Chương 2. Giới thiệu viết CT C

1. Viết chương trình đơn giản trong C
2. Sử dụng lệnh xuất nhập đơn giản
3. Các kiểu dữ liệu cơ bản
4. Khái niệm bộ nhớ máy tính
5. Sử dụng các biểu thức số học
6. Sự ưu tiên của các phép toán số học
7. Viết các lệnh quyết định đơn giản

Giới thiệu

- Ngôn ngữ lập trình C
  - Chặt chẽ
  - Có cấu trúc
- Cấu trúc chương trình C
  - Giới thiệu trong chương sau

1. Viết CT đơn giản trong C
In 1 dòng chữ

```c
/* Fig. 2.1: fig02_01.c
   A first program in C */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome to C!\n" );

   return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
Welcome to C!
```

1. Viết CT đơn giản trong C

- Chú thích:
  - Có 2 cách chú thích: // hoặc /* */
  - Trình biên dịch sẽ bỏ qua phần chú thích
  - Dùng để chú thích câu lệnh, đoạn lệnh, chương trình
- # include <stdio.h>
  - Khai báo tiền xử lý
  - Bắt đầu bằng #
  - Cho phép các lệnh input/output

1. Viết CT đơn giản trong C

- `int main()`
  - Chương trình C chứa một hoặc hơn một hàm
  - Dấu `( )` được sử dụng để chỉ 1 hàm
  - `int` có nghĩa là hàm `main` trả về giá trị số nguyên
  - Dấu `{ }` để chỉ một khối lệnh. Thân của các hàm phải được đặt trong dấu `{ }`

→ `int main (void)`

1. Viết CT đơn giản trong C

- `printf( "Welcome to C!\n" );`
  - Tên hàm: printf
  - Chức năng: In ký tự trong dấu ” ”
  - Cấu trúc lệnh:

  printf (“chuỗi định dạng”,[đối mục 1, đối mục 2,..]);

Đối mục: Mục dữ liệu cần in ra gồm: Biến, hằng, biểu thức

Chuỗi định dạng gồm 3 loại: Chuỗi ký tự (ký tự được in ra); mã định dạng; ký tự điều khiển

1. Viết CT đơn giản trong C

- Bảng mã định dạng

%c : Ký tự đơn

%s : Chuỗi

%d : Số nguyên thập phân có dấu

%f : Số chấm động (ký hiệu thập phân)

%e : ký hiệu có số mũ

%g : Số chấm động (%f hay %g)

%x : Số nguyên hex không dấu

%u : Số nguyên thập phân không dấu

%o : Số nguyên bát phân không dấu

l : Tiền tố dùng kèm với %d, %u, %x, %o để chỉ số nguyên dài (ví dụ %ld)

1. Viết CT đơn giản trong C

- Ký tự điều khiển và ký tự đặc biệt

\n : Nhảy xuống dòng kế tiếp canh về cột đầu tiên

\t : Canh cột tab ngang.

\r : Nhảy về đầu hàng, không xuống hàng.

\a : Tiếng kêu bip.

\\ : In ra dấu \

\" : In ra dấu "

\' : In ra dấu '

%%: In ra dấu %

\b ~ backspace (xóa 1 ký tự ngay trước)

1. Viết CT đơn giản trong C

- `return 0;`
  - Một cách để thoát một hàm
  - Trong trường hợp này còn có nghĩa là đã đạt kết quả
- Right brace }
  - Kết thúc chương trình chính
- Linker
  - Khi 1 CT được gọi, Linker xác định vị trí trong bộ nhớ
  - Chèn vào
  - Nếu tên hàm sai, linker sẽ báo lỗi vì không tìm được hàm

```c
// Display a sentence
#include <stdio.h>
#include <conio.h>
// function main begins program execution
int main(void){
   printf("Welcome to C!\n");
   _getch();
} // end function main
```

```c
// Display a string.
#include <stdio.h>
#include <conio.h>
// function main begins program execution
int main(void){
   printf("Welcome ");
   printf("to C!\n");
   getch();
} // end function main
```

Sử dụng nhiều lệnh printf

```
1  /* Fig. 2.3: fig02_03.c
2     Printing on one line with two printf statements */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     printf( "Welcome " );
9     printf( "to C!\n" );
10
11    return 0; /* indicate that program ended successfully */
12
13 } /* end function main */
```

```c
/* Fig. 2.4: fig02_04.c
   Printing multiple lines with a single printf */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome\nto\nC!\n" );

   return 0; /* indicate that program ended successfully */

} /* end function main */
```

Newline characters move the cursor to the next line

```
Welcome
to
C!
```

Exercise

1. Tìm lỗi trong các câu lệnh, đoạn lệnh sau

a. scanf("%d%d", number1, number2);

b. print("The value is %d\n", &number);

c.
```
if ( c < 7 );
    {
    printf( "C is less than 7\n" );
    }
```

d.
```
if ( c =< 7 )
    {
    prinf( "C is less than 7\n" )
    }
```

Exercise

2. Cho biết kết quả hiển thị của các câu lệnh sau:

    a. printf("This is a C program.\n\\");
    b. printf("This is a C\nprogram.\n");
    c. printf("This\nis\na\nC\nprogram.\n");
    d. printf("This\tis\ta\tC\tprogram.\n");

3. Viết chương trình hiển thị dòng chữ sau:

a. I want to

    Learn C

b. I

    want

    to

    learn

    C

Exercise

4. Tìm lỗi, sửa lỗi CT sau để hiển thị kết quả như sau:
Dau tien. Bai hoc C

```
#include <stdio.h>
main(void) {
   print ("Bai hoc C \nDau tien.\n)
   getch;
```

5. Viết CT hiển thị các thông tin sau: Tên trường, tên khoa, tên ngành, họ và tên, MSSV.
Mỗi thông tin 1 dòng

1. Write a C program to print your name, date of birth. and mobile number.
Expected Output:

```
Name   : Alexandra Abramov

DOB    : July 14, 1975

Mobile : 99-9999999999
```

2. Viết CT đơn giản trong C

Hiển thị số nhập từ bàn phím

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int integer1;
    printf(“Enter an integer: ");
    scanf("%d", &integer1);
    printf(“entered number is : %d.\n", integer1);
    getch();
}
```

Khai báo biến:

<Kiểu dữ liệu> <Tên biến> = [Giá trị khởi tạo của biến];

int integer1;
int là kiểu dữ liệu integer; integer1 là tên biến

Nếu có nhiều biến cùng kiểu dữ liệu

```
int integer1, integer2;
```

Hoặc

```
int integer1;
int integer2;
```

Khi khai báo biến có thể gắn giá trị khởi tạo hoặc không:

```
int integer1 = 0;
```

Lưu ý khi đặt tên biến:

- Dùng ký tự và số (không bắt đầu bằng số) và “_”
- Phân biệt chữ hoa và thường
- Nên đặt tên biến có nghĩa, gợi nhớ
- Khai báo trước khi sử dụng lệnh có chứa biến đó
- Không quá 31 ký tự
- Không được sử dụng keywords của C

Lệnh

scanf("%d", &integer1);

Đọc ngõ vào tiêu chuẩn (từ bàn phím)

%d: chuỗi điều khiển định dạng_chỉ loại dữ liệu người sử dụng nhập vào

&integer1: định vị trí trong bộ nhớ để lưu biến, đọc giá trị, hay đem giá trị gõ từ bàn phím lưu vào biến

printf(“entered number is : %d.\n", integer1);

Lệnh

printf(“entered number is : %d.\n", integer1);

%d: Số nguyên có dấu được in
integer1 : Giá trị số nguyên được in
Đối mục: tại vị trí integer1 có thể là biến, hằng hay biểu thức

Viết 01 CTC nhập vào 2 số nguyên a,b. In ra tổng, hiệu, tích, thương của 2 số vừa nhập

Viết 01 CTC nhập vào bán kính. Tính và in ra chu vi và diện tích hình tròn

```c
// Addition program.
#include <stdio.h>
#include <conio.h>
// function main begins program execution
int main(void)
{
int integer1; // first number to be entered by user
int integer2; // second number to be entered by user
int sum; // variable in which sum will be stored

printf("Enter first integer\n"); // prompt
scanf("%d", &integer1); // read an integer

printf("Enter first integer\n"); // prompt
scanf("%d", &integer2); // read an integer

sum = integer1 + integer2; // assign total to sum
printf("Sum is %d\n", sum); // print sum
getch();
} // end function main
```

3. Khái niệm bộ nhớ

- Các tên biến ứng với các vị trí trong ô nhớ
- Mỗi biến có tên, loại dữ liệu, giá trị, kích thước
- Khi có một giá trị mới đặt vào thì nó thay thế, làm mất giá trị cũ
- Đọc biến từ ô nhớ không làm thay đổi chúng

4. Các kiểu dữ liệu cơ bản

Có các kiểu dữ liệu cơ bản sau:

Bool (boolean): true hoặc false

Char (character): Ký tự

Int (integer): Số nguyên

Float, Double: Dấu chấm động

4. Các kiểu dữ liệu cơ bản

Kiểu	Cỡ lưu trữ	Dãy giá trị
char	1 byte	-128 tới 127 hoặc 0 tới 255
unsigned char	1 byte	0 tới 255
signed char	1 byte	-128 tới 127
int	2 hoặc 4 bytes	-32,768 tới 32,767 hoặc -2,147,483,648 tới 2,147,483,647
unsigned int	2 hoặc 4 bytes	0 tới 65,535 hoặc 0 tới 4,294,967,295
short	2 bytes	-32,768 tới 32,767
unsigned short	2 bytes	0 tới 65,535
long	4 bytes	-2,147,483,648 tới 2,147,483,647
unsigned long	4 bytes	0 tới 4,294,967,295

4. Các kiểu dữ liệu cơ bản

Kiểu	Cỡ lưu trữ	Dãy giá trị	Độ chính xác
float	4 byte	1.2E-38 tới 3.4E+38	6 vị trí thập phân
double	8 byte	2.3E-308 tới 1.7E+308	15 vị trí thập phân
long double	10 byte	3.4E-4932 tới 1.1E+4932	19 vị trí thập phân

5. Các từ khóa của C

Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

```c
#include <stdio.h>
#include <conio.h>
// function main begins program execution
int main(void)
{
int a=10, b=15, c=1234;
float m = 10.05, n = 0.1234, p = 0.2;
printf("Tong cua 2 so %d va %d la %d.\n", a, b, a+b);
printf("Tong cua 2 so %d va %d la %7d.\n", a, b, a + b);
printf("Tong cua 2 so %5d va %3d la %1d . \n", a, b, a +
b);
printf("Tong cua 2 so %-5d va %-3d la %-1d . \n", a, b, a +
b);
printf("Tong cua 2 so %02d va %02d la %04d . \n", a, b, a +
b);
printf("%7d%7d%7d.\n", a, b, c);
printf("%7d%7d%7d.\n", 165, 2, 965);
printf("%-7d%-7d%-7d.\n", a, b, c);
printf("%-7d%-7d%-7d.\n", 165, 2, 965);
printf("%7.2f%7.2f%7.2f.\n", m, n, p);
getch();
}
```

```
printf("Tong cua 2 so %d va %d la %d.\n", a, b, a+b);
```

```
Tong cua 2 so 10 va 15 la 25.
```

```
printf("Tong cua 2 so %d va %d la %7d.\n", a, b, a + b);
```

```
Tong cua 2 so 10 va 15 la      25.
```

```
printf("Tong cua 2 so %5d va %3d la %1d . \n", a, b, a + b)
```

```
Tong cua 2 so    10 va  15 la 25 .
```

```
printf("Tong cua 2 so %-5d va %-3d la %-1d . \n", a, b, a + b)
```

```
Tong cua 2 so 10    va 15  la 25 .
```

```
printf("Tong cua 2 so %02d va %02d la %04d . \n", a, b, a + b)
```

```
Tong cua 2 so 10 va 15 la 0025 .
```

```
printf("%7d%7d%7d.\n", a, b, c)
```

```
     10     15   1234.
```

```
printf("%7.2f%7.2f%7.2f.\n", m, n, p)
```

```
  10.05   0.12   0.20.
```

6. Các phép toán số học trong C

C opetration	Arithmetic operator	Algebraic expression	C expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x/y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

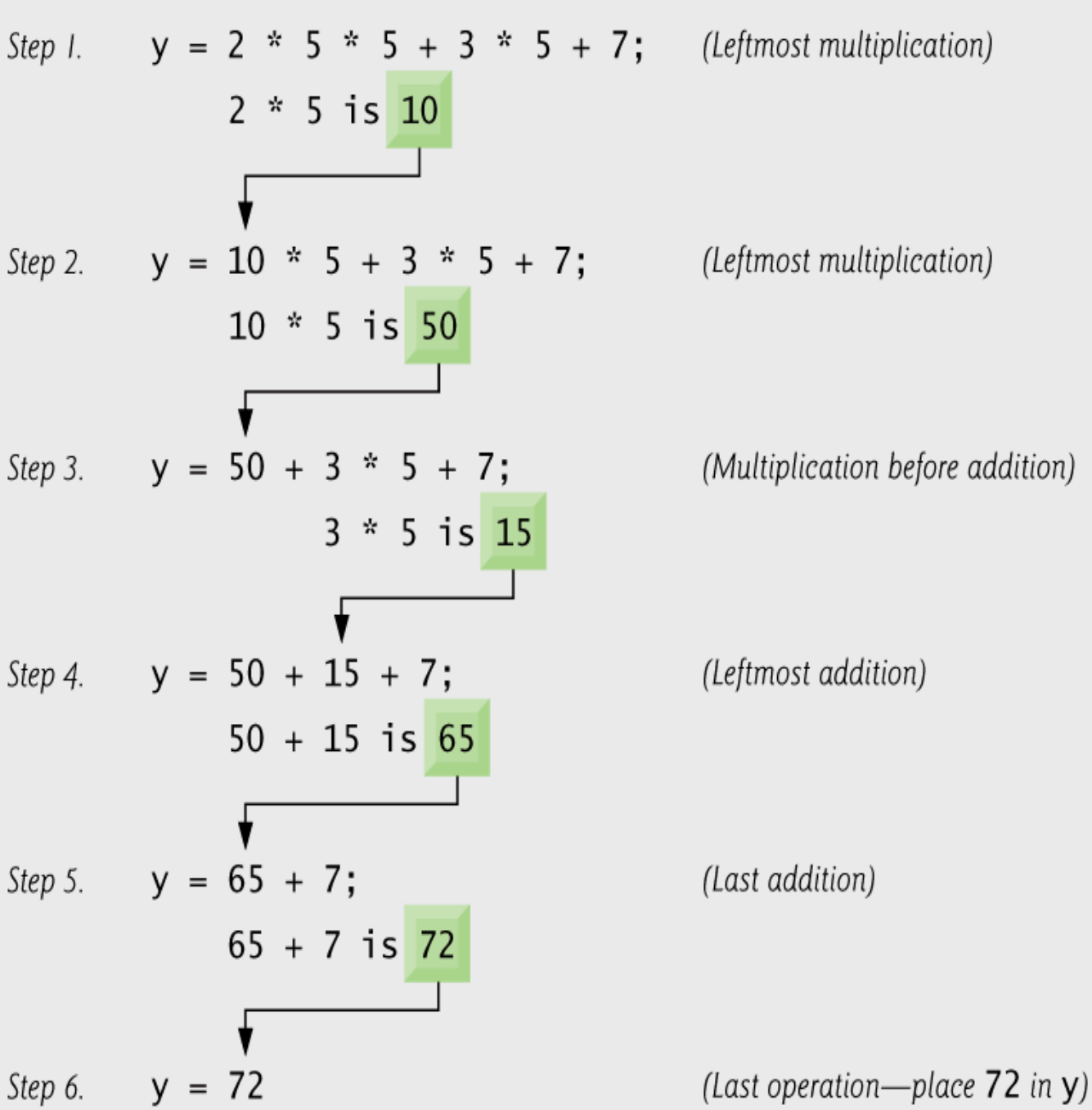
Thứ tự ưu tiên thực hiện phép toán trong C

1. ( )
2. * / %
3. + -
4. =

VD: Cho biết thứ tự thực hiện biểu thức sau:

Z = p * r % q + w/x - y

Y = a * x * x + b * x + c


Step 1.
y = 2 * 5 * 5 + 3 * 5 + 7;
(Leftmost multiplication)
2 * 5 is 10
Step 2.
y = 10 * 5 + 3 * 5 + 7;
(Leftmost multiplication)
10 * 5 is 50
Step 3.
y = 50 + 3 * 5 + 7;
(Multiplication before addition)
3 * 5 is 15
Step 4.
y = 50 + 15 + 7;
(Leftmost addition)
50 + 15 is 65
Step 5.
y = 65 + 7;
(Last addition)
65 + 7 is 72
Step 6.
y = 72
(Last operation—place 72 in y)

7. Quyết định

Đẳng thức

== : bằng (=) VD: x==y (nghĩa là x = y)

!= : Khác (#) VD: x!=y (nghĩa là x # y)

Biểu thức quan hệ

Ký hiệu	Ý nghĩa	VD
>	Lớn hơn	x > y
<	Nhỏ hơn	x < y
>=	Lớn hơn hoặc bằng	x >= y
<=	Nhỏ hơn hoặc bằng	x <= y

Nhập điểm. Cho biết SV đó đậu hay rớt biết điểm >=5: đậu. Nếu rớt thì in thêm “hoc lai”

```
int diem;
scanf (“%d”,&diem);
if (diem >=5)
    printf (“dau”);    // printf (“\nchuc mung”);
if (diem <5)
    printf (“rot”);
if (đk)
    lệnh;
if (đk) {
    Khối lệnh
}
```

7. Quyết định

```
#include <stdio.h>
#include <conio.h>
// function main begins program execution
int main(void){
    float grade;
    printf("Enter your grade:\n");
    scanf("%f", &grade);
    if (grade >= 5)
       printf("passed\n");
       printf("Congrats!\n");

    if (grade < 5)
       printf("failed\n");
       printf("Take your course again!\n");

    getch();
}
```

```c
#include <stdio.h>
#include <conio.h>
// function main begins program execution
int main(void)
    {
     int num1; // first number to be read from user
     int num2; // second number to be read from user
     printf("Enter two integers, and I will tell you\n");
     printf("the relationships they satisfy: ");
     scanf("%d%d", &num1, &num2); // read two integers
```

```
if (num1 == num2) {
    printf("%d is equal to %d\n", num1, num2);
    }
if (num1 != num2) {
    printf("%d is not equal to %d\n", num1, num2);
    }
if (num1 < num2) {
    printf("%d is less than %d\n", num1, num2);
    }
if (num1 > num2) {
    printf("%d is greater than %d\n", num1, num2);
    }
if (num1 <= num2) {
    printf("%d is less than or equal to %d\n", num1, num2);
    }
if (num1 >= num2) {
    printf("%d is greater than or equal to %d\n", num1,
num2);
    }
getch();}
```

Chú ý

1. Nên sử dụng chú thích (// được khuyến khích hơn)
2. Nên đặt tên biến có tính gợi nhớ
3. Thêm các khoảng trắng để chương trình dễ đọc
4. Trước khi viết hàm nên có mô tả chức năng của hàm
5. Viết lùi vào cùng số ký tự khoảng trắng cho các hàm, lệnh cùng cấp (3 khoảng trắng cho 1 cấp)
6. Đặt 1 khoảng trắng sau mỗi dấu , để dễ đọc
7. Đặt khoảng trắng trước và sau các toán tử
8. Biểu thức đặt bên phải của toán tử gán (=)
9. Nhớ đặt & trước biến trong hàm scanf

Các lỗi thông dụng

1. Thiếu dấu ; cuối mỗi lệnh
2. Viết sai từ khóa
3. Nhầm giữa 0 và o
4. Viết sai tên biến
5. Viết sai tên hàm
6. Nhầm giữa lệnh gán (=) và so sánh bằng (==)