BÀI 2
BIẾN, MẢNG VÀ BIỂU THỨC

MỤC TIÊU

Người học sau khi học xong bài 2 sẽ có các khái niệm cơ bản về các vấn đề sau:

- Các kiểu dữ liệu;
- Hằng và biến;
- Hàm số;
- Toán tử;
- Biểu thức;
- Tập ký tự trong ngôn ngữ C.

KIẾN THỨC CẦN CÓ

Các kiến thức cần thiết:

- Học xong bài 1.
- Biết phương pháp giải các bài toán cơ bản như phương trình bậc 2, hệ phương trình bậc 1.
- Khuyến nghị học môn Tin học cơ bản.

NỘI DUNG

1. Kiểu dữ liệu.
2. Hằng (CONSTANT).
3. Kiểu Enum.
4. Biến (VARIABLE).
5. Mảng (ARRAY).
6. Định nghĩa kiểu bằng typedef.
7. Khối lệnh.
8. Vài nét về hàm và chương trình.
9. Biến, mảng tự động.
10. Biến, mảng ngoài.
11. Toán tử size of.
12. Biến tĩnh, mảng tĩnh.
13. Biểu thức.
14. Phép toán số học.
15. Các thao tác BIT.
16. Phép toán so sánh và logic.
17. Chuyển đổi kiểu giá trị.
18. Phép toán tăng giảm.
19. Câu lệnh gán và biểu thức gán.
20. Biểu thức điều kiện.
21. Vài ví dụ.
22. Thứ tự ưu tiên của các phép toán.

1. KIỂU DỮ LIỆU

- Kiểu dữ liệu (data types) là một định dạng thông tin cho biến, hằng, mảng và các khai báo khác trong chương trình. Nó giúp cho chương trình phân cấp bộ nhớ hợp lý đồng thời phân loại và quản lý dữ liệu tốt hơn.
- Trong C có 4 loại kiểu dữ liệu chính:
  - Kiểu ký tự (char)
  - Kiểu số nguyên (int)
  - Kiểu số dấu phẩy động độ chính xác đơn (float)
  - Kiểu số dấu phẩy động độ chính xác kép (double)

1.1. KIỂU CHAR

- Kiểu char: Có 256 loại khác nhau, được quy định trong bảng mã ASCII, mỗi loại được xác định bởi 1 byte thông tin (8bit);
- Có 2 kiểu char là unsigned char và signed char:
  - Signed char: Biểu diễn 1 số nguyên có dấu, từ -128 đến 127.
  - Unsigned char: Biểu diễn 1 số nguyên không dấu, từ 0 đến 255.
- Hai kiểu char có sự khác nhau nhất định:

Ví dụ:

```
char x1;
unsigned char x2;
x1 = 200;
x2 = 200;
```

Với khai báo trên, thực chất thì:

x1 = 200-(127+1) +(-128) = -56

x2 = 200

1.2. KIỂU NGUYÊN

- Kiểu nguyên: Có 4 loại kiểu nguyên khác nhau:

Kiểu	Phạm vi	Kích thước
int	-32768 -> 32767	2 byte
unsigned int	0 -> 65535	2 byte
long	-2147483648 -> 2147483647	4 byte
unsigned long	0 -> 4294967295	4 byte

- Nếu xét về phạm vi biểu diễn, thì kiểu ký tự cũng được coi là một trường hợp đặc biệt của kiểu nguyên.

1.3. KIỂU DẤU PHẨY ĐỘNG

- Kiểu dấu phẩy động: Có 3 loại khác nhau

Kiểu	Phạm vi	Kích thước	Số chữ số có nghĩa
float	3.4E-38 ->3.4E38	4byte	7-8
double	1.8E-308 ->1.7E308	8byte	15-16
long double	3.4E-4932 ->1.1E4932	10byte	17-18

- Với mỗi kiểu trên, máy tính chỉ có thể lưu trữ được giá trị nằm trong các khoảng phạm vi. Ngoài khoảng thì máy tính coi là bằng 0 hoặc bằng ∞.

2. HẰNG (CONSTANT)

- Hằng (hằng số) là các đại lượng không thay đổi trong toàn bộ chương trình cũng như trong quá trình tính toán.
- Chương trình C có sử dụng các loại hằng số sau:
  - Hằng dấu phẩy động
  - Hằng int
  - Hằng long
  - Hằng int hệ 8
  - Hằng nguyên hệ 16
  - Hằng ký tự
  - Hằng xâu ký tự
  - Tên hằng

2. HẰNG (CONSTANT) (tiếp theo)

Hằng dấu phẩy động có 2 loại là float và double, có thể được viết theo 2 cách:

- Cách 1: Dạng thập phân gồm số nguyên (có thể ko có), dấu chấm và phần thập phân.

  Ví dụ:

214.35	.456	234.0

- Cách 2: Dạng khoa học (dạng mũ): Gồm phần định trị và phần bậc. Phần định trị là 1 số nguyên hoặc số thập phân, phần bậc là 1 số nguyên.

  Ví dụ:

123.4E-2 = 1.234	1E8 = 100000000.0

2. HẰNG (CONSTANT) (tiếptheo)

- Hằng int: Là 1 số nằm trong khoảng -32768 đến 32767

  Ví dụ:

  -56 +5632 8232

- Hằng long: Được xác định khi thêm chữ L hoặc l vào sau cùng.

  Ví dụ:

  -4672L hoặc 23324l

- Hằng int hệ 8: Được viết theo cách 0c1c2c3c…
  - Trong đó ci là 1 số nằm trong khoảng 0 đến 7.
  - Hằng int hệ 8 luôn mang dấu dương.
  - Muốn chuyển đổi hằng int hệ 8 sang hằng int hệ 10 ta dùng phương pháp chuyển đổi cơ số từ bát phân sang thập phân.

2. HẰNG (CONSTANT) (tiếp theo)

- Hằng nguyên hệ 16: Viết dưới dạng 0xc1c2c3c4….
- Trong đó:
  - 0x: tiền tố đầu của hệ 16
  - Hệ 16 có các số 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  - ci: là 1 trong các chữ hoặc số của hệ 16.

  Ví dụ:

0x67AF	0x3456

- Muốn chuyển hằng nguyên hệ 16 sang hằng int hệ 8 hay hệ 10 thì ta dùng các phương pháp chuyển đổi hệ cơ số thập lục phân, sang hệ cơ số bát phân hoặc thập phân.

2. HẰNG (CONSTANT) (tiếp theo)

- Hằng ký tự: Lấy vị trí của ký tự trong bảng mã ASCII làm giá trị cho hằng số. Ký tự phải được viết trong 2 dấu nháy đơn.

  Ví dụ:

```
'a' = 97
'a' – 7 = 90
```

- Ngoài ra, có thể định nghĩa hằng ký tự theo cách khác

  Ví dụ:

```
'\c1c2c3'
```

- Hằng xâu ký tự: Là 1 dãy ký tự bất kỳ được đặt trong 2 dấu ""

  Ví dụ:

```
"Hello"
"" /* hằng rỗng*/
```

2. HẰNG (CONSTANT) (tiếp theo)

Tên hằng: Là 1 kiểu khai báo hằng số cho toàn bộ chương trình C. Được đặt ở đầu chương trình:

Ví dụ 1:

```
#define Max 1000
```

Sau khai báo này, tất cả các biến Max được sử dụng trong chương trình sẽ có giá trị là 1000.

Ví dụ 2:

```
#define pi 3.1415926
```

Sau khai báo này, tất cả các biến pi được sử dụng trong chương trình sẽ có giá trị là 3,1415926

3. KIỂU ENUM

- Cú pháp:

```
enum tk {bt1,bt2,….} tb1,tb2…..;
enum tk {bt1,bt2…..};
enum {bt1,bt2,….} tb1,tb2…..;
enum {bt1,bt2…..}.
```

  - tk: Tên kiểu, có thể rỗng
  - bt1,bt2: Các phần tử
  - tb1,tb2: Là tên biến kiểu enum

- Tác dụng của kiểu enum:
  - Định nghĩa các macro
  - Khai báo hàng loạt các biến

3. KIỂU ENUM (tiếp theo)

```
#include  <stdio.h>
#include  <conio.h>
void main (void)
{
enum {T0.T1.T2};
enum day{cn,t2.t3.t4.t5.t6.t7} n1;
enum day n2;
int i,j = 2000,k = T2
clrscr();
i = 17;
n1= -1000;
n2 = j;
printf("\n n1 =%d n2 = %d i = %d ",n1,n2,i);
printf("\n k = %d T1 = %d",k,T1);
getch();
}
```

4. BIẾN (VARIABLE)

Biến cần phải được khai báo trước khi sử dụng.

Mẫu khai báo như sau:

```
type tên_biến;
```

- Type: Kiểu biến, tuân theo quy tắc và số lượng kiểu biến xác định trong c
- Tên_biến: Tuân theo quy tắc đặt tên và độ dài
- **;** kết thúc lệnh.

  Ví dụ:

```
int a,b,c;
long d,e,f;
double x,y,z;
```

4. BIẾN (VARIABLE) (tiếp theo)

- Khai báo biến phải được đặt ngay sau dấu { của chương trình chính

```
#include <stdio.h>
.............
void main(void)
{
/* Bắt đầu khai báo biến ở đây
}
```

- Có hai cách gán giá trị cho biến:
  - Gán trực tiếp giá trị cho biến

```
int a,b = 20, c, d= 10;
```

  - Gán giá trị gián tiếp cho biến

```
int a,b,c;
b = 20;
```

- Lấy vị trí của biến trong bộ nhớ dùng lệnh & tên_biến.

5. MẢNG (ARRAY)

- Mảng là tập hợp của các phần tử có cùng 1 kiểu giá trị và có chung 1 tên. Mỗi phần tử của mảng có vai trò như 1 biến và có thể chứa tối đa 1 giá trị.

- Thông tin cơ bản về mảng bao gồm:
  - Kiểu mảng
  - Tên mảng
  - Số chiều và kích thước của mỗi chiều → Số biến
  - Mảng 1 chiều và nhiều chiều.

25. MẢNG (ARRAY) (tiếp theo)

- Mảng a có:
  - Kiểu int
  - Tên là a
  - 1 chiều, kích thước 10

```
int a[10];
```

- Mảng b có:
  - Kiểu long
  - Tên b
  - Hai chiều, kích thước 2x4, số phần tử là 8

```
long b[2][4];
```

Ví dụ:

Về cấu trúc của mảng b và dữ liệu thành phần của chúng như sau:

b[0][0]	b[0][1]
b[1][0]	b[1][1]
b[2][0]	b[2][1]
b[3][0]	b[3][1]

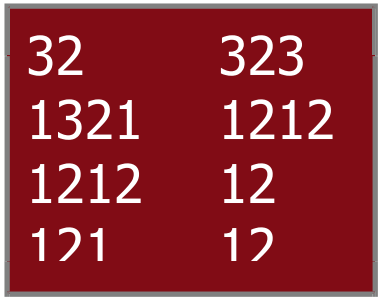
32	323
1321	1212
1212	12
121	12

5. MẢNG (ARRAY) (tiếp theo)

- Khi cần truy xuất vào các phần tử trong mảng, dùng các chỉ số, các chỉ số này là số thực, hoặc các biến động thay đổi theo chương trình.
- Chỉ số mảng phải nằm trong giới hạn của mảng

  Ví dụ: Mảng b[4][2], dùng thêm 2 chỉ số i và j để truy xuất vào mảng:

  - 0≤i ≤3;
  - 0≤j ≤2;
  - Giả sử i = 2, j = 1 → b[i][j] = 12.



- Có thể dùng lệnh & a[i] và & b[i][j] để lấy địa chỉ của phần tử trong mảng.

6. KIỂU BẢNG TYPEDEF

Ngoài các kiểu biến đã có sẵn, C cho phép người dùng tự định nghĩa 1 số kiểu biến khác:

Cấu trúc:

```
typedef tên_kiểu tên_kiểu_mới;
```

typedef: Từ khóa của lệnh

tên_kiểu: Các kiểu biến cũ, ví dụ int, long, double…

tên_kiểu_mới: Kiểu biến mới do người dùng tự định nghĩa

; kết thúc lệnh

Ví dụ:

```
typedef int nguoi;
nguoi x,y,a[10],b[20][30];
```

7. KHỐI LỆNH

- C thực thi các lệnh theo cấu trúc riêng lẻ, để C thực thi 1 loạt các lệnh thì ta nhóm chúng lại tạo thành 1 khối lệnh.
- **Cấu trúc:**
  - { và }: bắt đầu và kết thúc khối lệnh
  - Lệnh 1, lệnh 2…. các lệnh đơn

```
{
lệnh 1
lệnh 2
--------
lệnh n
}
```

Khối lệnh và lệnh có vai trò tương đương như nhau.

- Có thể khai báo thêm các biến ở đầu mỗi khối lệnh, sau dấu {
- Các khối lệnh có thể lồng nhau, được gọi là các khối lệnh kép.

8. VÀI NÉT VỀ HÀM VÀ CHƯƠNG TRÌNH

- Một chương trình có thể không có hoặc có nhiều hàm.

  Cấu trúc:

```
#include <stdio.h>
------
hàm 1
hàm 2
----
hàm n
```

- Hàm là một đơn vị độc lập của chương trình:
  - Nó được xây dựng riêng, không được phép xây dựng bên trong chương trình khác.
  - Nó có biến, mảng… riêng của mình và chỉ sử dụng bên trong hàm, các biến và mảng có thể khai báo trùng với biến bên ngoài hàm mà ko xảy ra xung đột.
- Việc sử dụng & trao đổi thông tin giữa các hàm thông qua các biến tĩnh bên ngoài hàm hoặc thông qua kết quả trả về của hàm.

9. BIẾN, MẢNG TỰ ĐỘNG

- Các biến, mảng được gọi là tự động khi chúng được khai báo bên trong một hàm.
- Các đối số của hàm hay kết quả trả về của hàm cũng đc gọi là biến tự động.
- Các đặc trưng riêng:
  - Chỉ hoạt động bên trong hàm
  - Thời gian tồn tại khi hàm đc gọi, kết thúc khi thoát hàm
  - Nếu tồn tại trong hàm main, thì sẽ tồn tại trong toàn bộ chương trình.
  - Biến và mảng có thể được khai báo nhưng không sử dụng, khi đó sẽ có cảnh báo của chương trình.

10. BIẾN, MẢNG NGOÀI

- Biến, mảng ngoài là các biến và mảng được khai báo bên ngoài các hàm.
- Các đặc trưng riêng:
  - Tồn tại trong suốt thời gian chương trình chạy.
  - Có hiệu quả bên trong chương trình chính và có thể được đọc bên trong các hàm.
  - Nếu khi khai báo không có giá trị đầu, thì chương trình sẽ gán giá trị 0 cho biến và mảng.
  - Kích thước của mảng có thể không cần khai báo khi khởi tạo, chương trình sẽ tự động nhận kích thước khi lần đầu tiên mảng được gán giá trị.

11. TOÁN TỬ SIZE OF

- Toán tử size of cho phép xác định kích thước của kiểu dữ liệu hoặc đối tượng dữ liệu

  Cấu trúc lệnh:

```
size of (kiểu_dữ_liệu);
size of (đối_tượng_dữ_liệu);
```

  - size of: Từ khóa lệnh
  - kiểu_dữ_liệu, đối_tượng_dữ_liệu: Tham số cần tìm
  - ; kết thúc lệnh
- Kiểu dữ liệu có thể là các kiểu chuẩn như int, float… hoặc các kiểu được người dùng tự khai báo bằng lệnh typedef, enum…
- Đối tượng dữ liệu bao gồm các mảng, cấu trúc.

12. BIẾN TĨNH, MẢNG TĨNH

- Các biến tĩnh, mảng tĩnh được cấp phát bộ nhớ đầy đủ và tồn tại trong suốt thời gian tồn tại của hàm.
- Cú pháp khái báo:

```
static int x,y,z[8];
static float a,b,c;
```

  - static: Từ khóa khai báo biến tĩnh
  - int, float: Kiểu biến
  - x,y,z,a,b,c: Tên biến, mảng
  - ; kết thúc lệnh khai báo.
- Nhận xét: Về cơ bản, biến tĩnh và mảng tĩnh khá giống với biến ngoài và mảng ngoài. Nếu chúng được gọi thông qua 1 hàm khác thì vai trò sẽ như nhau (thông qua lệnh #include….).

13. BIỂU THỨC

- Biểu thức là sự kết hợp của toán hạng và các phép toán để diễn tả một công thức toán học nào đó.
- Biểu thức được dùng trong:
  - Vế phải của lệnh;
  - Tham số thực của hàm, ví dụ hàm in printf;
  - Làm chỉ số, cho phép xác định vị trí biến hoặc mảng;
  - Trong các cấu trúc vòng lặp của chương trình:
    - do – while
    - Switch
    - For
    - If

14. PHÉP TOÁN SỐ HỌC

- Phép toán số học là các phép tính dựa trên thao tác cơ bản về các biến và mảng.
- Phép toán số học bao gồm 5 phép toán cơ bản sau:

Phép toán	Ý nghĩa	Ví dụ
+	Cộng	a + b
-	Trừ	a – b
*	Nhân	a * b
/	Chia	a / b
%	Lấy phần dư	a % b

Ví dụ: 11%3 = 2

- Phép toán *, /, % được ưu tiên cao nhất. Phép toán + và – được ưu tiên thấp hơn.

15. CÁC THAO TÁC BIT

Các thao tác trên bit được sử dụng theo đại số logic, nó có các phép toán đặc trưng đi kèm như sau:

Phép toán	Ý nghĩa	Ví dụ
&	AND	a & b
|	OR	a | b
^	XOR	a ^ b
<<	dịch trái	a << 4
>>	dịch phải	a >> 4
~	lấy phần bù	~ a

Ví dụ: 1|0 = 1;

0x1234 <<8 = 0x3400

0x1234 >>4 = 0x0123

16. PHÉP TOÁN SO SÁNH VÀ LOGIC

- Phép toán so sánh logic dùng để so sánh 2 biểu thức, biến, hàm với nhau.
- Đầu ra của phép so sánh logic chỉ có 1 trong 2 đáp số. Đúng (1) và Sai (0). Dưới đây là các phép toán logic phổ biến:

Phép toán	Ý nghĩa	Ví dụ
>	Lớn hơn	3>7 → 0
>=	Lớn hơn hoặc bằng	3>=2 → 1
<	Nhỏ hơn	3 <7 → 1
<=	Nhỏ hơn hoặc bằng	3 <=5 → 1
==	Bằng nhau	3 == 4 →0
!=	Khác nhau	3 != 4 →1

17. CHUYỂN ĐỔI KIỂU GIÁ TRỊ

- Khi làm việc với các toán hạng và biểu thức, chúng phải cùng kiểu, việc này có thể được tiến hành tự động nhờ chương trình hoặc tiến hành thủ công thông qua việc ép kiểu.
- Cú pháp ép kiểu:

```
(kiểu)(biểu_thức)
```

Ví dụ: int a,b,c;

(float) (a,b);

Sau 2 lệnh trên, c có kiểu là int, a và b có kiểu là float

Quá trình ép kiểu có thể cho ra kết quả khác so với việc chưa ép kiểu. Đặc biệt là với các số lẻ.

18. PHÉP TOÁN TĂNG GIẢM

- Phép toán tăng và giảm sẽ cho phép tác động thẳng vào biến được thao tác, quá trình đó sẽ làm cho giá trị của biến thay đổi 1 đơn vị.
- Phép toán này tương đương với việc + hoặc - 1.
- Cú pháp:

```
x++ /* tăng x lên 1, tăng sau khi dùng*/
x -- /* giảm x xuống 1, trừ sau khi dùng*/
++x /* tăng x lên 1, tăng trước khi dùng*/
--x /* giảm x xuống 1, trừ trước khi dùng*/
```

Ví dụ: x = 7

```
x ++ = 8
x-- = 7
n = ++x → n =8, x = 8
n = x++ → n = 7, x = 8
```

19. CÂU LỆNH GÁN VÀ BIỂU THỨC GÁN

- Câu lệnh gán và biểu thức gán dùng để thay đổi giá trị của biến và biểu thức.
- Cú pháp:

```
tên_biến = tên_biến + x;
biểu_thức_1 = biểu_thức_2
```

- Ngoài phép toán =, còn có thể dùng các phép toán khác. Nhưng nếu sử dụng phép toán logic thì đầu ra chỉ có kết quả là 1 hoặc 0.

Ví dụ:

```
x = x +2
x = y + 5
z = (x+3*y )/ (x % y)
```

20. BIỂU THỨC ĐIỀU KIỆN

- Biểu thức điều kiện dùng để xác định và lựa trọn 1 trong số các biểu thức cú pháp sau:

```
e1?e2:e3
```

- Giá trị của biểu thức này sẽ như sau:
  - e1, e2, e3 là các biểu thức.
  - Giá trị biểu thức = e2 nếu e1 đúng.
  - Giá trị biểu thức = e3 nếu e1 sai.

Ví dụ: s = a>b?a:b

Nếu đúng a>b thì s = a;

Nếu a<b thì s = b.

22. THỨ TỰ ƯU TIÊN CỦA CÁC PHÉP TOÁN

STT	Phép toán	Trình tự kết hợp
1	() [] ->	Trái qua phải
2	! ~ & * - ++ -- type of, size of	Phải qua trái
3	* / %	Trái qua phải
4	+ -	Trái qua phải
5	<< >>	Trái qua phải
6	< <= > >=	Trái qua phải
7	== !=	Trái qua phải
8	&	Trái qua phải
9	^	Trái qua phải
10	|	Trái qua phải
11	&&	Trái qua phải
12	||	Trái qua phải
13	?:	Trái qua phải
14	= += -= *= /= %= <<= >>= &= ^= |=	Phải qua trái
15	,	Trái qua phải

TÓM LƯỢC CUỐI BÀI

- Qua bài 2, người học sẽ có các khái niệm cơ bản về ngôn ngữ lập trình C.
- Tổng quan về các tập ký tự trong C.
- Phân biệt và sử dụng hằng số, biến số, mảng.
- Hiểu được khái niệm hàm trong C.
- Có thể áp dụng giải được các bài toán như giải phương trình bậc 1, phương trình bậc 2.