# Chương 5. HÀM

1. Giới thiệu
2. Module chương trình trong C
3. Các hàm thư viện toán
4. Hàm
5. Định nghĩa hàm
6. Hàm nguyên mẫu
7. Ngăn xếp gọi hàm và khung ngăn xếp
8. Header
9. Truyền tham biến và truyền tham trị
10. Cấp số ngẫu nhiên
11. Đệ quy
12. Đệ quy và lặp

# 1. Giới thiệu

- Phát triển và bảo dưỡng CT lớn → module →chia nhỏ
- Giới thiệu và cung cấp kỹ năng thiết kế, thực hiện, vận hành, bảo dưỡng CT lớn

# 2. Module CT trong C

- Các module trong C gọi là các hàm
- CT C bao gồm: hàm người lập trình viết + hàm trong thư viện chuẩn
- Hàm trong TV chuẩn gồm:  vào/ra, thao tác ký tự, thao tác chuỗi, toán học...

  → lập trình viên dễ thực hiện công việc

  → Nên sử dụng hàm trong thư viện chuẩn
- Các hàm được gọi bởi lệnh gọi hàm

# Gọi hàm

- Gọi tên hàm và giá trị biến được truyền vào khi gọi hàm
- Hàm diễn tả hành động hay thao tác
- Hàm chỉ trả kết quả
- VD: Sếp giao việc cho nhân viên
  - Nhân viên lấy thông tin, thực hiện và trả kết quả
  - Thông tin được ẩn đi

# 3. Hàm thư viện toán

- Thực hiện các phép toán thông dụng
- Khai báo thư viện: #include <math.h>
- Định dạng lời gọi hàm **function_name (argument);**
- Nếu nhiều argument thì dùng dấu " , " để tách ra
- VD: printf ("%.2f",sqrt (900.0));
  - Sqrt: trả về giá trị căn bậc hai
  - Tất cả các hàm toán trả về giá trị "double)
- Argument (giá trị biến truyền vào có thể là hằng số, biến, biểu thức.
- Các hàm trong thư viện toán

| Method | Description | Example |
|---|---|---|
| **ceil( x )** | làm tròn *x* tới số nguyên nhỏ nhất không nhỏ hơn *x* | ceil( 9.2 ) is 10.0<br>ceil( -9.8 ) is -9.0 |
| **cos( x )** | cos của *x* (lượng giác)<br>(*x* tính theo đơn vị radian) | cos( 0.0 ) is 1.0 |
| **exp( x )** | hàm mũ: *e mũ x* | exp( 1.0 ) is 2.71828<br>exp( 2.0 ) is 7.38906 |
| **fabs( x )** | giá trị tuyệt đối của *x* | fabs( 5.1 ) is 5.1<br>fabs( 0.0 ) is 0.0<br>fabs( -8.76 ) is 8.76 |
| **floor( x )** | làm tròn *x* xuống số nguyên lớn nhất không lớn hơn *x* | floor( 9.2 ) is 9.0<br>floor( -9.8 ) is -10.0 |
| **fmod( x, y )** | phần dư của phép chia *x/y* , tính bằng kiểu số thực | fmod( 13.657, 2.333 ) is 1.992 |
| **log( x )** | loga tự nhiên của x (cơ số *e*) | log( 2.718282 ) is 1.0<br>log( 7.389056 ) is 2.0 |
| **log10( x )** | loga cơ số 10 của *x* | log10( 10.0 ) is 1.0<br>log10( 100.0 ) is 2.0 |
| **pow( x, y )** | *x* mũ *y* | pow( 2, 7 ) is 128<br>pow( 9, .5 ) is 3 |
| **sin( x )** | sin *x* (lượng giác)<br>(*x* tính theo radian) | sin( 0.0 ) is 0 |
| **sqrt( x )** | căn bậc hai của *x* | sqrt( 900.0 ) is 30.0<br>sqrt( 9.0 ) is 3.0 |
| **tan( x )** | tang *x* (lượng giác)<br>(*x* tính theo radian) | tan( 0.0 ) is 0 |

# 4. Hàm

- Module hóa một chương trình
- Tất cả biến được định nghĩa trong hàm là biến cục bộ, chỉ sử dụng trong hàm được định nghĩa
- Việc trao đổi thông tin giữa các hàm là các biến cục bộ
- Ưu điểm của hàm:
  - Chia nhỏ để quản lý, bảo dưỡng và phát triển chương trình
  - Tránh việc lặp lại code
  - Sử dụng những hàm đang tồn tại như 1 block cho các CT khác
  - Ẩn đi các chi tiết

# 5. Định nghĩa hàm

- Định dạng

  **return_value_type function_name (parameter_list)**

  **{declarations & statements }**

- Trong đó:
  - **return_value_type:** mặc định là **int**, không trả về dùng **void**
  - **function_name:** Hợp lệ
  - **parameter_list:** cách nhau dấu “ **,** ” , khai báo thông số rõ ràng
- Thân hàm: Biến được định nghĩa trong thân; hàm không được định nghĩa bên trong các hàm khác
- Điều khiển trả về:
  - Không trả về: return; hoặc }
  - Trả về: return expression;

# 6. Hàm nguyên mẫu

- Dùng để sử dụng 1 hàm mà chưa cần định nghĩa ngay
- Gồm:
  - Tên hàm
  - Kiểu dữ liệu hàm trả về
  - Đối số mà hàm cần cung cấp
  - Kiểu dữ liệu đối số
  - Thứ tự đối  số
- VD: int add (int a, int b);
- Tương tự như dòng tiêu đề trong khai báo hàm nhưng có dấu “ ; ” ở cuối
- Tên đối số không ảnh hưởng đến quá trình biên dịch → Có thể đổi tên đối số
- VD: int add (int, int);

- Định nghĩa  hàm bình phương
- Định nghĩa hàm tìm max trong 3 số
- Sử dụng hàm tính y^x trong thư viện toán học
- ĐN hàm tính tổng các số tự nhiên từ 1 → n
- ĐN hàm tính tổng các số tự nhiên từ a → b
- Định nghĩa hàm in ra n ký tự *
- Định nghĩa hàm in ra n ký tự bất kỳ

Viết 01 CTC cho nhập vào 3 số nguyên, tìm max sử dụng hàm, tính và in ra tổng S các số tự nhiên từ min – max (sd hàm). Tính và in ra max^min dùng hàm trong thư viện math.h. In ra S ký tự * sử dụng hàm

```c
#include <stdio.h>
#include <conio.h>
int square(int);
// function main begins program execution
int main(void)
{
     int n;
     for (n = 1; n < 10; n++)
     printf("%d\n", square (n));
     getch();
}
// Function square definition
int square(int x)
{
     return x*x;
}
```

```c
#include <stdio.h>
#include <conio.h>
int main(void){
	int n1, n2, n3;
	printf("%s", "Enter three integers: ");
	scanf("%d%d%d", &n1, &n2, &n3);
	int max = n1;
	if (n2 > max)
	max = n2;
	if (n3 > max)
	max = n3;

	printf("Maximum is: %d\n", max);
	getch();
}
```

```c
#include <stdio.h>
#include <conio.h>
int maximum(int x, int y, int z);
// function main begins program execution
int main(void)
{
      int n1, n2, n3;
      printf("%s", "Enter three integers: ");
      scanf("%d%d%d", &n1, &n2, &n3);
      printf("Maximum is: %d\n", maximum(n1, n2, n3));
      getch();
}
// Function maximum definition
int maximum(int x, int y, int z)
{
      int max = x;
      if (y > max)
      max = y;
      if (z > max)
      max = z;
      return max;
}
```

```c
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main(void)
{
float n, x, y;
printf("enter x,y\n");
scanf("%f%f", &x, &y);
printf("%.2f\n", pow(x, y));
getch();
}
```

# Cách trở về CT chính

- Không trả về giá trị chỉ sử dụng dấu } để kết thúc hàm
- Return 0;
- Return expression

# Các lỗi biên dịch

- Hàm nguyên mẫu không tương thích với lời gọi hàm
- Hàm nguyên mẫu và định nghĩa hàm không phù hợp
- Nếu lời gọi hàm trước hàm nguyên mẫu

# Ép kiểu đối số và quy tắc chuyển đổi

- Là tính năng quan trọng của hàm nguyên mẫu
- Ép đối số thành kiểu thích hợp trước khi hàm được gọi
- Trình biên dịch thực hiện
- VD:  sqrt (4) →kết quả 2.000
- Có thể dẫn đến kết quả không chính xác nếu không tuân thủ quy tắc
- Có 2 kiểu: ngầm định và tường minh

Ngầm định:

VD: double d = 13; → d = 13.0

Int  n = 16.5 → n = 16

Tường minh: theo ý người lập trình

VD: Cho biết kết quả in ra màn hình

```
printf ("%f\n", float(10 / 4));
printf("%f\n", (float(10) / 4));
```

# Ép kiểu đối số và quy tắc chuyển đổi

- Quy tắc chuyển đổi: chuyển thành dạng cao hơn. Nếu làm ngược lại sẽ gây ra lỗi
- Các kiểu dữ liệu từ thấp đến cao:

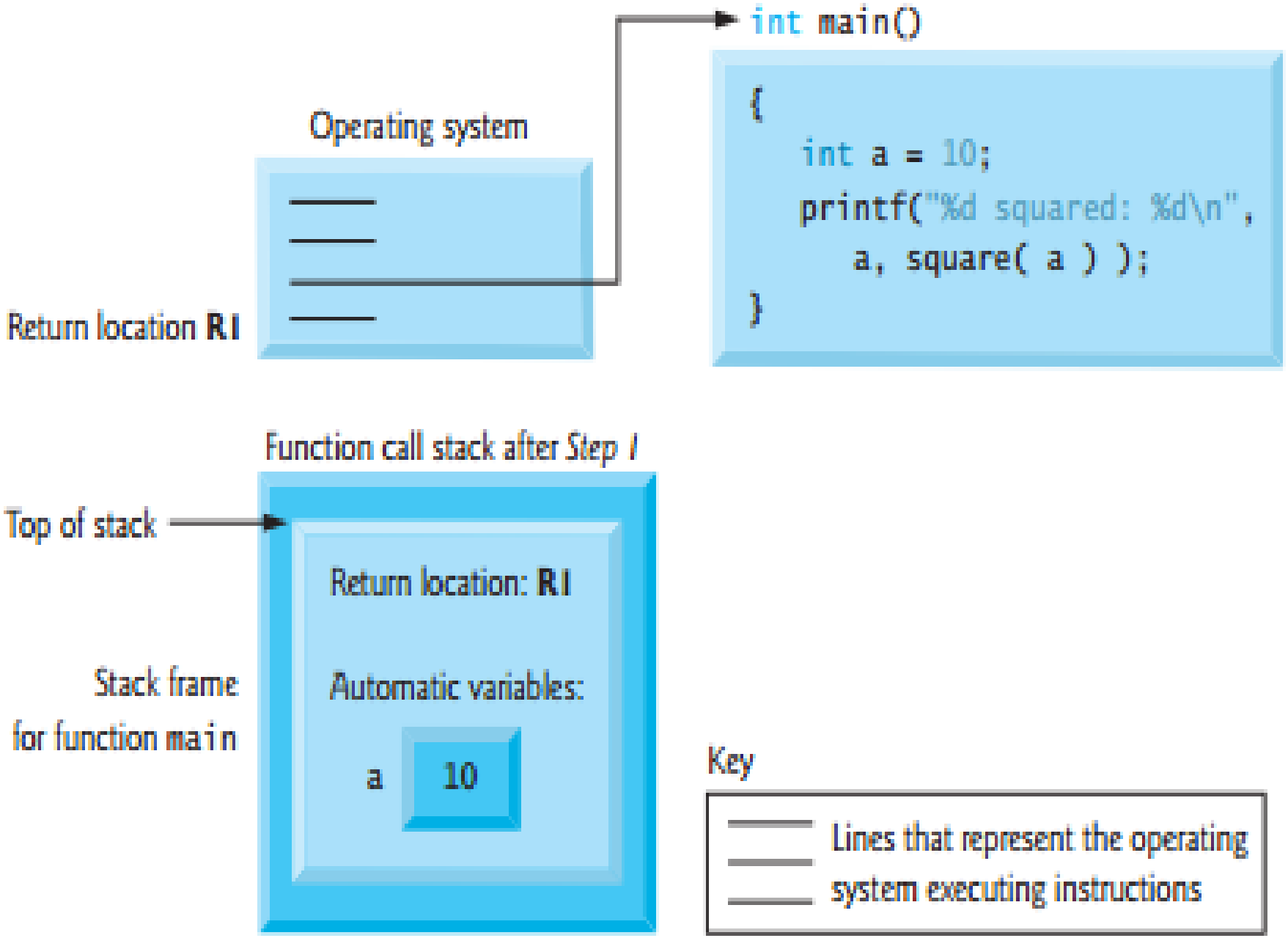
Int – unsigned int – long int – unsigned long int – float – double – long double

# 7. Ngăn xếp gọi hàm và khung ngăn xếp

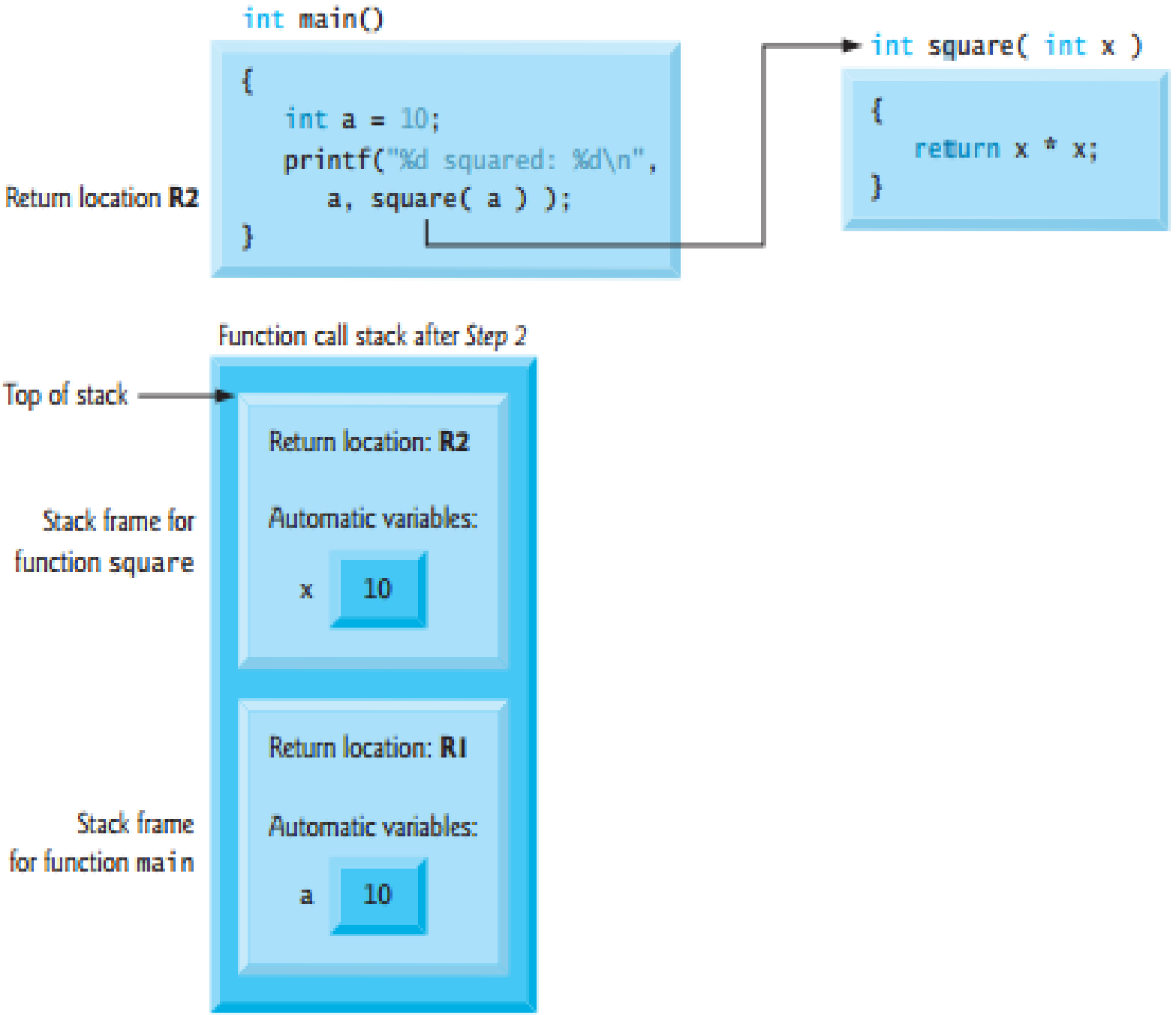
- Cấu trúc ngăn xếp như 1 chồng đĩa LIFO (last in – first out)
- VD

```
#include <stdio.h>
#include <conio.h>
int square(int);
int main(void)
{    int n = 10;
     printf("%d square is: %d\n", n,square (n));
     getch(); }
int square(int x)
{return x*x;}
```

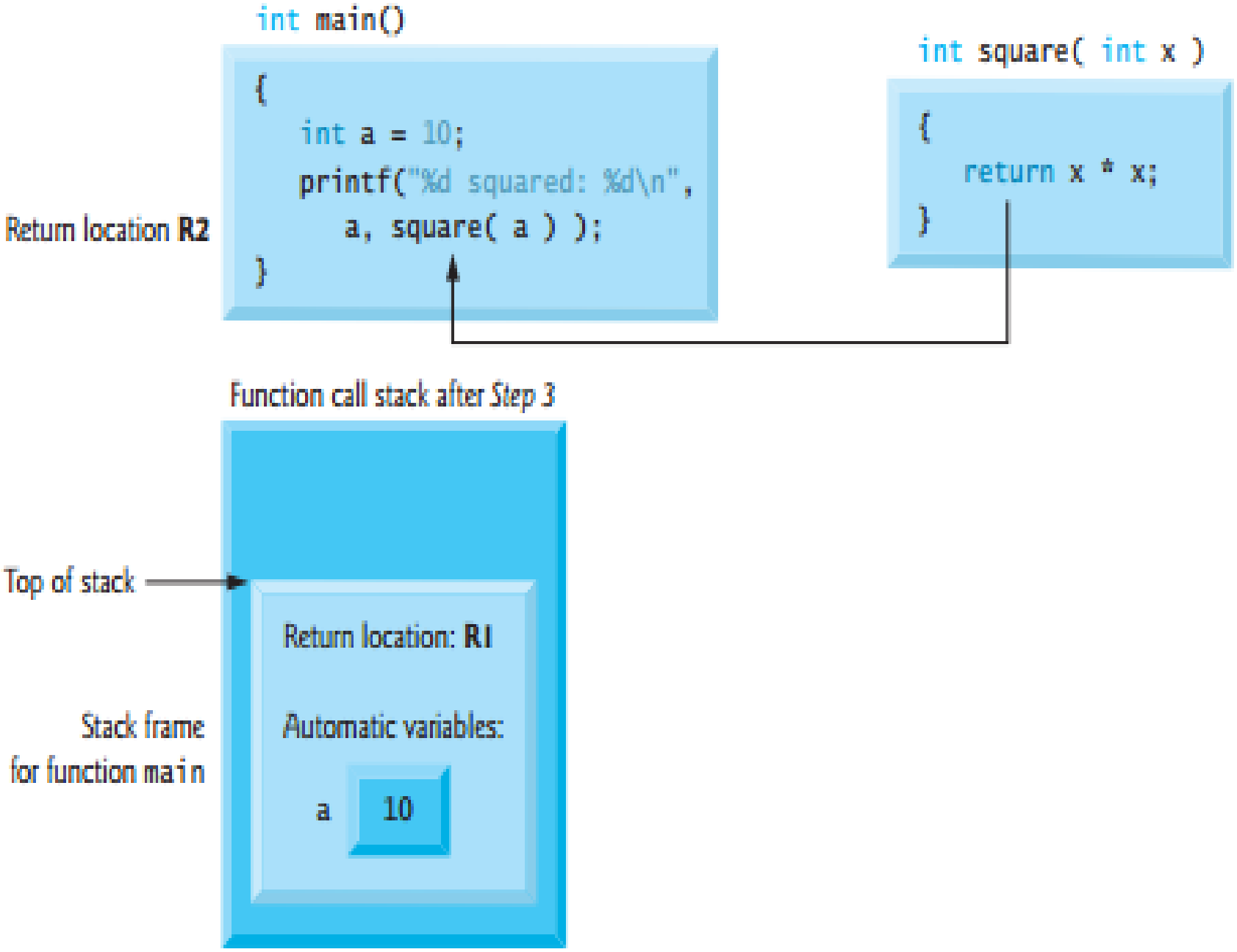
*Step 1*: Operating system invokes `main` to execute application



*Step 2:* `main` invokes function `square` to perform calculation



*Step 3*: square returns its result to main



# 8. Header

- Các file header: Chứa các hàm nguyên mẫu cho các hàm thư viện
- VD: stdio.h, math.h, conio.h....
- Khi load thì sử dụng:  #include “filename.h”
- Tạo file header:

  Header file/ add/ new item

  Lưu với tên filename.h

  Load file:  #include “filename.h”

  Sử dụng lại
- Các header trong thư viện tiêu chuẩn

- <assert.h>: chứa hỗ trợ gỡ lỗi
- <ctype.h>: chứa các hàm nguyên mẫu cho các hàm kiểm tra ký tự, chuyển chữ thường sang hoa và ngược lại
- <errno.h> Dùng để thử (hiển thị) các lỗi được báo cáo từ các hàm thư viện
- <float.h> ĐN các hằng nêu ra các đặc tính của thư viện chấm động
- <limits.h>ĐN các hằng có đặc tính đặc biệt của các kiểu nguyên
- <locale.h> Dùng cho setlocale() và các hằng có liên quan
- <math.h> tính các hàm số thông dụng
- <setjmp.h> Khai báo setjump/longjump được dùng trong việc thoát
- <signal.h> Để kiểm soát các điều kiện phát sinh trong thực hiện CT
- <stdarg.h> Để truy cập số lượng khác nhau của các đối số được chuyển vào hàm
- <stddef.h> ĐN các kiểu thông dụng để thực hiện tính toán
- <stdio.h> Các hàm thư viện vào ra
- <stdlib.h>chuyển đổi số sang văn bản và văn bản thành số, phân bổ bộ nhớ, số ngẫu nhiên, và các chức năng tiện ích khác
- <string.h> Để điều chỉnh nhiều loại dãy ký tự
- <time.h> chuyển đổi giữa các định dạng khác nhau về thì giờ và ngày tháng

# VD: Tạo và sử dụng 1 header tính bình phương, tìm max của 3 số

```
#include <stdio.h>
#include <conio.h>
#include "myheader.h"
int square(int);
void main()
{
int a;
printf("nhap a: ");
scanf("%d", &a);
printf("%d squared: %d\n", a, square(a));
getch();
}
```

```c
#include <stdio.h>
#include <conio.h>
#include "myheader.h"
void main()
{
int a,b,c,max;
printf("nhap a, b, c: ");
scanf_s("%d%d%d", &a,&b,&c);
max = maximum(a, b, c);
printf("max : %d\n", max);
printf("%d squared: %d\n", max, square(max));
_getch();
}
```

# Trong myheader.h

```
int square(int x)
{
      return x * x;
}

int maximum(int x, int y, int z)
{
      int max = x;
      if (y > max)
      max = y;
      if (z > max)
      max = z;
      return max;
}
```

```c
#include <conio.h>
#include "myheader.h"
void main(){
    int a;
    char kytu;

    printf("nhap ky tu: \n");
    kytu = getchar();
    printf("\nnhap a: \n");
    scanf_s("%d", &a);
    hinh_vuong_dac_1(a, kytu);
    _getch();
}
```

```
void hinh_vuong_dac_1(int n, char a){
    int i, j;
    for (i = 1; i <= n; i++){
        for (j = 1; j <= n; j++){
            printf("%c",a);
        }
        printf("\n");
    }
}
```

# BT

- Viết CT sử dụng hàm in ra hình vuông rỗng cạnh n hình * với n nhập từ bàn phím
- Viết CT sử dụng hàm in ra hình vuông rỗng cạnh n hình bất kỳ với n nhập từ bàn phím

# 9. Truyền tham chiếu và tham trị

- Tham số: Là những biến được khai báo
- Đối số: là giá trị được truyền vào hàm
- Truyền tham trị:

  Bản sao đối số được tạo ra và chuyển đến hàm được gọi, truyền dữ liệu

  Không làm thay đổi biến ban đầu

  Sử dụng khi hàm không cần sửa lại biến ban đầu
- Truyền tham chiếu

  Truyền bản thân đối số, truyền địa chỉ

  Thay đổi biến ban đầu

  Chỉ nên sử dụng hàm khi cần sửa lại biến ban đầu
- → Nên sử dụng truyền tham trị ngăn ngừa việc thay đổi biến nhiều lần

# VD:

- Void hoanvi (int a, int b) → tham trị (1)
- Void hoanvi (int &a, int &b) → tham chiếu (2)
- Cho x = 2, y = 3 (1) → sau hoán vị x = 2, y = 3

  (2)→ sau hoán vị x = 3, y = 2
- VD: Hàm tăng 1 cho 2 trường hợp

```
void hoanvi (int a, int b){
    int t = a;
    a = b;
    b = t;
}
int main (void){
  int x = 2, y = 3;
  hoanvi(x,y);
  printf("x = %d, y = %d",x,y);
}
```

```c
#include <stdio.h>
#include <conio.h>
int increment0(int a);
int increment1(int &a);
int main(void){
int n;
printf("enter n\n");
scanf("%d", &n);
printf("n= %d\n", n);
printf("n0= %d\n", increment0(n));
printf("n= %d\n", n);

printf("\nn= %d\n", n);
printf("n1= %d\n", increment1(n));
printf("n= %d\n", n);
getch();}
int increment0(int a)
{a = a + 1;
return a;}
int increment1(int &b)
{b = b + 1;
return b;}
```

```
#include <stdio.h>
#include <conio.h>
void ham2(int &a, int b);
int main(void){
    int x = 4, y = 3;
    ham2(x, y);
    printf("x= %d, y = %d\n", x, y);
}

void ham2(int &a, int b){
    if (a>b) {a = b; b++;}
    else a = a*a;
}
```

- Định nghĩa hàm mũ 3 và hàm a^b (truyền tham chiếu)
- Viết 1 CTC cho nhập vào 2 số nguyên x, y. Tính và in ra lập phương của x sử dụng hàm. Tính và in ra x^y sử dụng hàm.

# 10. Hàm lấy số ngẫu nhiên

- Cấu trúc I = rand ();
- Trả giá trị ngẫu nhiên từ 0 → max (ít nhất 32762)
- Trong thư viện <stdlib.h>
- Dãy giá trị thường khác nhau với 1 ứng dụng cụ thể
- VD: Giá trị ngẫu nhiên xúc xắc có giá trị từ 1 → 6
- Cấu trúc lệnh lấy ngẫu nhiên từ 1 đến n

  1 + (rand()%n)
- Trong đó (rand()%n) trả lại 1 số trong đoạn từ 0 → n-1
- VD: Hiển thị 20 số ngẫu nhiên từ 1 →6, mỗi hàng có 5 số

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main(void)
{
int i;
for (i = 1; i <= 20; i++)
{
/* pick random number from 1 to 6 and output it */
     printf("%10d", 1 + (rand() % 6));
/* if counter is divisible by 5, begin new line of output */
if (i % 5 == 0)
{ printf("\n");   }
}
getch();}
```

# Nhận xét

- Kết quả dãy số ngẫu nhiên không thay đổi nếu chạy lại CT
- Điều chỉnh CT để lấy dãy số ngẫu nhiên từ 100 → 105
- Làm sao biết dãy số trên là ngẫu nhiên???? Tăng số lần thực hiện và đếm số lần xuất hiện mỗi giá trị.

VD 6 ngàn lần

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main(void)
{
int f1 = 0, f2 = 0, f3 = 0, f4 = 0, f5 = 0, f6 = 0;
int roll, face;
for (roll = 1; roll <= 6000; roll++)
{
face = 1 + rand() % 6;
switch (face)
{ case 1:++f1;
break;
case 2: ++f2;
break;
```

```
case 3: ++f3;
break;
case 4: ++f4;
break;
case 5: ++f5;
break;
case 6: ++f6;
break;}}
printf("%s%13s\n", "Face", "Frequency");
printf("   1%13d\n", frequency1);
printf("   2%13d\n", frequency2);
printf("   3%13d\n", frequency3);
printf("   4%13d\n", frequency4);
printf("   5%13d\n", frequency5);
printf("   6%13d\n", frequency6);
getch();
return 0;
}
```

# 10. Hàm lấy số ngẫu nhiên

- Hàm: srand(seed);
- Chuỗi số ngẫu nhiên thay đổi khi thay đổi seed.

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main(void)
{
	int i;
	unsigned seed;
	printf("Enter seed: ");
	scanf("%u", &seed);
	srand(seed);
	for (i = 1; i <= 10; i++)
	{
		printf("%10d", 1 + (rand() % 6));
		if (i % 5 == 0)
			{printf("\n");}
	}
getch();
}
```

# 10. Hàm lấy số ngẫu nhiên

- Để sử dụng chuỗi số ngẫu nhiên thay đổi mà không cần thay đổi seed thì sử dụng:

```
srand(time(NULL));
```

- Trong đó time (NULL) trong thư viện <time.h>

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
int main(void)
{
	int i;
	srand((unsigned)time(NULL));
	for (i = 1; i <= 10; i++)
	{
		printf("%10d", 1+ rand() % 6);
		if (i % 5 == 0)
			{printf("\n");}
	}
	getch();
}
```

# 11. Đệ quy

- Hàm đệ quy là hàm có thể gọi chính nó
- Chỉ xét khái niệm cơ bản và 1 số CT có chứa hàm đệ quy
- Một hàm đệ quy được gọi để giải quyết vấn đề, trong đó hàm đã giải quyết được vấn đề cơ bản nhất
- Nếu 1 hàm được gọi với vấn đề cơ bản nhất thì trả về 1 giá trị
- Nếu hàm được gọi với vấn đề phức tạp hơn thì chia làm 2 mảng:

  Đã biết cách giải (cơ bản)

  Chưa biết cách giải
- Đối với phần chưa biết cách giải (tương tự như vấn đề ban đầu), hàm sẽ gọi bản sao mới của chính nó để làm việc với vấn đề nhỏ hơn (1 bước đệ quy) và tương tự đến khi kết thúc

# 11. Đệ quy

- VD: Giai thừa

  5! = 5*4*3*2*1

  = 5*4!

  = 5*4*3!

  = .... → tính bằng PP đệ quy

- Chú ý: 1! = 0! = 1
- Giải quyết vấn đề từ cơ bản nhất
- 1! =1 → 2! = 2*1! = 2*1 = 2

  3! = 3*2! = 3*2 = 6

  4! = 4*3! = 4*6 = 24

  .....

# VD. Tính và in ra giai thừa của 10 số tự nhiên đầu tiên

```
#include <stdio.h>
#include <conio.h>
long factorial(long number);
int main(void)
{
      int i;
      for (i = 0; i <= 10; i++)
      {printf("%2d! = %ld\n", i, factorial(i));}
      getch();
}
long factorial(long number)
{
      if (number <= 1)
            {return 1;}
      else
          {return (number * factorial(number - 1));}
}
```

# VD. Chuỗi Fibonacci

- Mỗi số là tổng 2 số trước. VD: 0,1,1,2,3,5,8,13....
- fib(n) = fib (n-1)+fib(n-2)
- Code:

```
long fibonacci (long n)
        { if (( n ==0)||(n==1))
                return n;
        else
                return (fibonacci (n-1)+fibonacci (n-2)
        }
```

```c
#include <stdio.h>
#include <conio.h>
long fibonacci(long n);
int main(void)
{
	long result, number;
	printf("Enter an integer: ");
	scanf("%ld", &number);
	result = fibonacci(number);
	printf("Fibonacci( %ld ) = %ld\n", number, result);
	getch();
}
long fibonacci(long n)
{
	if (n == 0 || n == 1)
		{return n;}
	else
		{return fibonacci(n - 1) + fibonacci(n - 2);}
}
```

# BT. In chuỗi số fibonacci với n<20

```
#include <stdio.h>
#include <conio.h>
long fibonacci(long n);
int main(void)
{
        long result, number;
        printf("Enter an integer: ");
        scanf("%ld", &number);
        for (i = 0; i < number; i++)
        {       result = fibonacci(i);
                printf("Fibonacci( %ld ) = %ld\n", i, result);}
        getch();
}
long fibonacci(long n)
{
        if (n == 0 || n == 1)
                {return n;}
        else
                {return fibonacci(n - 1) + fibonacci(n - 2);}
}
```

# 11. Đệ quy

- Nhược  điểm: liên tục gọi nên tốn bộ nhớ và thời gian xử lý
- Bất kỳ vấn đề nào xử lý đệ quy cũng có thể xử lý lặp hoặc lặp không rõ ràng

# 12. Đệ quy và vòng lặp

- Cấu trúc sử dụng:
  - Vòng lặp: sử dụng cấu trúc lặp
  - Đệ quy: sử dụng cấu trúc lựa chọn
- Lệnh sử dụng:
  - Vòng lặp sử dụng lệnh lặp
  - Đệ quy sử dụng gọi hàm lặp
- Thoát lặp:
  - Điều kiện vòng lặp không thỏa
  - Trường hợp cơ bản được nhận dạng
- Cả hai đều có thể xảy ra lặp vô hạn

# Bài tập

- Viết CT có sử dụng hàm hoán đổi vị trí
- Viết chương trình tính chu vi, diện tích hình tròn sử dụng định nghĩa số Pi, hàm diện tích và hàm chu vi trong thư viện tự tạo
- Viết CT kiểm tra số chẵn lẻ sử dụng hàm và sử dụng biến toàn cục
- Viết hàm kiểm tra số nguyên tố
- Viết hàm in ra các ước số
- Viết hàm kiểm tra năm nhuần

# Bài tập

- Viết hàm in ra số nguyên tố nhỏ hơn N
- Viết hàm tính x^n dùng đệ quy
- Viết CT tìm ước chung lớn nhất dùng đệ quy đệ quy và cách thông thường

```
void main(void)
{
int n, a;
do {
guess();
printf("play again (y/n)?");
scanf_s("%c", &a);
a = getchar();
}
while (a  == 'y');
_getch();
}
```

```
unsigned int mystery(unsigned int a, unsigned int b)
{

        if (1 == b) {
                return a;
        }
        else {
                 return a + mystery(a, b - 1);

        }
 }
```