Chương 03
TỔ CHỨC DỮ LIỆU TRONG CHƯƠNG TRÌNH

Nguyễn Thanh Tùng

Nội dung

- Dữ liệu và Kiểu dữ liệu
- Từ khoá
- Biến và Khai báo biến
- Tầm vực biến
- Phép toán và biểu thức
- Kiểu enum
- Hằng số
- Chuyển đổi kiểu dữ liệu
- Bài tập

Dữ liệu và Kiểu dữ liệu

- Tại sao phải cần đến kiểu dữ liệu?
  - Mọi chương trình đều cần đến dữ liệu
  - Ví dụ:
    - Một chương trình in ra tên đơn giản

```
int main(){
        printf("LAP TRINH C/C++");
        return 0;
}
```

      - => Cần lưu trữ dữ liệu "LAP TRINH C/C++" để xuất ra màn hình

Dữ liệu và Kiểu dữ liệu

- Tại sao phải cần đến kiểu dữ liệu?
  - Mọi chương trình đều cần đến dữ liệu
  - Ví dụ:
    - Một chương trình giải Phương trình bậc 2
      - Dữ liệu:
        - Các hệ số A,B,C của Phương trình bậc 2
        - Delta
        - Các nghiệm của phương trình
    - Một chương trình Quản lý nhân sự
      - Dữ liệu:
        - Mã số nhân sự, họ tên, hệ số lương, v.v.

Dữ liệu và Kiểu dữ liệu

- Tại sao phải cần đến kiểu dữ liệu?
  - Mọi chương trình đều cần đến dữ liệu
  - Người lập trình cần vùng nhớ (thuộc RAM của máy tính) để lưu trữ dữ liệu trong quá trình chương trình thực thi
    - Khi người dùng nhập dữ liệu (thông qua bàn phím, chọn trên màn hình, đọc từ sensor, v.v): dữ liệu sẽ được lưu vào các vùng nhớ của RAM
      - Ví dụ: Đọc các hệ số A,B,và C cho Phương trình bậc 2 từ bàn phím
    - Trong quá trình chương trình thực thi: các vùng nhớ này có thể đọc và xử lý
      - Ví dụ: khi tính DELTA trong giải Phương trình bậc 2, các hệ số sẽ được đọc và các giá trị sẽ được dùng trong biểu thức để tính DELTA (DELTA = B*B – 4*A*C;)

Dữ liệu và Kiểu dữ liệu

- Các kiểu dữ liệu
  - Dữ liệu mà chương trình lưu trữ có thể thuộc nhiều dạng (gọi là **kiểu** hay **kiểu dữ liệu**, **data type**) khác nhau
    - Ký tự (character)
    - Một trong hai trạng thái: có hay không, đúng hay sai
    - Các con số
      - Số nguyên
      - Số thực
    - Một chuỗi: “LAP TRINH C/C++”
    - Một dãy các giá trị
    - Một tổ hợp các giá trị (struct, class)
    - Một trong một số giá trị cho trước (enum)
    - …

Dữ liệu và Kiểu dữ liệu

- Các kiểu dữ liệu
  - Ngoài tính chất lưu trữ khác nhau, chương trình cũng cần thiết phân biệt các kiểu dữ liệu nói trên, vì mỗi kiểu quy định thông tin đi kèm khác nhau
    - Cách tổ chức các bit (lưu trữ)
      - Ví dụ:
        - Với số nguyên: Ý nghĩa bit có trọng số lớn nhất (MSB) phụ thuộc vào nó có kiểu là số có dấu hay không dấu
          - Số không dấu: bit này tham gia vào tính độ lớn giá trị
          - Số có dấu: bit này chỉ ra là số dương hay âm
    - Các phép toán
      - Ví dụ:
        - Với hai con số: có thể thực hiện phép toán: nhân hay chia
        - Không thực hiện nhân hay chia với hai chuỗi ký tự

Dữ liệu và Kiểu dữ liệu

- Ngôn ngữ lập trình phân biệt kiểu dữ liệu như thế nào?
  - Ngôn ngữ C/C++ (các ngôn ngữ khác cũng vậy) gắn ngữ nghĩa (quy ước ngữ nghĩa) với một loạt các tên kiểu mà nó cung cấp sẵn.
  - Các kiểu này được gọi là kiểu cơ bản (fundamental data types)
  - Tên các kiểu có sẵn này đã được gắn sẵn ngữ nghĩa nên nó là **từ khoá.** Người lập trình không được dùng tên này để đặt tên cho các kiểu (hàm, biến, v.v) mà họ tạo ra.

Dữ liệu và Kiểu dữ liệu

- Các loại kiểu
  - Kiểu dữ liệu cơ bản (fundamental data type)
    - Tên kiểu là từ khoá
    - Ngữa nghĩa của tên này được quy định bởi ngôn ngữ lập trình
  - Kiểu dữ liệu do người lập trình định nghĩa (user-defined data type)
    - Tên kiểu do người lập trình đặt ra
    - Ngữ nghĩa do người lập trình quy định thông qua
      - Kiểu người lập trình tạo ra trước đó
      - Và/hoặc, kiểu dữ liệu cơ bản
    - Các kiểu nổi tiếng của C/C++
      - C: struct, enum
      - C++: class

Dữ liệu và Kiểu dữ liệu

- Các loại kiểu
  - Kiểu dữ liệu cơ bản (fundamental data type)
  - Kiểu dữ liệu do người lập trình định nghĩa (user-defined data type)
  - Kiểu dữ liệu dẫn xuất (derived data type)
    - C/C++ cung cấp các ký hiệu để tạo ra kiểu mới từ các kiểu khác (cơ bản hay người lập trình định nghĩa)
    - Ví dụ:
      - Mảng (array)
        - Mảng của các ký tự, của các số nguyên, của các số thực, v.v.
      - Con trỏ (pointer)
        - Con trỏ đến ký tự, đến con số, v.v.

Dữ liệu và Kiểu dữ liệu

- Kiểu dữ liệu cơ bản

Loại	Tên kiểu	#bytes	Giá trị
Không kiểu	void		
Ký tự	char	1	‘a’, ‘\n’
Luận lý	bool	1	True, false
Số thực			
	float	4	1.5f, 100f
	double	8	1.5, 100
	long double	8	1.5l, 100l

Dữ liệu và Kiểu dữ liệu

- Kiểu dữ liệu cơ bản

Loại	Tên kiểu	#bytes	Giá trị
Số nguyên			
(1)	char, signed char	1	-2, 0, 4
(2)	unsigned char	1	0, 1, 255
(3)	short int, signed short int, short, signed short	2	10, -100
(4)	unsigned short int, unsigned short	2	0, 15, 100
(5)	int, signed int, long int, signed long int, long, signed long	4	10, -100
(6)	unsigned int, unigned long int, unsigned long	4	0, 15, 100
(7)	long long int, signed long long int	8	10, -100
(8)	unsigned long long int	8	0, 15, 100

Dữ liệu và Kiểu dữ liệu

- Kiểu dữ liệu cơ bản

(*) Số bytes tuỳ thuộc vào phiên bản, tuy nhiên sẽ thoả mãn:

1 == sizeof(char) <= sizeof(short) <= sizeof(int)
<= sizeof(long) <= sizeof(long long)

Dữ liệu và Kiểu dữ liệu

- Chương trình in kích thước kiểu (số bytes)
  - Hàm: sizeof(.) trả về số byte của kiểu ở thông số

```
#include <stdio.h>
#include <stdlib.h>

int main(){
            //bool
            printf("sizeof(bool) = %3d\n\n", sizeof(bool));

            //char
            printf("char:\n");
            printf("sizeof(char) = %3d\n", sizeof(char));
            printf("sizeof(signed char) = %3d\n", sizeof(signed char));
            printf("sizeof(unsigned char) = %3d\n\n", sizeof(unsigned char));

            //short
            printf("short:\n");
            printf("sizeof(short) = %3d\n", sizeof(short));
            printf("sizeof(signed short) = %3d\n", sizeof(signed short));
            printf("sizeof(unsigned short) = %3d\n\n", sizeof(unsigned short));



            system("pause");
            return 0;
}
```

Dữ liệu và Kiểu dữ liệu

- Đọc thêm
  - Định nghĩa kiểu: typedef

Từ khoá

- Từ khoá là gì
  - Là từ có ý nghĩa đặc biệt đã được quy định trước bởi ngôn ngữ lập trình.
    - Như tên của các kiểu cơ bản nói trên
  - Người lập trình không được dùng từ khoá để đặt tên cho các tên mình tạo ra như tên biến, tên kiểu, tên hàm, tên hằng, v.v.

Từ khoá

- Từ khoá trong C

Keywords in C Language

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

Biến và Khai báo biến

- Biến là gì?
  - Là nơi lưu trữ dữ liệu của chương trình
  - Là tên của vùng nhớ lưu trữ dữ liệu của chương trình
  - Do có tên, nên khi cần đọc/ghi với vùng nhớ này, người lập trình chỉ cần dùng tên thay cho một địa chỉ của nó.
- Sử dụng biến như thế nào?
  - Biến cần được khai báo trước khi dùng (đọc/ghi)
  - Chương trình tự động cấp phát vùng nhờ khi gặp một khai báo biến

Biến và Khai báo biến

- Các vị dụ về khai báo biến:
  - Tạo một biến

    ```
    int a;
    char c;
    ```

  - Tạo nhiều biến cùng kiểu

    ```
    int a, b;
    char c1, c2;
    ```

  - Tạo biến và khởi động giá trị

    ```
    int a=10, b;
    char c1='A', c2='a';
    ```

Biến và Khai báo biến

- Khi khai báo biến cần xác định điều gì?
  - Biến lưu trữ dữ liệu gì? → xác định được kiểu
  - Ý nghĩa của biến là gì? → xác định được tên sẽ được đặt cho nó
  - Ví dụ:
    - Khi giải Phương trình bậc 2
      - Lưu trữ hệ số:
        - Kiểu: float hoặc double. Vì sao?
        - Tên: a, b, c. Vì sao?
      - Lưu trữ delta:
        - Kiểu: float hoặc double. Vì sao?
        - Tên: delta
      - Lưu trữ nghiệm:
        - Kiểu: float hoặc double. Vì sao?
        - Tên: x1, x2, s1, s2, sol1, sol2, v.v. Có luật gì không?

Biến và Khai báo biến

- Quy tắc đặt tên biến
  - Theo quy tắc đặt tên một danh hiệu
- Quy tắc đặt tên danh hiệu
  - Không được trùng với từ khoá
  - Ký tự đầu:
    - Một chữ (a, A, b, B, …) hay gạch dưới (_)
    - Không được là ký hiệu nào khác: !,@,#,$,%,^,&,*,(,), …
  - Các ký tự tiếp theo:
    - chữ, số, gạch dưới

Biến và Khai báo biến

- Minh hoạ

```c
#include <stdio.h>
#include <stdlib.h>
//Chuong trinh giai Phuong trinh bac 2
int main(){
        //Khao bao cac bien
        float a,b,c;
        float delta;
        float x1, x2;
        //Lay a,b,c tu nguoi dung
        //Giai cho truong hop bac 0: a = b = 0
        //Giai cho truong hop bac 1: a = 0
        //Giai cho truong hop 2: a va b <> 0
                //Tinh delta
                //Truong hop: vo nghiem
                //Truong hop: nghiem kep
                //Truong hop: hai nghiem khac nhau
        system("pause");
        return 0;
}
```

Các tầm vực của biến

- Tầm vực là gì?
  - Là vùng chương trình mà một biến tồn tại và sử dụng được
- Các loại tầm vực
  - Toàn cục: bên ngoài tất cả các hàm
  - Cục bộ:
    - Thân hàm: từ dấu { đến dấu } của thân hàm
    - Hoặc các khối con (từ dấu { đến dấu } của khối)
  - Thông số của hàm: từ { đến } của thân hàm

```
#include <stdio.h>
#include <stdlib.h>
#include <typeinfo>

/*Bien cuc bo*/
float g;
double d;
int main(){
        float g;
        double d;

        for(;;){
                float g;
                double d;

        }
        {
                {

                }

        }
        system("pause");
        return 0;
}
```

Hai biến: thuộc tầm vực toàn cục, bên ngoài tất cả các hàm

Hai biến: thuộc tầm vực A

Hai biến: thuộc tầm vực B

TVCB: B

TVCB: C

TVCB: D

Tầm vực cục bộ A

Các tầm vực của biến

- Chương trình có 4 tầm vực cục bộ đặt tên: A, B, C, D
  - A bao hàm của B, C (và D)
  - C bao hàm D
  - Biến g và d thuộc vùng toàn cục có thể dùng được trong toàn bộ chương trình.
    - Bị che khuất tạm thời bởi g và d trong tầm A
    - Do đó, dùng tên đầy đủ là ::g và ::d
  - Biến g và d thuộc A được dùng từ lúc khai báo đến hết A
    - Tuy nhiên, bị che khuất trong B
    - Không dùng được trong B
  - Biến g và d trong B
    - chỉ dùng được cho hết B

Các tầm vực của biến

- Chương trình có 4 tầm vực cục bộ đặt tên: A, B, C, D
  - Khi tham chiếu đến g và d trong A từ sau khi B kết thúc (nghĩa là bao hàm C và D)
    - Thì g và d là g và d của A
  - Nếu muốn g và d của toàn cục thì dùng
    - ::g và ::d

Các tầm vực của biến

- Khởi động cho các biến:
  - Biến cục bộ: không được khởi động hay tuỳ vào trình biên dịch
  - Thông số của hàm: được truyền từ lời gọi hàm
  - Biến toàn cục: khởi động theo bảng sau

Data Type	Initial Default Value
int	0
char	'\0'
float	0
double	0
pointer	NULL

Phép toán và biểu thức

- Phép toán
  - Số học
  - Luận lý và quan hệ
  - Trên các bit
  - Toán tử gán
  - Toán tử khác

Phép toán và biểu thức
Số học

Ký hiệu	Ý nghĩa	Kiểu áp dụng	Ví dụ
+	Cộng hai toán hạng	Các kiểu số	x + y
-	Trừ toán hạng thứ 2 từ toán hạng 1	Các kiểu số	x - y
*	Nhân hai toán hạng	Các kiểu số	x * y
/	Chia tử cho mẫu	Các kiểu số	x / y
%	Lấy phần dư của phép chia nguyên	Kiểu số nguyên hoặc enum	n % 2
++	Tăng một số lên 1	Các kiểu số	++x; y++
--	Giảm một số đi 1	Các kiểu số	--x; y--

Phép toán và biểu thức
Số học

- Toán tử ++
  - Tăng trước:
    - Ví dụ: (++x – y)
      - Tăng giá trị của x lên trước
      - Dùng giá trị của x đã tăng vào biểu thức (cộng với y)
      - Do đó: Nếu x = 4 và y = 5 trước khi đánh giá biểu thức
      - Thì (++x – y) cho giá trị là: 0
  - Tăng sau: x++
    - Ví dụ: (x++ – y)
      - Đánh giá biểu thức dựa trên giá trị của x đang có trước
      - Sau đó, tăng x lên 1 sau
      - Do đó: Nếu x = 4 và y = 5 trước khi đánh giá biểu thức
      - Thì (x++ – y) cho giá trị là: -1
  - Trong cả 2 ví dụ trên: giá trị của x điều là 5 sau khi đánh giá biểu thức

Phép toán và biểu thức
Số học

- Toán tử --
  - Thứ tự đánh giá phép toán tương tự như ++

- Trường hợp thường dùng của ++ và –
  - Để điều chỉnh giá trị biến điều khiển trong các vòng lặp (for, while và do…while)

Phép toán và biểu thức

Toán tử luận lý và quan hệ

Ký hiệu	Ý nghĩa	Kiểu áp dụng	Ví dụ
==	Kiểm tra bằng nhau	Kiểu số	a == b
!=	Kiểm tra khác nhau	Kiểu số	a != b
>	Lớn hơn	Kiểu số	a > b
<	Nhỏ hơn	Kiểu số	a < b
>=	Lớn hơn hay bằng	Kiểu số	a >= b
<=	Nhỏ hơn hay bằng	Kiểu số	a <= b
&&	AND luận lý	Luận lý	b1 && b2
||	OR luận lý	Luận lý	b1 || b2
!	Phủ định luận lý	Luận lý	!flag

Phép toán và biểu thức

Toán tử trên các bit

Ký hiệu	Ý nghĩa	Kiểu áp dụng	Ví dụ
&	AND trên bit	Kiểu số	a & PATTERN
|	OR trên bit	Kiểu số	a | PATTERN
^	XOR trên bit	Kiểu số	a ^ b
~	Phủ định trên bit (Bù 1)	Kiểu số	~a
<<	Dịch trái chuỗi bit	Kiểu số	a << 2
>>	Dịch phải chuỗi bit	Kiểu số	a >> 2

Phép toán và biểu thức

Toán tử gán

Ký hiệu	Ý nghĩa	Kiểu áp dụng	Ví dụ
=	Phép gán A = B	Tất cả	a = (b + c)*f
+=	Tương đương A = A+B	Kiểu số	a += 2
-=	A = A-B	Kiểu số	a -= 2
*=	A = A*B	Kiểu số	a *= 2
/=	A = A/B	Kiểu số	a /= 2
%=	A = A%B	Kiểu số nguyên	a %= 2
<<=	A = A << B	Kiểu số	a <<= 2
>>=	A = A >> B	Kiểu số	a >>= 2
&=	A = A & B	Kiểu số	a &= 2
^=	A = A ^ B	Kiểu số	a ^= 2
|=	A = A | B	Kiểu số	a |= 2

Phép toán và biểu thức
Toán tử khác

Ký hiệu	Ý nghĩa	Kiểu áp dụng
sizeof()	Trả về kích thước của biến hay kiểu	Kiểu cơ bản
&	Trả về địa chỉ của một biến	Tất cả
*	Con trỏ đến một biến	Tất cả
? :	Toán tử điều kiện (ba ngôi)	(C? A: B) C: Biểu thức kiểu luận lý A, B là biểu thức kiểu bất kỳ khác
[]	Truy cập phần tử của mảng	Mảng của kiểu bất kỳ

Phép toán và biểu thức

Độ ưu tiên của các toán tử

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type) * & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Phép toán và biểu thức

- Phép toán
  - Xem thêm: http://www.tutorialspoint.com/ansi_c/c_operator_types.htm

Phép toán và biểu thức

- Biểu thức
  - Được tạo thành từ toán tử và toán hạng
    - Biểu thức: X + Y
      - X, Y: Toán hạng
      - + : Toán tử
      - Đây là toán tử hai ngôi vì có hai toán hạng
    - Biểu thức: ! (A && B)
      - (A && B): Biểu thức con
      - ! : Toán tử một ngôi, vì cần 01 toán hạng
      - &&: Toán tử hai ngôi, hai toán hạng là A và B
  - Toán hạng phải có kiểu tương thích với kiểu mà toán tử có thể thực hiện được

Phép toán và biểu thức

- Biểu thức
  - Các toán hạng có thể là hằng số

Kiểu enum

- Ứng dụng
  - Với các chương trình lớn, một hằng số như 1, 2, v.v không mang đến thông tin về ý nghĩa con số này. Nó có thể là
    - Lựa chọn mà người dùng nhập vào
    - Một tháng trong năm hay một ngày trong tháng
    - Một ký hiệu về màu sắc
    - V.v
  - Để tăng tính dễ đọc, dễ hiểu, dễ bảo trì, các hằng số nên được ký hiệu hoá bởi một tên, tên này được người lập trình chọn để lồng ghép ý nghĩa của tên này. Ở trường hợp này nên dùng enum

Kiểu enum

- Ví dụ về khai báo một enum

  (1) Tập màu sắc:

  enum colors {RED, GREEN, BLUE};

  (2) Tập các tháng:

  enum months {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC};

  (3) Tập các lựa chọn cho người một chương trình:

  enum user_choices {LOAD_DATA, INPUT_DATA, PRINT_DATA};

Kiểu enum

- enum là gì?
  - Có thể được xem như một kiểu dữ liệu.
    - Ở các ví dụ trên ta có các kiểu là: colors, months, user_choices.
      - Một biến kiểu colors chỉ có thể RED, GREEN, BLUE như đã khai báo
      - Một biến kiểu months ở trên chỉ có thể có các giá trị JAN, FEB, MAR, v.v.
      - Một biến kiểu user_choices ở trên chỉ có thể có các giá trị LOAD_DATA, INPUT_DATA, v.v.
    - Nghĩa là người dùng có thể tạo ra kiểu mới

Kiểu enum

- enum là gì?
  - Có thể được xem như một tập hợp các hằng số.
    - Ở các ví dụ trên ta có các tập hợp là: colors, months, user_choices.
      - Với tập colors có các hằng: RED, GREEN, BLUE như đã khai báo
      - Với tập months có các hằng: JAN, FEB, MAR, v.v.
      - Với tập user_choices có các hằng: LOAD_DATA, INPUT_DATA, v.v.
    - Về bản chất, các giá trị trong tập hợp (enum) được TỰ ĐỘNG gán một con số nguyên. Hằng đầu tiên là 0, kế tiếp là 1, v.v. Tuy nhiên lập trình viên không nên giả thiết các hằng đó là 0, 1, …

Kiểu enum

- Khai báo và dùng biến kiểu enum

```
#include <stdio.h>
#include <stdlib.h>

int main(){
        enum colors {RED, GREEN, BLUE};
        colors b = BLUE, c = GREEN;
        printf("b= %d\n", b);
        printf("c= %d\n", c);

        system("pause");
        return 0;
}
```

Chương trình in ra:
b = 2
c = 1
Vì sao?

Kiểu enum

- In ra tên của các hằng trong enum như thế nào?

```
#include <stdio.h>
#include <stdlib.h>

int main(){
        enum colors {RED, GREEN, BLUE};
        char* colors_names[] = {"RED", "GREEN", "BLUE"};
        colors b = BLUE, c = GREEN;
        printf("b= %s\n", colors_names[b - RED]);
        printf("b= %s\n", colors_names[c - RED]);

        system("pause");
        return 0;
}
```

Chương trình minh hoạ cần đến kiểu mảng (array), sẽ học sau.

Các giá trị bất biến

- Giá trị bất biến là gì?
  - Là giá trị không thay đổi trong quá trình chương trình thực thi.
  - Do đó,
    - giá trị bất biến phải được xác định tại thời điểm biên dịch (lập chương trình)
- Ví dụ
  - (1) Sử dụng từ khoá const

    ```
    const int MAX = 50;
    ```
  - (2) Sử dụng macro

    ```
    #define MAX 50
    ```
  - (3) Sử dụng enum

    ```
    enum {MAX};
    enum {MAX = 50};
    ```

Các giá trị bất biến

- Giá trị bất biến và cách thể hiện trong mã chương trình
  - Kiểu ký tự

    const char c = ‘a’;

    const char c = ‘A’;

  - Kiểu chuỗi

    const char c[] = “LAP TRINH C/C++”;

    const char c[] = “SAI GON”;

  - Kiểu số

    const int a = 100;

    const float f = 10.5f;

    const double d = 10.5;

Chuyển đổi kiểu

- Chuyển đổi kiểu là gì?
  - Khi ta gán một giá trị vào một biến cùng một kiểu dữ liệu, hay khi thực hiện phép toán nhị phân giữa hai giá trị cùng một kiểu dữ liệu thì không cần sự chuyển đổi kiểu nào xảy ra
    - Ví dụ

```
int a = 100;
int b;
b = a;

float f = 10.5f;
float fa;
fa = f;
```

Chuyển đổi kiểu

- Chuyển đổi kiểu là gì?
  - Khi người lập trình thực hiện phép gán mà kiểu của bên phải phép gán (kiểu nguồn) và kiểu bên trái phép gán (kiểu đích) khá nhau, một sự chuyển đổi giá trị từ kiểu nguồn sang kiểu đích là cần thiết
  - Khi hai toán hạng trong cùng một phép toán nhị phân khác kiểu với nhau thì cũng cần sự chuyển đổi dữ liệu
    - Ví dụ

```
short a = 100;
Int b;
b = a;

double d = 10.5;
float fa;
fa = d;
```

Chuyển đổi kiểu

- Các dạng chuyển đổi
  - Chuyển đổi ngầm (mặc nhiên)
    - Có thể giữa nguyên giá trị nguồn
    - Có thể biến đổi giá trị nguồn
  - Ép kiểu
    - Người lập trình can thiệp bằng toán tử đặc biệt: cặp dấu ”(” và “)”.

Chuyển đổi kiểu

- Chuyển đổi ngầm (mặc nhiên)
  - Áp dụng khi giá trị kiểu nguồn được cópy vào kiểu đích **tương thích** với kiểu nguồn.
  - Ví dụ:

```
short s = 100;
int a = s;  // có sự chuyển đổi giá trị
```

    - Biến a có kiểu "int" là kiểu đích của phép chuyển đổi
    - Biến s có kiểu "short" là kiểu nguồn
    - "short" tương thích với "int" nên trình biên dịch không báo lỗi và chương trình chuyển tự động mà không cần người lập trình đặt tả gì thêm

Chuyển đổi kiểu

- Chuyển đổi ngầm
  - Trường hợp giữ nguyên giá trị nguồn
    - Khi kiểu nguồn có số bit thấp hơn kiểu đích
    - Ví dụ
      - char vào short
      - short vào int
      - Int vào long long
      - Int vào double

Chuyển đổi kiểu

- Chuyển đổi ngầm
  - Trường hợp giữ nguyên giá trị nguồn
  - Trường hợp cần sự thay đổi
    - unsigned vào signed và ngược lại
      - Áp dụng nguyên tắc “bù 2” (two’s complement)
        - Bù 2 = Bù 1 + 1
        - Bù 1 = Bù trên bit (0 → 1, 1 → 0)
    - Luận lý vào số:
      - true → 1
      - false → 0
    - số vào Luận lý
      - Khác 0 → true
      - Bằng 0 → false

Chuyển đổi kiểu

- Chuyển đổi ngầm
  - Trường hợp giữ nguyên giá trị nguồn
  - Trường hợp cần sự thay đổi
    - unsigned vào signed và ngược lại
    - Luận lý vào số:
    - số vào Luận lý
    - float hay double vào số nguyên (char, short, int, v.v)
      - Phần thập phân bị cắt bỏ (2.456 → 2)
      - Nếu sau khi cắt mà phần nguyên vẫn nằm ngoài phạm vi của kiểu đích thì …. tuỳ vào hiện thực (không xác định)

Chuyển đổi kiểu

- Ép kiểu (type casting)
  - Là chuyển đổi kiểu tường minh, người lập trình đặt tả kiểu đích
  - Cú pháp
    - Dạng hàm

```
double x = 10.5;
int a = int(x);
```

    - Dạng c

```
double x = 10.5;
int a = (int) x;
```

  - Ví dụ
    - Xác định giá trị của x và y trong ví dụ cho sau đây

Chuyển đổi kiểu

- Ép kiểu (type casting)

```
#include <stdio.h>
#include <stdlib.h>

int main(){
        double x = 3/2;
        double y = (double)3/2;

        printf("x = %4.2g\n", x);
        printf("y = %4.2g\n", y);

        system("pause");
        return 0;
}
```

Chuyển đổi kiểu

- Ép kiểu (type casting)
  - Ví dụ
    - Xác định giá trị của a và b và kết quả xuất ra màn hình trong ví dụ cho sau đây

Chuyển đổi kiểu

- Ép kiểu (type casting)

```
#include <stdio.h>
#include <stdlib.h>

int main(){
        unsigned char a;
        signed char b;
        a = -1;
        b = 255;

        printf("a= %3d\n", a);
        printf("b= %3d\n", b);
        printf("a < 5 is %s\n", a < 5 ? "true": "false");
        printf("b > 5 is %s\n", b > 5 ? "true": "false");

        system("pause");
        return 0;
}
```

Bài tập

Tổng kết

Chủ đề	Yêu cầu (Kiến thức & Kỹ năng)
Dữ liệu và Kiểu dữ liệu	- Biết được danh mục kiểu dữ liệu cơ bản - Biết được kích thước, phép toán áp dụng được với các kiểu cơ bản - Quy tắc tên danh hiệu - Khai báo được biến & khở động chúng
Từ khoá	
Biến và Khai báo biến	
Tầm vực biến	Các loại tầm vực và ý nghĩa
Phép toán và biểu thức	Sử dụng phép toán trong biểu thức
Kiểu enum	Định nghĩa và sử dụng enum
Kiểu cấu trúc	
Hằng số	Sử dụng được hằng ở nhiều cách
Chuyển đổi kiểu dữ liệu	Hiểu được quy tắc chuyển đổi và sử dụng ép kiểu được