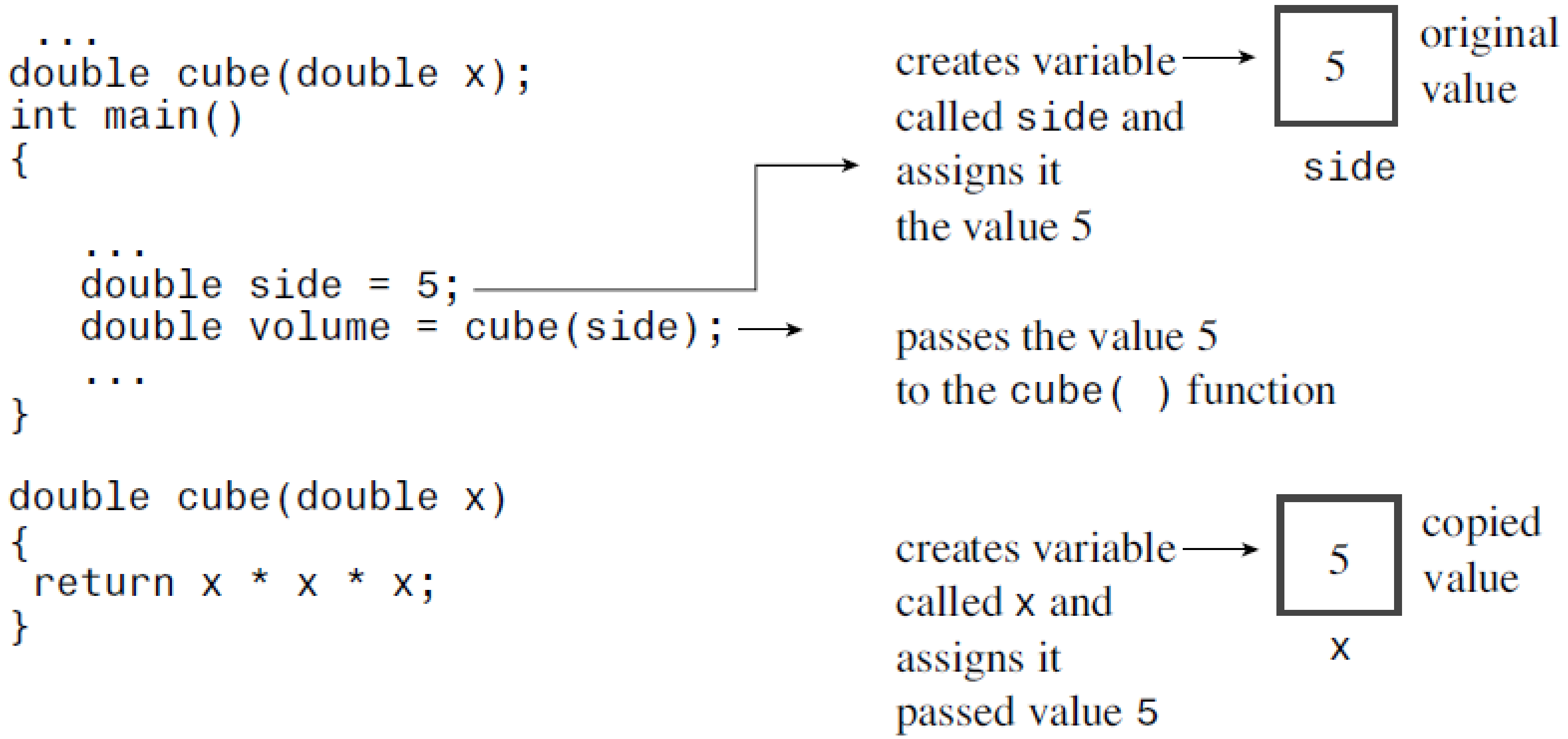
Figure 7.4 Telling a function about an array.

Reference variable

Listing 8.2 **firstref.cpp**

```
// firstref.cpp -- defining and using a reference
#include <iostream>
int main()
{
    using namespace std;
    int rats = 101;
    int & rodents = rats;   // rodents is a reference
    cout << "rats = " << rats;
    cout << ", rodents = " << rodents << endl;
    rodents++;
    cout << "rats = " << rats;
    cout << ", rodents = " << rodents << endl;

// some implementations require type casting the following
// addresses to type unsigned
    cout << "rats address = " << &rats;
    cout << ", rodents address = " << &rodents << endl;
    return 0;
}
```

Passing by value

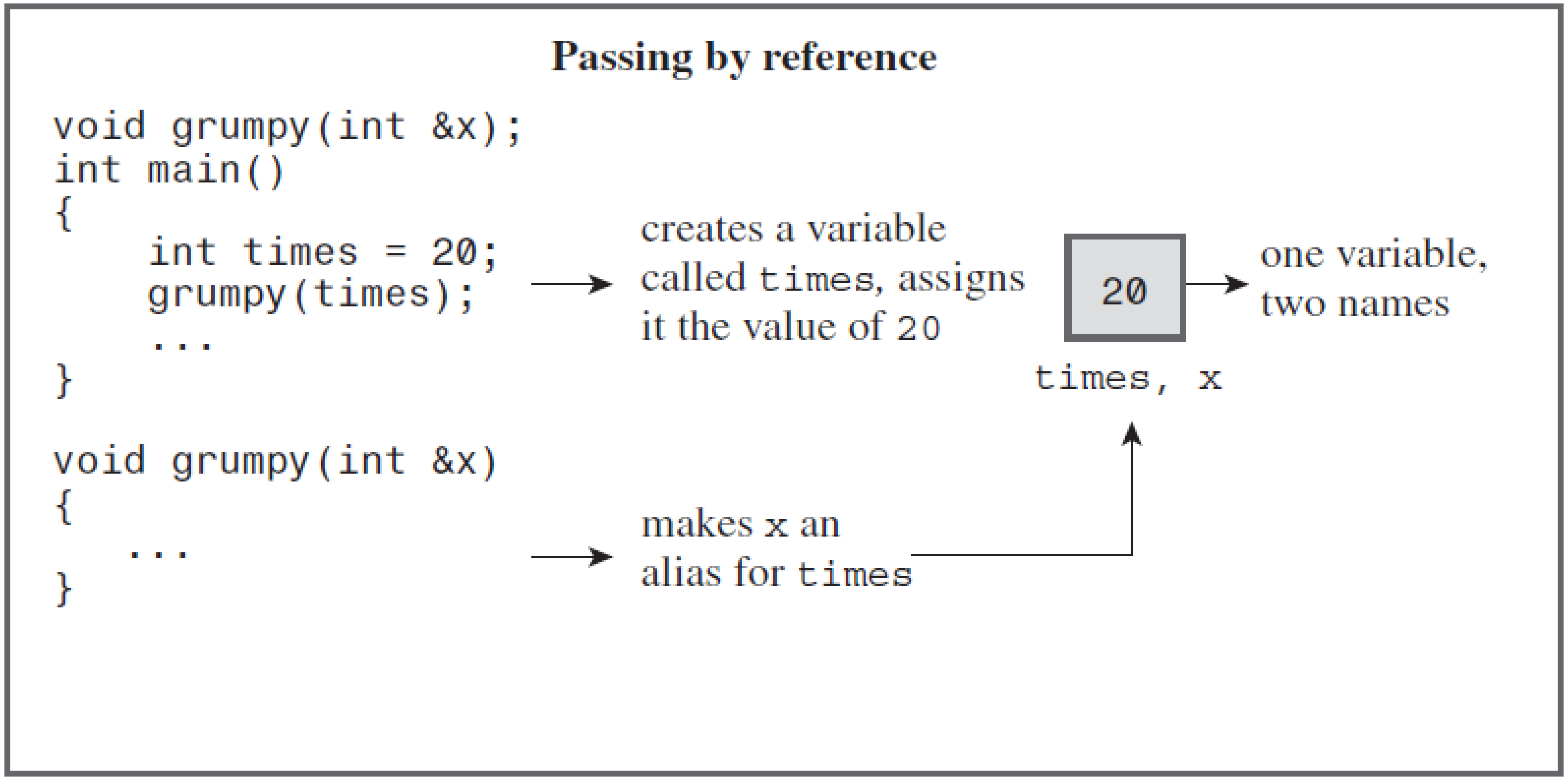


```
 ...
double cube(double x);
int main()
{

   ...
   double side = 5;
   double volume = cube(side);
   ...
}

double cube(double x)
{
 return x * x * x;
}
```

creates variable called side and assigns it the value 5
5
original value
side
passes the value 5 to the cube( ) function
creates variable called x and assigns it passed value 5
5
copied value
x

Figure 7.2 Passing by value.

Passing by reference


Passing by reference

```
void grumpy(int &x);
int main()
{
    int times = 20;
    grumpy(times);
    ...
}

void grumpy(int &x)
{
   ...
}
```

creates a variable called times, assigns it the value of 20
20
one variable, two names
times, x
makes x an alias for times

Listing 8.5 **cubes.cpp**

```
// cubes.cpp -- regular and reference arguments
#include <iostream>
double cube(double a);
double refcube(double &ra);
int main ()
{
    using namespace std;
    double x = 3.0;

    cout << cube(x);
    cout << " = cube of " << x << endl;
    cout << refcube(x);
    cout << " = cube of " << x << endl;
    return 0;
}

double cube(double a)
{
    a *= a * a;
    return a;
}

double refcube(double &ra)
{
    ra *= ra * ra;
    return ra;
}
```

Function overloading

```
void print(const char * str, int width);  // #1
void print(double d, int width);          // #2
void print(long l, int width);            // #3
void print(int i, int width);             // #4
void print(const char *str);              // #5
```

Random number

- rand() / srand(seed)
  - rand() : 0과 RAND_MAX 사이의 임의의 정수 리턴
  - srand(seed) : pseudo-random number 생성을 위한 seed 값 설정

```
v1 = rand() % 100;          // v1 in the range 0 to 99
v2 = rand() % 100 + 1;      // v2 in the range 1 to 100
v3 = rand() % 30 + 1985;    // v3 in the range 1985-2014
```

<random>

- #include <random>
  - C++ 11 에서 추가된 header
  - Generators: uniform distribution 의 난수 생성
  - Distributions: generator에서 생성된 난수를 해당 분포로 transform

- Generator
  - default_random_engine
  - mt19937
  - mt19937_64

- Distributions
  - binomial_distribution
  - poisson_distribution
  - exponential_distribution
  - normal_distribution
  - lognormal_distribution etc.

Normal random number

```
#include <iostream>
#include <random>
int main() {
        const int nrolls=10000;  // number of experiments
        const int nstars=100;    // maximum number of stars to distribute
        std::default_random_engine generator;
        std::normal_distribution<double> distribution(5.0,2.0);
        int p[10]={};
        for (int i=0; i<nrolls; ++i) {
                double number = distribution(generator);
                if ((number>=0.0)&&(number<10.0)) ++p[int(number)];
        }
        std::cout << "normal_distribution (5.0,2.0):" << std::endl;
        for (int i=0; i<10; ++i) {
                std::cout << i << "-" << (i+1) << ": ";
                std::cout << std::string(p[i]*nstars/nrolls,'*') << std::endl;
        }
        return 0;
}
```

Normal distribution pdf & cdf

- pdf is straight forward.

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- cdf is not easy. (error function을 이용함)

```
double normalCFD(double value) {
        return 0.5 * erfc(-value * M_SQRT1_2);
}
```

  - 표준정규분포의 cdf
  - M_SQRT1_2 = 1/(√2)

$$\operatorname{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2}\,\mathrm{d}t = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2}\,\mathrm{d}t.$$

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2}\,\mathrm{d}t = e^{-x^2} \operatorname{erfcx}(x),$$

- Or use boost!!

Q & A