LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
(Object Oriented Programming)

KHOA CÔNG NGHỆ THÔNG TIN, ĐH SƯ PHẠM HÀ NỘI

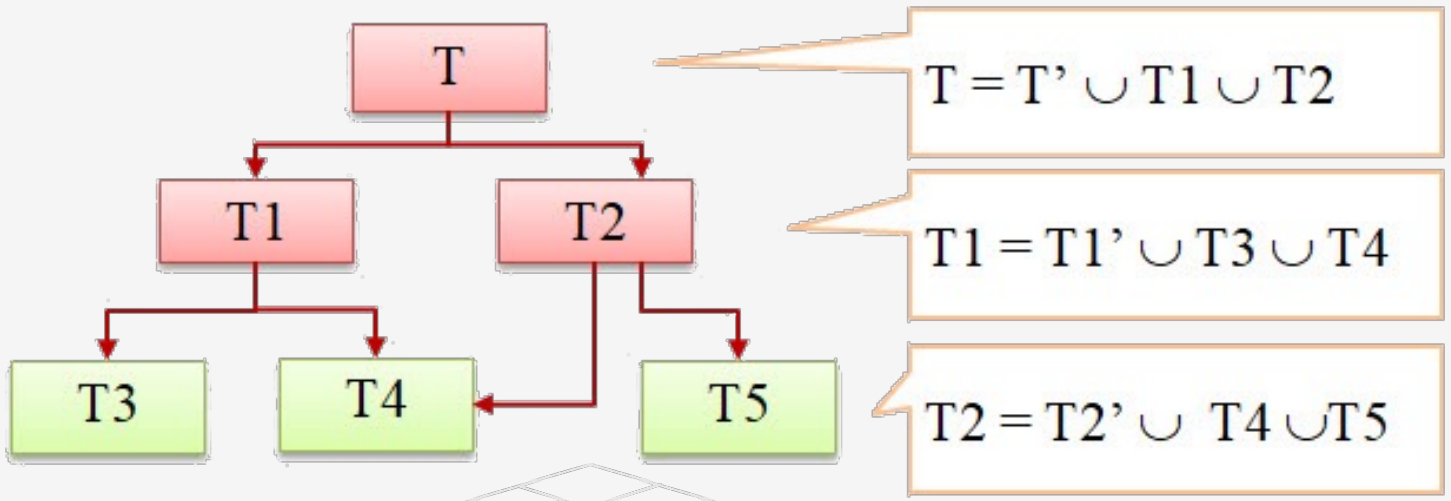
HÀ NỘI, THÁNG 9-2022

CHƯƠNG 2: HÀM VÀ TRUYỀN THAM SỐ CHO HÀM

- Khái niệm hàm
- Định nghĩa, khai báo và sử dụng
- Truyền tham số cho hàm
- Biến toàn cục và cục bộ
- Nạp chồng hàm
- Kỹ thuật đệ quy

TƯ TƯỞNG CHIA ĐỂ TRỊ

- Chia để trị - Lập trình top-down



T = T' ∪ T1 ∪ T2 nghĩa là *Bài toán T giải bằng thuật toán T', trong đó sử dụng hai bài toán con là T1 và T2.*

TƯ TƯỞNG CHIA ĐỂ TRỊ

- Trong lập trình top-down, mỗi bài toán con được giải quyết bởi một ***chương trình con***.
- Trong C++, chương trình con là ***hàm*** ***(****hàm thành phần)*.
- **Ví dụ**: Mỗi công việc thành một hàm
  - T1, T2, T3… là các hàm.
- Bản thân main() là hàm buộc phải có, là nơi bắt đầu chương trình

TƯ TƯỞNG "SỬ DỤNG LẠI"

- Hàm là một số câu lệnh được nhóm lại thành một đơn vị và được đặt tên riêng.
- Hàm sau khi định nghĩa có thể được các phần khác của chương trình sử dụng thông qua các lời gọi hàm.

TƯ TƯỞNG "SỬ DỤNG LẠI"

Ví dụ 2-1: Viết chương trình in ra miền giá trị của các kiểu `char`, `short` và `long;` Kết quả được biểu diễn như hình dưới

```
**********************************************

Khoang gia tri cua kieu du lieu

**********************************************

Kieu char: -128 den 127

Kieu short: -32,768 den 32,767

Kieu long: -2,147,483,648 den 2,147,483,647

**********************************************
```

```
1  #include <iostream>
2  using namespace std;
3  // khai bao ham
4  void star_line() {
5      cout << "**********************";
6  }
7  int main() {
8      star_line(); //loi goi ham
9      cout << "Khoang gia tri cua kieu du lieu" << endl;
10     star_line(); //loi goi ham
11     cout << "char -128 to 127" << endl
12         << "short -32,768 to 32,767" << endl
13         << "long -2,147,483,648 to 2,147,483,647" << endl;
14     star_line(); //loi goi ham
15     return 0;
16 }
```

CẤU TRÚC MỘT HÀM

- **Tiêu đề**:
  - ***<Kieu_ham>*:** Kiểu giá trị mà hàm có thể mang
  - ***<Ten_ham>*:** định danh cho hàm, được đặt theo quy tắc đặt tên biến, có phân biệt chữ hoa chữ thường
  - ***<Danh_sach_tham_so>*:** dữ liệu truyền vào cho hàm, góp phần định danh hàm

```
<kieu_ham> <ten_ham> ([<danh_sach_tham_so>])
{
        <cac_cau_lenh>;
        [return <gia_tri>]
}
```

- **Thân hàm** là đoạn mã nằm trong dấu ngoặc kép {}
  - Là tập các câu lệnh (lệnh ghép)
  - Có lệnh `return` nếu `<kieu_ham>` khác `void`

VÍ DỤ KHAI BÁO HÀM

```
int demchuso(int N) {
    int d = 0;
    while (N != 0) {
        N = N / 10;
        d = d + 1;
    }
    return d;
}
```

Hàm có giá trị tả về

Hàm không có giá trị trả về (thủ tục)

```
void hienthi(double x1, double x2) {
    cout << "Phuong trinh co hai nghiem: ";
    cout << "x1 = " << x1 << endl;
    cout << "x2 = " << x2 << endl;
}
```

VỊ TRÍ CỦA HÀM

- Hàm phải được khai báo trước khi sử dụng (gọi hàm).
- Có thể định nghĩa hàm trước hoặc sau hàm `main()`
- Nếu định nghĩa sau hàm `main()`
  - Phải khai báo nguyên mẫu (prototype) của hàm ở đầu chương trình (Trước hàm `main()`)
  - Phần triển khai nội dung hàm (định nghĩa hàm) có thể đặt trước hoặc sau hàm `main()`
- Nguyên mẫu hàm:
  - Khai báo kiểu hàm, tên hàm và các tham số, kết thúc bằng dấu chấm phẩy
  - Ví dụ: `int  SUM(int x, int y);` => nguyên mẫu của hàm SUM
  - Chú ý:
    - Trong định nghĩa hàm, danh sách tham số, và kiểu dữ liệu của hàm phải trùng khớp với những gì đã khai báo trong phần nguyên mẫu.

VỊ TRÍ CỦA HÀM

```cpp
#include <iostream>
using namespace std;
int x = 0;
//-----------------
void f1()
{
    x++;
}
//-----------------
void f2()
{
    x+=4;
    f1();
}
//-----------------
int main()
{
    f2();
    cout << x << endl;
    return 0;
}
```

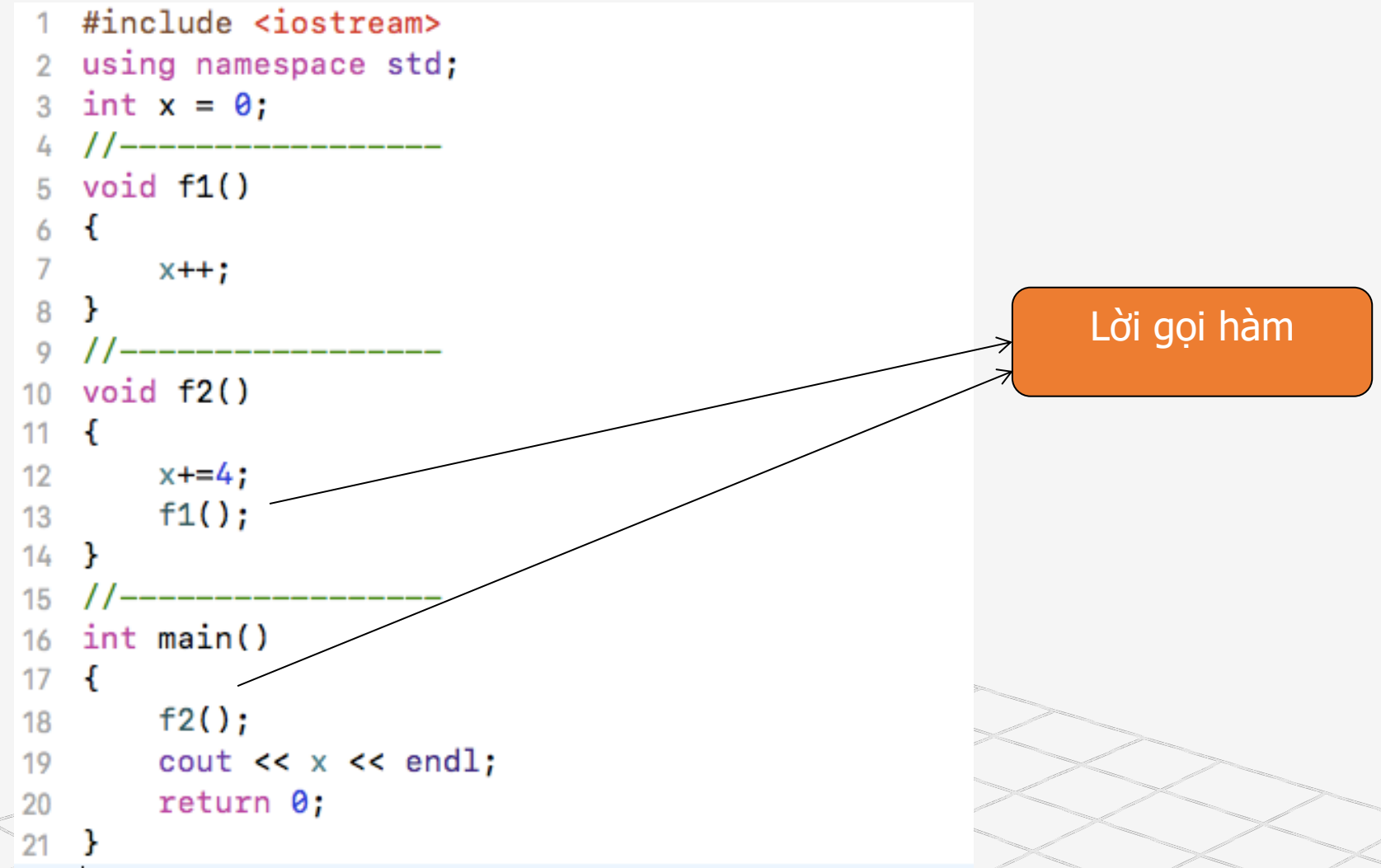
```cpp
#include <iostream>
using namespace std;
int x;
void f1();
void f2();
//-----------------
int main()
{
    f2();
    cout << x << endl;
    return 0;
}
//-----------------
void f1()
{
    x++;
}
//-----------------
void f2()
{
    x += 4;
    f1();
}
```

LỜI GỌI HÀM

- Hàm được gọi bất kì nơi đâu trong chương trình, có thể từ hàm main() hoặc một hàm khác trong chương trình.
- Khi hàm A muốn gọi hàm B thì hoặc là B phải triển khai trước A hoặc là B đã khai báo prototype ở đầu chương trình.
- Hàm được gọi thông qua tên hàm và danh sách các tham số.
  - Ví dụ : Sum(10,20) hoặc Sum(20,a),..
- Hàm có thể được gọi theo nhiều cách khác nhau
- Trong C++, không có các hàm lồng nhau

LỜI GỌI HÀM

```
1  #include <iostream>
2  using namespace std;
3  int x = 0;
4  //-----------------
5  void f1()
6  {
7      x++;
8  }
9  //-----------------
10 void f2()
11 {
12     x+=4;
13     f1();
14 }
15 //-----------------
16 int main()
17 {
18     f2();
19     cout << x << endl;
20     return 0;
21 }
```



CÁC THÀNH PHẦN CỦA HÀM

- Kiểu hàm: là một kiểu dữ liệu chuẩn của C++ (`int, bool, double`...) hoặc kiểu do người dùng tự định nghĩa
- Trường hợp hàm không có giá trị trả về
  - Sử dụng từ khóa void để mô tả kiểu của hàm

```
void  <ten_ham>( <danh sach tham so>)
        { …. }
```

  - `void`: là một kiểu dữ liệu đặc biệt của C++ để mô tả sự việc không có giá trị
- Ví dụ: `int tong(int a, int b)`

CÁC THÀNH PHẦN CỦA HÀM

- Lệnh return
  - Trả về giá trị của hàm, có thể trả về là giá trị của 1 biểu thức
  - Có thể có nhiều lệnh return trong 1 hàm
  - Đối với hàm có giá trị trả về, phải CÓ ít nhất 1 lệnh return
  - Là lệnh cuối cùng của một hàm
- Ví dụ lệnh return

```
int tong(int a, int b)
{
    int t = a+b;
    return t;
}
```

```
16 int main()
17 {
18     int a,b;
19     nhapDL(a);
20     nhapDL(b);
21    // tong(a,b); //xay ra loi
22     cout << "Tong cua a va b la:" << tong(a,b) << endl;
23     return 0;
24 }
```

CÁC THÀNH PHẦN CỦA HÀM

- Danh sách tham số: các tham số hình thức của hàm
- Trường hợp hàm không có tham số
  - Sử dụng từ khóa void để mô tả danh sách tham số,

```
<kieu_ham> <ten_ham>(void)
```

  - Hoặc để trống

```
<kieu_ham>  <ten_ham>()
```

- Trường hợp hàm không khai báo kiểu tường minh
  - Sẽ sử dụng kiểu ngầm định là int

HÀM CÓ THAM SỐ VÀ KHÔNG CÓ THAM SỐ

Ví dụ 2-2 . Viết chương trình nhập và in một số nguyên dương (có sử dụng hàm để nhập)

```
1  #include <iostream>
2  using namespace std;
3  int N;
4  //-----------------
5  void nhapDL()
6  {
7      cout << "Nhap vao mot so nguyen duong: ";
8      cin >> N;
9  }
10 //-----------------
11 int main()
12 {
13     nhapDL();
14     cout << "Gia tri da nhap la: "<< N << endl;
15     return 0;
16 }
```

```
1  #include <iostream>
2  using namespace std;
3  //-----------------
4  void nhapDL(int &n)
5  {
6      cout << "Nhap vao mot so nguyen duong: ";
7      cin >> n;
8  }
9  //-----------------
10 int main()
11 {
12     int N;
13     nhapDL(N);
14     cout << "Gia tri da nhap la: "<< N << endl;
15     return 0;
16 }
```

BÀI TẬP THỰC HÀNH TẠI LỚP

Bài 2-1: Viết một chương trình sử dụng hàm để in ra bảng cửu chương như hình.

x	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

TRUYỀN THAM SỐ CHO HÀM

- Tham số hình thức : là tham số đầu vào khi khai báo, định nghĩa hàm
- Tham số thực sự (đối số) là giá trị cụ thể truyền vào khi gọi hàm
  - Giá trị có thể là hằng số
  - Giá trị của biến
- Lưu ý:
  - Tham số hình thức phải được khai báo kiểu dữ liệu, có thể không cần xác định tên
  - Khi gọi hàm, tên tham số thực sự được chỉ ra, không cần nêu kiểu dữ liệu.
- Hai kiểu truyền tham số:
  - Truyền tham trị
  - Truyền tham chiếu

PHÂN BIỆT THAM TRỊ VÀ THAM CHIẾU

- Truyền **tham trị**:
  - Về bản chất khi gọi hàm với tham trị thì chỉ truyền cho hàm giá trị thực của các hàm hoặc các biến
  - Sau khi ra khỏi hàm thì giá trị của các biến đã truyền vào không hề thay đổi.
- Truyền **tham chiếu**:
  - Bản chất gọi hàm với tham biến là truyền cho hàm địa chỉ của các biến, khi đó các thao tác trong hàm sẽ là thao tác trực tiếp tại địa chỉ của biến.
  - Ra khỏi hàm giá trị của các biến sẽ thay đổi theo các thao tác trong hàm.
  - Tham chiếu được chỉ ra bằng cách sử dụng khai báo với kí hiệu **&** trước tên của nó

```
nhapDL(int &n)
```

- Lựa chọn: Khi hàm không có nhiệm vụ thay đổi giá trị tham số đầu vào nào thì tham số đó để kiểu *tham trị*, ngược lại để kiểu *tham chiếu*

THAM CHIẾU VÀ THAM TRỊ

- Tham số truyền vào hàm (nếu có) chia thành 2 loại: ***Tham trị*** và ***tham chiếu***
- Ví dụ 2-2. Quan sát định nghĩa hàm và kết quả

```
1  #include <iostream>
2  using namespace std;
3  //-----------------
4  void nhapDL(int n)
5  {
6      cout << "Nhap vao mot so nguyen duong: ";
7      cin >> n;
8  }
9  //-----------------
10 int main()
11 {
12     int N;
13     nhapDL(N);
14     cout << "Gia tri da nhap la: "<< N << endl;
15     return 0;
16 }
```

```
1  #include <iostream>
2  using namespace std;
3  //-----------------
4  void nhapDL(int &n)
5  {
6      cout << "Nhap vao mot so nguyen duong: ";
7      cin >> n;
8  }
9  //-----------------
10 int main()
11 {
12     int N;
13     nhapDL(N);
14     cout << "Gia tri da nhap la: "<< N << endl;
15     return 0;
16 }
```

GIÁ TRỊ MẶC ĐỊNH CỦA THAM SỐ

- Khi định nghĩa hàm, chúng ta có thể định nghĩa giá trị mặc định sẽ được truyền cho các tham số trong trường hợp nó bị bỏ qua khi gọi hàm.
- Nếu giá trị của tham số đó vẫn được chỉ định khi gọi hàm, thì giá trị mặc định sẽ bị bỏ qua.
- Chú ý: Khi khai báo có giá trị mặc định cho tham số thì các tham số có giá trị mặc định phải khai báo sau các tham số không có giá trị mặc định

VÍ DỤ:

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

int Sum(int  a, int  b = 5) {
    return a + b;
}

main() {
    int x = Sum(10,20);
    int y = Sum(20);
    cout<<"X = "<<x;
    cout<<"Y = "<<y;
    return 0;
}
```

Hàm có giá trị tả về

HÀM CÓ THAM SỐ VÀ KHÔNG CÓ THAM SỐ

Ví dụ 2-3 . Viết chương trình kiểm tra một số có là số nguyên tố hay không và in ra thông báo

```cpp
#include <iostream>
#include <math.h>
using namespace std;
bool kiemtra_nto(int n)
{
    for (int i = 2; i< sqrt(n); i++)
        if (n % i == 0)
            return false;
    return true;
}
```

```cpp
int main()
{
    int n;
    cout << "Nhap n: "; cin >> n;
    if (kiemtra_nto(n) == true)
        cout << n << " la so nguyen to";
    else
        cout << n << " khong la so nguyen to";

    return 0;
}
```

BÀI TẬP THỰC HÀNH TẠI LỚP

Bài 2-2. Viết hàm kiểm tra một số có phải số ***chính phương*** hay không. Dựa trên hàm đó, viết chương trình:

- Hiển thị tất cả các số chính phương nhỏ hơn 10000.
- Đếm tất cả số chính phương nhỏ hơn 1000

BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

- Biến toàn cục:
  - Biến được khai báo bên ngoài các hàm
  - Tất cả các hàm có thể thấy và sử dụng
  - Tồn tại trong suốt thời gian hoạt động của chương trình
- Biến cục bộ:
  - Biến cục bộ được khai báo bên trong thân hàm,
  - Tồn tại trong thời gian hàm thực hiện và bị huỷ khi thoát khỏi hàm
  - Phải khởi tạo giá trị cho biến cục bộ trước khi sử dụng
- Trong trường hợp biến cục bộ và biến toàn cục có cùng tên, hàm sẽ sử dụng biến cục bộ

BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

```
int x = 10; //bien toan cuc

int main()

{
    int y = 5; //bien cuc bo

    if (x > y)

    {
        int z = 11; //bien cuc bo

    }

    for (int i = 0; i < 10; i++) //bien cuc bo

    {

    }


}
```

Phạm vi hoạt động của x

Phạm vi hoạt động của y

Phạm vi hoạt động của z

Phạm vi hoạt động của i

BÀI THỰC HÀNH TẠI LỚP

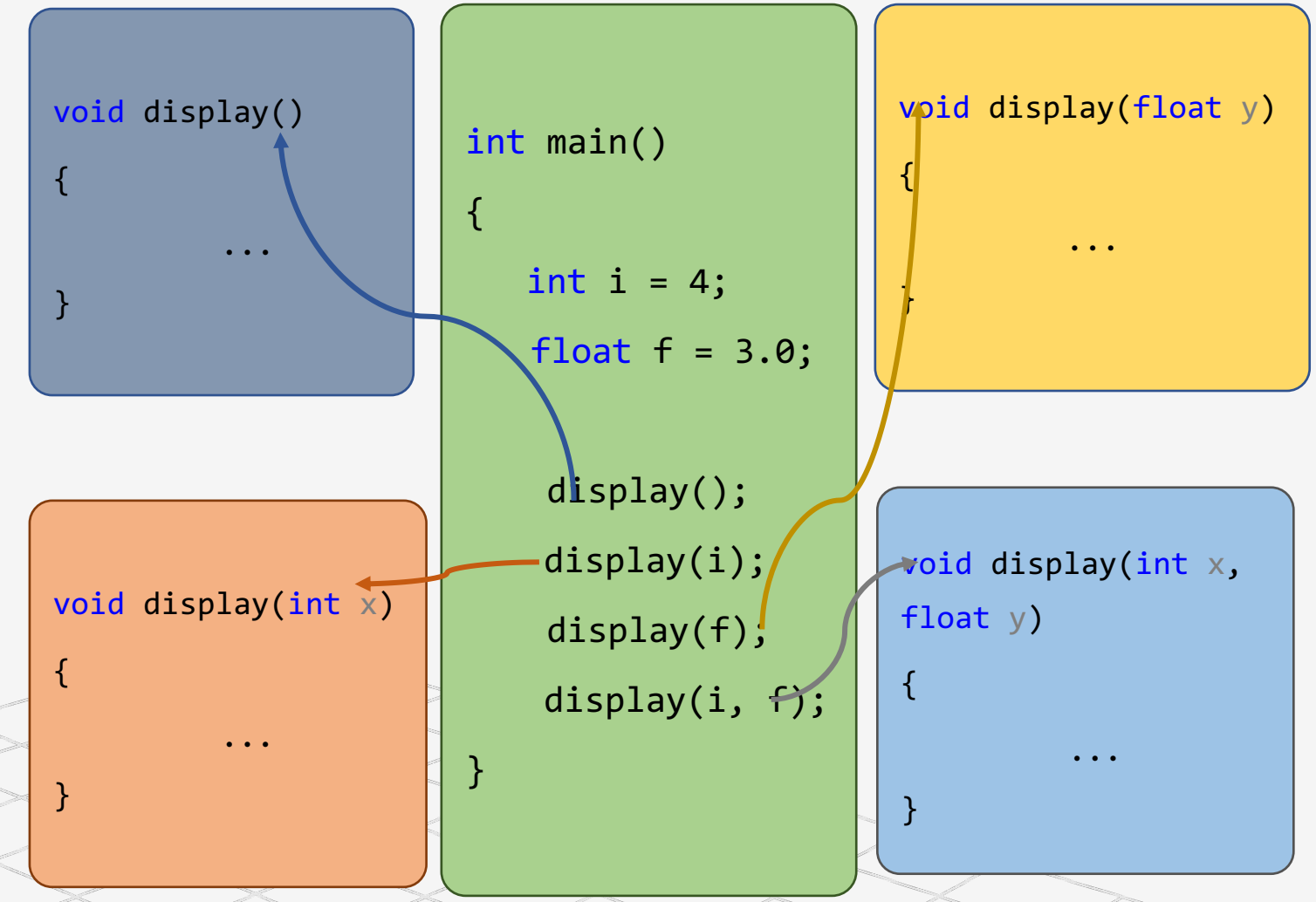
Bài 2.3:

Viết chương trình gồm các hàm sau đây:

- Nhập vào số nguyên dương
- Cho biết số chữ số của N
- Cho biết chữ số lớn nhất của N
- Cho biết chữ số nhỏ nhất của N
- Cho biết N có phải là một số nguyên tố hay không

NẠP CHỒNG HÀM

- Sử dụng nhiều hàm cùng tên, thực hiện công việc tương tự
- Khác nhau:
  - Số lượng tham số
  - Kiểu tham số
  - Vị trí tham số



NẠP CHỒNG HÀM

Ví dụ 2-4. Viết chương trình có các hàm cho phép:

- Hiển thị dòng gồm 45 kí tự '*'
- Hàm cho phép hiển thị 45 kí tự *ch* (kí tự *ch* do người dùng chọn)
- Hàm cho phép hiển thị *n* kí tự bất kỳ ch(*n* và *ch* do người dùng chọn)

NẠP CHỒNG HÀM

Ví dụ 2-4

```
1  #include <iostream>
2  using namespace std;
3
4  void display_char_line(); //khai bao cac h
5  void display_char_line(char);
6  void display_char_line(char, int);
7
8  int main()
9  {
10         display_char_line();
11         display_char_line('=');
12         display_char_line('+', 30);
13         return 0;
14 }
```

```
16 // display_char_line() : hien thi 45 dau *
17 void display_char_line()
18 {
19         for (int j = 0; j < 45; j++) // lap 45 lan
20                 cout << '*'; // hien thi dau *
21         cout << endl;
22 }
23 //-------------------------------------------------------
24 // display_char_line(): Hien thi 45 lan mot ki tu ch
25 void display_char_line(char ch)
26 {
27         for (int j = 0; j < 45; j++) // Lap lai 45 lan
28                 cout << ch; // in ki tu da duoc xac dinh ch
29         cout << endl;
30 }
31 //-------------------------------------------------------
32 // display_char_line(): hien thi mot so n lan mot ki tu ch
33 void display_char_line(char ch, int n)
34 {
35         for (int j = 0; j < n; j++) // lap n lan
36                 cout << ch; // in ki tu da xac dinh
37         cout << endl;
38 }
```

BÀI TẬP THỰC HÀNH TẠI LỚP

Bài 2-4: Viết hàm tính trung bình nhân của 2 số nguyên, hàm tính trung bình nhân của 2 số thực (sử dụng nạp chồng hàm)

KỸ THUẬT ĐỆ QUY

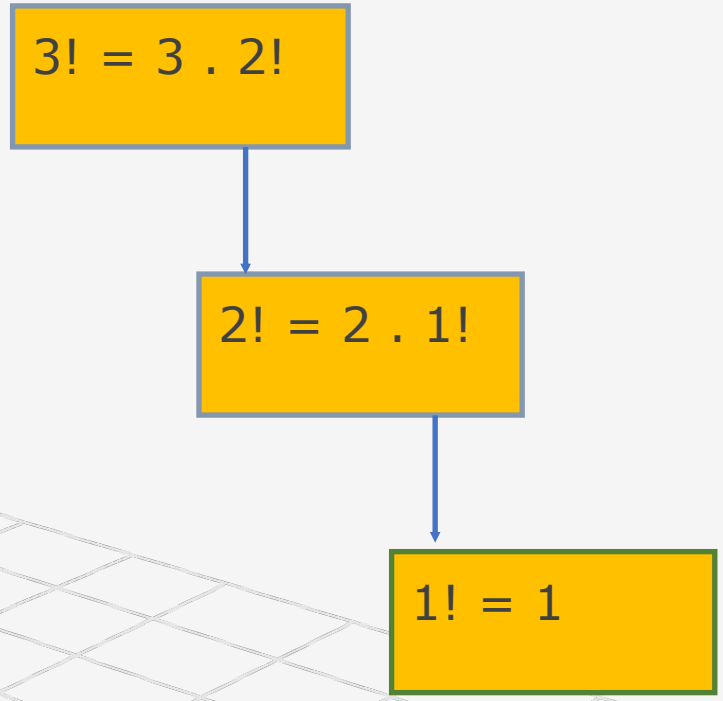
- Tư tưởng:
  - Xác định trường hợp cơ bản
  - Suy biến bài toán về trường hợp cơ bản
- Cài đặt: sử dụng hàm
  - Hàm gọi lại chính nó
  - Dừng khi gọi tới trường hợp cơ bản

Ví dụ 2-5. Viết chương trình thực hiện yêu cầu sau:

- Nhập vào một số nguyên N (N<20)
- Tính và in ra giá trị của N giai thừa (N!) sử dụng kỹ thuật đệ quy

Giải thuật: N = 3:

- Sử dụng hàm factorial(n)
  - factorial(3) = 3*factorial(2)
  - factorial(2) = 2*factorial(1)
  - factorial(1) = 1;



KỸ THUẬT ĐỆ QUY

```
23 int factorial(int n)
24 {
25     if (n==1)
26         return 1;
27     else
28         return n * factorial(n-1);
29 }
30 //---------------
31 int main()
32 {
33     int N;
34     nhapDL(N);
35     cout << "N! = " << factorial(N) << endl;
36     return 0;
37 }
```

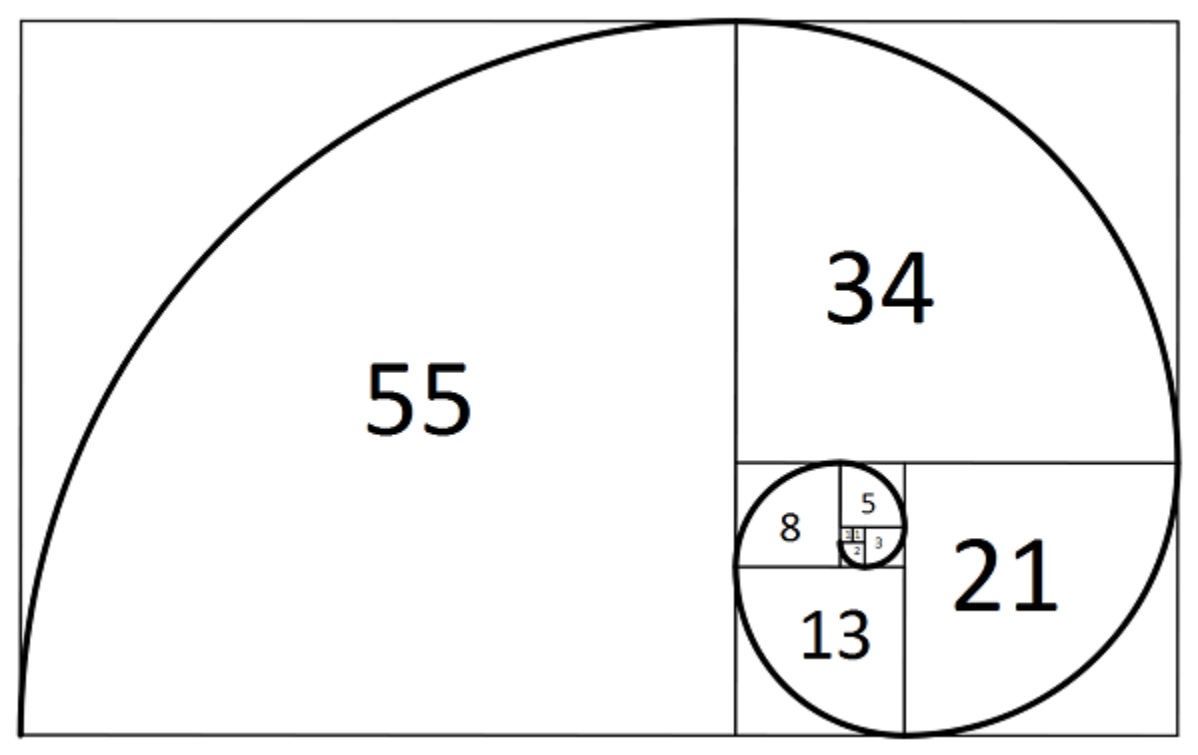
Output:

- N=1 => N!=1
- N=3 => N!=6
- N=5 => N!=120
- N=10 => N!=3628800
- N=19 =>
  N! = 109641728

- **N=20????**

BÀI TẬP THỰC HÀNH TẠI LỚP

Bài 2-5: Viết chương trình tìm n số Fibonacci đầu tiên. Số nguyên dương n được nhập từ bàn phím.



Quy luật của dãy số Fibonacci: số tiếp theo bằng tổng của 2 số trước, 2 số đầu tiên của dãy số là 0, 1.

Ví dụ: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...