



# NGÔN NGỮ LẬP TRÌNH

## Bài 3: Hàm và Nạp chồng hàm

Giảng viên: Lý Anh Tuấn

Email: [tuarla@tlu.edu.vn](mailto:tuarla@tlu.edu.vn)

# Nội dung

1. Hàm định nghĩa trước
  - Hàm trả về giá trị và hàm không trả về giá trị
2. Hàm người dùng định nghĩa
  - Định nghĩa, khai báo, gọi hàm
3. Phạm vi
  - Biến cục bộ
  - Hằng và biến toàn cục
4. Tham số
  - Truyền giá trị
  - Truyền tham biến
5. Nạp chồng và tham số mặc định

# Giới thiệu hàm

- Xây dựng các khối cho chương trình
- Cách gọi trong các ngôn ngữ khác
  - Thủ tục, chương trình con, phương thức
  - Trong C++: hàm
- I-P-O
  - Đầu vào – Xử lý – Đầu ra
  - Là các thành phần cơ bản của mỗi chương trình
  - Sử dụng hàm cho mỗi thành phần này

# Hàm định nghĩa trước

- Trong các thư viện có sẵn rất nhiều hàm
- Hai kiểu hàm:
  - Hàm trả về giá trị
  - Hàm không trả về giá trị (void)
- Phải “#include” thư viện phù hợp
  - Ví dụ:
    - <cmath>, <cstdlib> (các thư viện của “C”)
    - <iostream> (dùng cho cout, cin)

# Hàm định nghĩa trước

- Có rất nhiều hàm toán học
  - Nằm trong thư viện `<cmath.h>`
  - Hầu hết trả về một giá trị (câu trả lời)
- Ví dụ: `theRoot = sqrt(9.0);`
  - Các thành phần:
    - `sqrt` = tên của hàm thư viện
    - `theRoot` = biến được sử dụng để nhận câu trả lời
    - `9.0` = đối số hoặc “khởi tạo đầu vào” của hàm

# Lời gọi hàm

- Xét lệnh gán: `theRoot = sqrt(9.0);`
  - Biểu thức “`sqrt(9.0)`” được hiểu như là một lời gọi hàm
  - Đối số trong lời gọi hàm (9.0) có thể là một literal, một biến, hoặc một biểu thức
- Lời gọi có thể là một phần của biểu thức:
  - VD: `bonus = sqrt(sales)/10;`
  - Dựa vào kiểu trả về của hàm để biết nơi được phép sử dụng lời hàm

# Ví dụ: Hàm định nghĩa trước

## Display 3.1 A Predefined Function That Returns a Value

---

```
1  //Computes the size of a doghouse that can be purchased
2  //given the user's budget.
3  #include <iostream>
4  #include <cmath>
5  using namespace std;

6  int main( )
7  {
8      const double COST_PER_SQ_FT = 10.50;
9      double budget, area, lengthSide;

10     cout << "Enter the amount budgeted for your doghouse $";
11     cin >> budget;

12     area = budget/COST_PER_SQ_FT;
13     lengthSide = sqrt(area);
```

# Ví dụ: Hàm định nghĩa trước

```
14     cout.setf(ios::fixed);
15     cout.setf(ios::showpoint);
16     cout.precision(2);
17     cout << "For a price of $" << budget << endl
18         << "I can build you a luxurious square doghouse\n"
19         << "that is " << lengthSide
20         << " feet on each side.\n";

21     return 0;
22 }
```

## SAMPLE DIALOGUE

Enter the amount budgeted for your doghouse \$25.00  
For a price of \$25.00  
I can build you a luxurious square doghouse  
that is 1.54 feet on each side.



# Một số hàm định nghĩa trước

- `#include <cstdlib>`, thư viện gồm các hàm:
  - `abs()` // Trả về giá trị tuyệt đối của một số int
  - `labs()` // Trả về giá trị tuyệt đối của một số long int
  - `fabs()` // Trả về giá trị tuyệt đối của một số float
- Hàm `pow(x, y)`: Trả về  $x$  mũ  $y$ 
  - VD: Cho biết kết quả in ra của đoạn mã lệnh  
`double result, x = 3.0, y = 2.0;`  
`result = pow(x, y);`  
`cout << result;`

# Một số hàm toán học

**Display 3.2** Some Predefined Functions

NAME	DESCRIPTION	TYPE OF ARGUMENTS	TYPE OF VALUE RETURNED	EXAMPLE	VALUE	LIBRARY HEADER
sqrt	Square root	double	double	sqrt(4.0)	2.0	cmath
pow	Powers	double	double	pow(2.0,3.0)	8.0	cmath
abs	Absolute value for <code>int</code>	int	int	abs(-7) abs(7)	7 7	cstdlib
labs	Absolute value for <code>long</code>	long	long	labs(-70000) labs(70000)	70000 70000	cstdlib
fabs	Absolute value for <code>double</code>	double	double	fabs(-7.5) fabs(7.5)	7.5 7.5	cmath

# Một số hàm toán học

ceil	Ceiling (round up)	double	double	ceil(3.2) ceil(3.9)	4.0 4.0	cmath
floor	Floor (round down)	double	double	floor(3.2) floor(3.9)	3.0 3.0	cmath
exit	End program	int	void	exit(1);	None	cstdlib
rand	Random number	None	int	rand( )	Varies	cstdlib
srand	Set seed for rand	unsigned int	void	srand(42);	None	cstdlib

# Hàm void định nghĩa trước

- Không trả về giá trị
- Thực hiện một hành động, nhưng không gửi câu trả lời
- Khi được gọi, bản thân nó là một câu lệnh
  - VD: `exit(1);` //Không trả về giá trị, do vậy không được sử dụng để gán
- Các khía cạnh khác tương tự như hàm trả về giá trị

# Bộ phát sinh số ngẫu nhiên

- Trả về số “được chọn ngẫu nhiên”
- Sử dụng trong mô phỏng, trò chơi
  - `rand()`: không có tham số, trả về giá trị giữa 0 & `RAND_MAX`
  - Thu hẹp phạm vi:  
`rand() % 6`: trả về số ngẫu nhiên giữa 0 & 5
  - Tịnh tiến:  
`rand() % 6 + 1`: dịch chuyển giữa 1 & 6

# Hạt giống số ngẫu nhiên

- Các số giả ngẫu nhiên
  - Gọi rand() tạo ra một chuỗi các số ngẫu nhiên
- Sử dụng “hạt giống” để sửa đổi chuỗi  
srand(seed\_value);
  - Là hàm void có một đối số
  - Có thể sử dụng bất cứ giá trị hạt giống nào,
- VD: srand(time(0));
  - time() trả về thời gian hệ thống
  - time() nằm trong thư viện <time>

# Các ví dụ ngẫu nhiên

- Số thực ngẫu nhiên giữa 0.0 & 1.0:  
 $(\text{RAND\_MAX} - \text{rand}()) / \text{static\_cast<double>} (\text{RAND\_MAX})$ 
  - Ép kiểu cho phép chia số thực
- Số nguyên ngẫu nhiên giữa 1 & 6:  
 $\text{rand}() \% 6 + 1$ 
  - “%” là toán tử chia lấy phần dư
- Số nguyên ngẫu nhiên giữa 10 & 20:  
 $\text{rand}() \% 11 + 10$

# Hàm người dùng định nghĩa

- Cho phép tự viết hàm của riêng mình
- Xây dựng các khối chương trình
  - Chia để trị
  - Khả đọc
  - Sử dụng lại
- Định nghĩa hàm có thể nằm:
  - Cùng file với hàm main(), hoặc
  - Trong file riêng rẽ để những người khác cũng có thể sử dụng



# Các thành phần của hàm

- Khai báo hàm/nguyên mẫu hàm
  - Thông tin cho trình biên dịch
  - Thông dịch chính xác lời gọi
- Định nghĩa hàm
  - Sự thực thi hay mã lệnh thực hiện công việc của hàm
- Lời gọi hàm
  - Chuyển điều khiển cho hàm

# Khai báo hàm

- Còn được gọi là nguyên mẫu hàm
- Bộ biên dịch dựa vào nó để thông dịch lời gọi
  - Cú pháp:  
`<return_type> FnName(<formal-parameter-list>);`
  - Ví dụ:  
`double totalCost( int numberParameter,  
 double priceParameter);`
- Được đặt trước bất kỳ lời gọi nào
  - Trong không gian khai báo của hàm main()
  - Hoặc trong không gian toàn cục trước hàm main()

# Định nghĩa hàm

- Sự thực thi của hàm, giống như sự thi hàm main()

- Ví dụ:

```
double totalCost( int numberParameter,  
                  double priceParameter)  
{  
    const double TAXRATE = 0.05;  
    double subTotal;  
    subtotal = priceParameter * numberParameter;  
    return (subtotal + subtotal * TAXRATE);  
}
```

- Lưu ý thụt vào đầu dòng chuẩn

# Vị trí đặt định nghĩa hàm

- Đặt sau hàm main(), không nằm bên trong hàm main()
- Các hàm là bình đẳng, không hàm nào là thành phần của hàm khác
- Các tham số hình thức trong định nghĩa
  - Giữ chỗ cho dữ liệu gửi vào
  - Sử dụng tên biến để tham chiếu tới dữ liệu trong định nghĩa
- Lệnh return
  - Trả dữ liệu về cho lời gọi

# Lời gọi hàm

- Giống lời gọi hàm định nghĩa trước  
`bill = totalCost(number, price);`
- `totalCost` trả về giá trị kiểu `double`, giá trị này được gán cho biến `bill`
- Các đối số: `number, price`
  - Đối số có thể là literal, biến, biểu thức, hoặc sự kết hợp của chúng
  - Trong lời gọi hàm, đối số thường được gọi là “đối số thực sự”

# Ví dụ hàm

```
1  #include <iostream>
2  using namespace std;

3  double totalCost(int numberParameter, double priceParameter);
4  //Computes the total cost, including 5% sales tax,
5  //on numberParameter items at a cost of priceParameter each.

6  int main( )
7  {
8      double price, bill;
9      int number;

10     cout << "Enter the number of items purchased: ";
11     cin >> number;
12     cout << "Enter the price per item $";
13     cin >> price;

14     bill = totalCost(number, price);
```

*Function declaration;  
also called the function  
prototype*

*Function call*

# Ví dụ hàm

```
15     cout.setf(ios::fixed);
16     cout.setf(ios::showpoint);
17     cout.precision(2);
18     cout << number << " items at "
19         << "$" << price << " each.\n"
20         << "Final bill, including tax, is $" << bill
21         << endl;

22     return 0;
23 }
```

```
24 double totalCost(int numberParameter, double priceParameter)
25 {
26     const double TAXRATE = 0.05; //5% sales tax
27     double subtotal;

28     subtotal = priceParameter * numberParameter;
29     return (subtotal + subtotal*TAXRATE);
30 }
```

*Function  
head*

*Function  
body*

*Function  
definition*

## SAMPLE DIALOGUE

Enter the number of items purchased: 2  
Enter the price per item: \$10.10  
2 items at \$10.10 each.  
Final bill, including tax, is \$21.21

# Khai báo hàm thay thế

- Khai báo hàm cung cấp thông tin cho bộ biên dịch
- Bộ biên dịch chỉ cần biết:
  - Giá trị trả về
  - Tên hàm
  - Danh sách tham số
- Không cần tên tham số hình thức:  
`double totalCost(int, double);`
- Tuy nhiên vẫn nên đưa vào cho dễ đọc



# Hàm gọi hàm

- Đã làm việc này rồi: do main() là một hàm
- Khai báo hàm phải xuất hiện trước lời gọi hàm
- Định nghĩa hàm thường nằm:
  - Sau định nghĩa hàm main(), hoặc
  - Trong file riêng rẽ
- Hàm có thể gọi đến chính nó → “Đệ quy”

# Hàm trả về kiểu bool

- Kiểu trả về có thể là bất kỳ kiểu dữ liệu nào
- Một khai báo hàm/nguyên mẫu hàm:  
`bool appropriate(int rate);`
- Định nghĩa hàm tương ứng:  
`bool appropriate (int rate)  
{  
 return (((rate>=10)&&(rate<20))||(rate==0));  
}`
- Trả về “true” hoặc “false”
- Lời gọi hàm, từ một hàm khác:  
`if (appropriate(entered_rate))  
 cout << "Rate is valid\n";`

# Khai báo hàm void

- Tương tự như hàm trả về giá trị
- Kiểu trả về là “void”
- VD: Khai báo hàm/nguyên mẫu hàm:  
void showResults( double fDegrees,  
double cDegrees);
- Kiểu trả về là “void”, nghĩa là không trả về gì

# Khai báo hàm void

- Định nghĩa hàm:  

```
void showResults(double fDegrees, double cDegrees)
{
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(1);
    cout    << fDegrees
            << " degrees fahrenheit equals \n"
            << cDegrees << " degrees celsius.\n";
}
```
- Lưu ý: Không có câu lệnh return

# Gọi hàm void

- Giống như gọi hàm void định nghĩa trước
- Gọi từ một hàm khác, chẳng hạn main():  
    showResults(degreesF, degreesC);  
    showResults(32.5, 0.3);
- Không sử dụng để gán, vì không có giá trị trả về
- Các đối số thực sự (degreesF, degreesC)
  - Được truyền cho hàm
  - Hàm được gọi để thực hiện công việc với dữ liệu được truyền

# Câu lệnh return

- Chuyển điều khiển về cho lời gọi hàm
  - Phải có câu lệnh return nếu kiểu trả về khác void
  - Thường là câu lệnh cuối cùng trong định nghĩa hàm
- Với hàm void, câu lệnh return là tùy chọn
  - Dấu } sẽ chuyển điều khiển từ hàm void

# Hàm main()

- main() là một hàm
- Chỉ tồn tại duy nhất một hàm main() trong một chương trình
- Hàm main() được gọi bởi hệ điều hành
- Nó thường có câu lệnh return
  - Giá trị được trả về cho hệ điều hành
- Nên trả về “int” hoặc “void”

# Phạm vi

- Biến cục bộ
  - Được khai báo bên trong thân của một hàm
  - Chỉ tồn tại trong hàm đó
- Có thể khai báo các biến có cùng tên trong các hàm khác nhau
  - Phạm vi cục bộ: hàm là phạm vi của nó
- Lợi ích của biến cục bộ
  - Duy trì kiểm soát riêng rẽ với dữ liệu
  - Hàm nên khai báo bất kỳ dữ liệu cục bộ nào mà nó cần



# Hằng toàn cục và biến toàn cục

- Được khai báo bên ngoài thân hàm
  - Toàn cục với tất cả các hàm trong file
- Khai báo toàn cục cho hằng:
  - `const double TAXRATE = 0.05;`
  - Khai báo là toàn cục do vậy có phạm vi với tất cả các hàm
- Biến toàn cục
  - Được phép nhưng hiếm khi sử dụng
  - Khó kiểm soát khi sử dụng

# Khối

- Khai báo dữ liệu bên trong lệnh kép
  - Lệnh kép được gọi là một “khối” và có “phạm vi khối”
- Các định nghĩa hàm đều là khối
  - Tạo ra “phạm vi hàm” cục bộ
- Khối lặp:

```
for (int ctr=0;ctr<10;ctr++)  
{  
    sum+=ctr;  
}
```

  - Biến ctr chỉ có phạm vi trong khối thân vòng lặp
- Các biến có cùng tên được phép khai báo trong nhiều khối

# Các tham số

- Hai phương pháp truyền đối số làm tham số
- Truyền giá trị
  - Truyền bản sao của giá trị
- Truyền tham chiếu
  - Truyền “địa chỉ” của đối số thực sự

# Tham số truyền giá trị

- Bản sao của đối số thực sự được truyền
- Là “biến cục bộ” bên trong hàm
- Nếu sửa đổi, chỉ “bản sao cục bộ” thay đổi
  - Hàm không truy cập tới “đối số thực sự” từ lời gọi
- Đây là phương pháp mặc định
  - Được sử dụng trong tất cả các ví dụ phía trước

# Ví dụ truyền giá trị

## Display 4.1 Formal Parameter Used as a Local Variable

---

```
1  //Law office billing program.
2  #include <iostream>
3  using namespace std;

4  const double RATE = 150.00; //Dollars per quarter hour.

5  double fee(int hoursWorked, int minutesWorked);
6  //Returns the charges for hoursWorked hours and
7  //minutesWorked minutes of legal services.

8  int main()
9  {
10     int hours, minutes;
11     double bill;
```

# Ví dụ truyền giá trị


```
12 cout << "Welcome to the law office of\n"
13     << "Dewey, Cheatham, and Howe.\n"
14     << "The law office with a heart.\n"
15     << "Enter the hours and minutes"
16     << " of your consultation:\n";
17 cin >> hours >> minutes;

18 bill = fee(hours, minutes);

19 cout.setf(ios::fixed);
20 cout.setf(ios::showpoint);
21 cout.precision(2);
22 cout << "For " << hours << " hours and " << minutes
23     << " minutes, your bill is $" << bill << endl;

24 return 0;
25 }
```

*The value of minutes is not changed by the call to fee.*



(continued)

# Ví dụ truyền giá trị

## Display 4.1 Formal Parameter Used as a Local Variable

```
26 double fee(int hoursWorked, int minutesWorked)
27 {
28     int quarterHours;

29     minutesWorked = hoursWorked*60 + minutesWorked;
30     quarterHours = minutesWorked/15;
31     return (quarterHours*RATE);
32 }
```

*minutesWorked is a local variable initialized to the value of minutes.*

### SAMPLE DIALOGUE

Welcome to the law office of  
Dewey, Cheatham, and Howe.

The law office with a heart.

Enter the hours and minutes of your consultation:

**5 46**

For 5 hours and 46 minutes, your bill is \$3450.00

# Lỗi thường gặp khi truyền giá trị

- Khai báo lại tham số bên trong hàm  

```
double fee(int hoursWorked, int minutesWorked)
{
    int quarterHours;           // local variable
    int minutesWorked           // NO!
}
```
- Bộ biên dịch sẽ báo lỗi khai báo lại
- Đối số giá trị giống như “biến cục bộ”, nhưng hàm tự động có nó



# Tham số truyền tham chiếu

- Cung cấp truy cập đến đối số thực sự của lời gọi
- Dữ liệu của lời gọi có thể được sửa đổi bởi hàm được gọi
- Thường được sử dụng cho hàm đầu vào
  - Để lấy dữ liệu và đưa cho lời gọi
- Xác định bằng dấu &, sau khi nhập danh sách tham số hình thức
- Tham chiếu quay về đối số thực sự của lời gọi
  - Trỏ đến vùng nhớ của đối số thực sự
  - Được gọi là “địa chỉ”, là một số duy nhất trỏ đến một vùng riêng biệt của bộ nhớ

# Ví dụ truyền tham chiếu

## Display 4.2 Call-by-Reference Parameters

---

```
1  //Program to demonstrate call-by-reference parameters.
2  #include <iostream>
3  using namespace std;

4  void getNumbers(int& input1, int& input2);
5  //Reads two integers from the keyboard.

6  void swapValues(int& variable1, int& variable2);
7  //Interchanges the values of variable1 and variable2.

8  void showResults(int output1, int output2);
9  //Shows the values of variable1 and variable2, in that order.

10 int main()
11 {
12     int firstNum, secondNum;

13     getNumbers(firstNum, secondNum);
14     swapValues(firstNum, secondNum);
15     showResults(firstNum, secondNum);
16     return 0;
17 }
```

# Ví dụ truyền tham chiếu

```
18 void getNumbers(int& input1, int& input2)
19 {
20     cout << "Enter two integers: ";
21     cin >> input1
22     >> input2;
23 }

24 void swapValues(int& variable1, int& variable2)
25 {
26     int temp;

27     temp = variable1;
28     variable1 = variable2;
29     variable2 = temp;
30 }
31
32 void showResults(int output1, int output2)
33 {
34     cout << "In reverse order the numbers are: "
35     << output1 << " " << output2 << endl;
36 }
```

# Ví dụ truyền tham chiếu

## Display 4.2 Call-by-Reference Parameters

---

### SAMPLE DIALOGUE

Enter two integers: 5 6

In reverse order the numbers are: 6 5

---

# Tham số tham chiếu hằng

- Sử dụng đối số tham chiếu có nhiều rủi ro
  - Dữ liệu của lời gọi có thể bị thay đổi
  - Đôi khi chúng ta không mong muốn điều này
- Để bảo vệ dữ liệu, và vẫn truyền tham chiếu
  - Sử dụng từ khóa `const`  
`void sendConstRef( const int &par1,  
const int &par2);`
  - Tạo đối số “chỉ đọc” bởi hàm
  - Không cho phép thay đổi bên trong thân hàm

# Danh sách tham số trộn

- Kết hợp các kỹ thuật truyền, bao gồm các tham số truyền giá trị và truyền tham biến
- Trật tự của các đối số trong danh sách là rất quan trọng:  
`void mixedCall(int & par1, int par2, double & par3);`
- Lời gọi hàm: `mixedCall(arg1, arg2, arg3);`
  - `arg1` là kiểu nguyên, được truyền theo tham chiếu
  - `arg2` là kiểu nguyên, được truyền theo giá trị
  - `arg3` là kiểu thực, được truyền theo tham chiếu

# Nạp chồng

- Tên hàm giống nhau, danh sách các tham số khác nhau
- Hai định nghĩa hàm riêng biệt
- Tín hiệu hàm
  - Tên hàm & danh sách tham số
  - Phải là “duy nhất” cho mỗi định nghĩa hàm
- Cho phép thực hiện cùng một công việc trên các dữ liệu khác nhau

# Ví dụ nạp chồng: average

- Hàm tính giá trị trung bình của hai số:  
`double average(double n1, double n2)`  
{  
    `return ((n1 + n2) / 2.0);`  
}
- Hàm tính giá trị trung bình của 3 số:  
`double average(double n1, double n2, double n3)`  
{  
    `return ((n1 + n2 + n3) / 3.0);`  
}
- Hai hàm có cùng tên



# Nạp chồng average()

- Việc hàm nào được gọi phụ thuộc vào bản thân lời gọi hàm
  - `avg = average(5.2, 6.7);` //Gọi `average()` hai tham số
  - `avg = average(6.5, 8.5, 4.2);` //Gọi `average()` ba tham số
- Bộ biên dịch xử lý dựa trên tín hiệu của lời gọi hàm
  - Khớp lời gọi với hàm phù hợp
  - Xem xét mỗi hàm riêng biệt
- Lưu ý: chỉ nạp chồng các hàm thực hiện cùng một công việc

# Ví dụ nạp chồng

- Cho các hàm sau đây:
  1. void f(int n, double m);
  2. void f(double n, int m);
  3. void f(int n, int m);
- Các lời gọi:
  - f(98, 99); → Calls #3
  - f(5.3, 4); → Calls #2
  - f(4.3, 5.2); → Calls ???
- Tránh việc nạp chồng nhập nhằng

# Chuyển kiểu tự động và nạp chồng

- Tham số dạng số thường ở kiểu "double"
- Cho phép bất kỳ kiểu số nào: dữ liệu phụ thuộc tự động được chuyển đổi
  - int → double
  - float → double
  - char → double
- Tránh việc nạp chồng cho các kiểu số khác nhau

# Chuyển kiểu tự động và nạp chồng

- `double mpg(double miles, double gallons)`  
    {  
        return (miles/gallons);  
    }

Các lời gọi ví dụ:

- `mpgComputed = mpg(5, 20);`
  - Chuyển 5 & 20 thành doubles, rồi truyền
- `mpgComputed = mpg(5.8, 20.2);`
  - Không cần chuyển kiểu
- `mpgComputed = mpg(5, 2.4);`
  - Chuyển 5 thành 5.0, sau đó truyền các giá trị cho hàm

# Các tham số mặc định

- Cho phép lờ đi một vài tham số
- Được chỉ ra trong khai báo hàm  

```
void showVolume(          int length,  
                      int width = 1,  
                      int height = 1);
```

  - Hai tham số cuối là mặc định
- Các lời gọi có thể có:
  - `showVolume(2, 4, 6);` //tất cả các tham số được cung cấp
  - `showVolume(3, 5);` //height mặc định là 1
  - `showVolume(7);` //width & height mặc định là 1

# Các tham số mặc định

## Display 4.8 Default Arguments

```
1
2  #include <iostream>
3  using namespace std;

4  void showVolume(int length, int width = 1, int height = 1);
5  //Returns the volume of a box.
6  //If no height is given, the height is assumed to be 1.
7  //If neither height nor width is given, both are assumed to be 1.

8  int main( )
9  {
10     showVolume(4, 6, 2);
11     showVolume(4, 6);
12     showVolume(4);

13     return 0;
14 }

15 void showVolume(int length, int width, int height)
```

*Default arguments*

*A default argument should not be given a second time.*

# Các tham số mặc định

```
16 {  
17     cout << "Volume of a box with \n"  
18         << "Length = " << length << ", Width = " << width << endl  
19         << "and Height = " << height  
20         << " is " << length*width*height << endl;  
21 }
```

## SAMPLE DIALOGUE

Volume of a box with  
Length = 4, Width = 6  
and Height = 2 is 48  
Volume of a box with  
Length = 4, Width = 6  
and Height = 1 is 24  
Volume of a box with  
Length = 4, Width = 1  
and Height = 1 is 4

# Tóm tắt

- Hai kiểu hàm: hàm trả về giá trị và hàm void
- Dữ liệu cục bộ: được khai báo trong định nghĩa hàm
- Dữ liệu toàn cục: Được khai báo bên ngoài các định nghĩa hàm, phù hợp cho hằng nhưng không phù hợp cho biến
- Tham số/Đối số:
  - Hình thức: Trong khai báo và định nghĩa hàm
  - Thực sự: Trong lời gọi hàm



# Tóm tắt

- Tham số hình thức là để giữ chỗ, được điền bằng đối số thực sự trong lời gọi hàm
- Tham số truyền giá trị là bản sao cục bộ trong thân hàm nhận
- Truyền tham chiếu truyền địa chỉ bộ nhớ của đối số thực sự
- Được phép viết nhiều định nghĩa cho cùng một tên hàm: được gọi là nạp chồng
- Các tham số mặc định cho phép lời gọi hàm bỏ đi một số đối số trong danh sách