KỸ THUẬT LẬP TRÌNH C/C++

Thi-Lan Le
Thi-Lan.Le@mica.edu.vn; lan.lethi1@hust.edu.vn
Webpage: http://www.mica.edu.vn/perso/Le-Thi-Lan

Dịch c và c++ sử dụng Visual Studio

- Project-> Properties->C/C++->Advanced->Compile As:
  - Compile as C Code (/TC): cho C code
  - Compile as C++ Code (/TC): cho C++ code

Chương trình C đơn giản nhất

```
/* hello.c */
#include <stdio.h>
int main() {
   printf("Xin chao!\n");
   return 0;
}
```

- Chương trình in ra màn hình:

```
Xin chao!
```

Chương trình C đơn giản nhất

- Dừng màn hình:
  - Cách 1:
    - system(“PAUSE”);
    - Cần include <iostream>
  - Cách 2:
    - getch();

Phân tích chương trình ví dụ

- Chương trình trên có:
  - Định nghĩa hàm main()
  - Một dòng chú thích
  - Một dẫn hướng biên dịch (dùng thư viện)
  - Một câu lệnh xuất ra màn hình (đầu ra chuẩn)
  - Một câu lệnh trả kết quả
- Chương trình thực hiện:
  - Yêu cầu máy tính in ra một dòng chữ ra màn hình
  - Trả kết quả về là 0 cho chương trình gọi nó

Hàm main()

- Là hàm dùng để bắt đầu chạy một chương trình C, và bắt buộc phải có
- Khai báo bằng một trong hai cú pháp:
  - `int main() { … }`
  - `int main(int argc, char* argv[]) { … }`
  - Trong C++ có thể khai báo hàm main() với kiểu trả về là void
- Khi bắt đầu chạy, một số tham số sẽ được truyền cho chương trình; và khi kết thúc, chương trình sẽ trả về một giá trị. VD:
  - `C:\>copy /B a.dat b.dat`

Ví dụ 2: Tính diện tích hình tròn

```
#include <stdio.h>
int main()
{
    float R;
    printf("Ban kinh = ");
    scanf("%f", &R);
    printf("Dien tich hinh tron: %.3f\n", 3.14 * R*R);
    return 0;
}
```

- Kết quả chạy:

```
Ban kinh = 1
Dien tich hinh tron: 3.140
```

Biến (variable) và kiểu (type)

- Biến chứa giá trị, có thể thay đổi trong khi chạy
- Biến cần được khai báo trước khi dùng và có kiểu
- Phạm vi toàn cục hoặc chỉ trong nội bộ một hàm
- Trong C chuẩn, biến nội bộ cần được khai báo ở đầu hàm, trước các câu lệnh
- Khai báo biến: <kiểu> <danh sách các biến>;
  - `int a, b, c;`
  - `unsigned char u;`
- Các kiểu cơ bản:
  - `char, int, short, long`
  - `float, double`

Câu lệnh gán (assignment)

- Thay đổi giá trị của biến bằng giá trị mới
- Cú pháp:
  - <biến> = <hằng, biến> hoặc <biểu thức>
- Ví dụ:
  - `count = 100;`
  - `value = cos(x);`
  - `i = i + 2;`
- Biến có thể được khởi tạo giá trị khi khai báo (nếu không sẽ có giá trị không xác định):
  - `int count = 100;`
  - `char key = 'K';`

Hằng (constant)

- Tương tự như biến nhưng giá trị của nó không thể bị thay đổi trong quá trình chạy
- Khai báo bằng cách thêm từ khoá `const` ở trước
- Hằng trong C có chiếm bộ nhớ giống như biến
- Ví dụ:
  - `const double PI = 3.14159;`
  - `const char* name = "Nguyen Viet Tung";`
  - `PI = 3.14; /* sẽ báo lỗi */`
- Cách khác để khai báo hằng: tạo macro → không chiếm bộ nhớ (nhưng không có kiểu)
  - `#define PI 3.14159`

Các kiểu dữ liệu cơ bản

Kiểu	Độ dài (Kích thước)	Loại
char	1	Số nguyên, ký tự
int	(tuỳ thuộc: 2, 4, 8)	Số nguyên
short	2	Số nguyên
long	4	Số nguyên
long long	8	Số nguyên
float	4	Số thực (dấu chấm động)
double	8	Số thực (dấu chấm động)
void	0	Không có ý nghĩa xác định

- Ký tự trong C được hiểu là số nguyên 8 bit
- Toán tử sizeof() tính độ dài của biến hoặc kiểu dữ liệu theo số byte:
  - `sizeof(x)`
  - `sizeof(int)`

Ép kiểu (type casting)

- Là việc chuyển từ một biểu thức có kiểu nào đó sang một kiểu khác
- Chuyển kiểu ngầm định:
- Chuyển kiểu tường minh:
  - `int a = (int)5.6; /* lấy phần nguyên */`
  - `float f = (float)1/3;`
- Không phải kiểu nào cũng chuyển được cho nhau
  - `char* s = 2.3; /* không dịch được */`
  - `int x = "7"; /* dịch được nhưng sai */`

Kích thước biến, giới hạn giá trị

- Số có dấu và không dấu:

```
signed char  (8 bits)  –128 ~ +127
signed short (16 bits) –32768 ~ +32767
signed int   (32 bits) –2147483648 ~ +2147483648
signed long  (32 bits) –2147483648 ~ +2147483648
unsigned char  (8 bits) 0 ~ +255
unsigned short (16 bits) 0 ~ +65535
unsigned int   (32 bits) 0 ~ +4294967295
unsigned long  (32 bits) 0 ~ +4294967295
```

- Chú ý:
  - Ngầm định là có dấu
  - Kiểu int có kích thước tuỳ thuộc vào cấu hình

Kiểu liệt kê (enum)

- Dùng để liệt kê các giá trị có thể có của một kiểu
- Cú pháp: enum <tên kiểu> { <các giá trị> };
- Ví dụ:
  - `enum DongVat { Meo, Cho, Ho, Bao };`
  - `enum Ngay { Thu2 = 2, Thu3, Thu4, Thu5, Thu6, Thu7, CN = 1 };`
- Sử dụng:
  - `enum DongVat dv = Meo;`
  - `dv = Bao;`
  - `enum Ngay n = Thu5;`

Kiểu cấu trúc (struct)

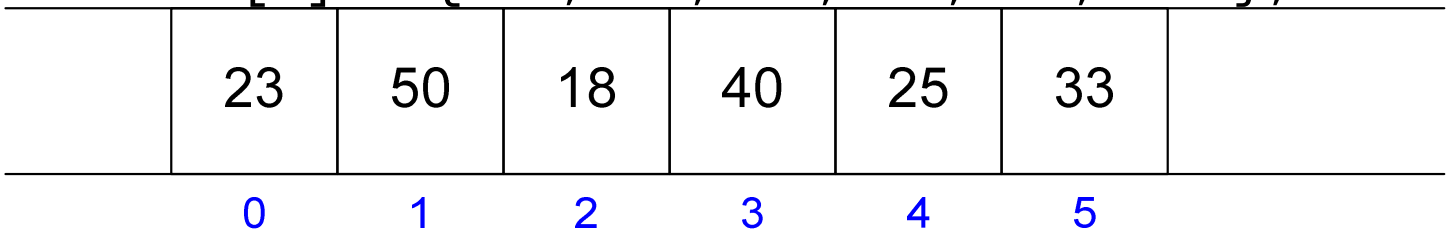
- Khai báo các kiểu phức tạp, chứa các biến con
- Cú pháp: struct <tên kiểu> { <các thuộc tính> };
- Ví dụ:
  - ```
    struct SinhVien {
        char ten[20];
        int nam_sinh;
        int khoa;
    };
    ```
- Sử dụng:
  - `struct SinhVien sv = {"Le Duc Tho", 1984, 56};`
  - `sv.nam_sinh = 1985;`
  - `sv.khoa = 54;`
    ```

Định nghĩa tên mới cho kiểu (typedef)

- Để dùng với tên mới ngắn gọn hơn, hoặc mang ý nghĩa khác
- Cú pháp: typedef <kiểu gốc> <tên kiểu định nghĩa>;
- Ví dụ:
  - `typedef double ChieuCao;`
  - `typedef unsigned char byte;`
  - `typedef enum DongVat DV;`
  - `typedef struct { … } SinhVien;`
- Khai báo biến
  - `ChieuCao d = 165.5;`
  - `byte b = 30;`
  - `DV dv = Cho;`

Kiểu mảng (array)

- Chứa các phần tử cùng kiểu trên một vùng nhớ liên tục. Bản chất của mảng là con trỏ tĩnh.
- Cú pháp: <kiểu> <tên> [<số phần tử>];
- Ví dụ:
  - `int tuoi[6] = { 23, 50, 18, 40, 25, 33 };`



- Truy xuất phần tử: số thự tự tính từ 0
  - `tuoi[3] = 20;`
- Mảng hai chiều (và nhiều chiều):
  - `float ma_tran[10][20];`
    `ma_tran[5][15] = 1.23;`

Một số kiểu khác

- Kiểu boolean:
  - Không có trong C
  - Dùng int/char hoặc enum để thay thế:
    - `typedef int bool;`
    - `typedef enum {false, true} bool;`
- Kiểu chuỗi ký tự:
    - `char* ho_ten = "Nguyen Viet Tung";`
- Kiểu union
  - Chứa các biến thành phần ở cùng một địa chỉ bộ nhớ
    -
```
union color {
        struct {unsigned char R,G,B,A;} s_color;
        unsigned int i_color;
 };
```

Kiểu hợp

- Có thể định nghĩa kết hợp giữa các kiểu cùng loại hoặc khác loại
  - ```
    typedef struct {
        char ho_ten[20];
        unsigned int tuoi;
        enum {Nam, Nu} gioi_tinh;
        struct {
            char thanh_pho[20];
            char duong[20];
            int so_nha;
        } dia_chi;
    } SinhVien;
    ```
    ```

Hiển thị ra màn hình

- Cú pháp:
  - `printf("Chuỗi định dạng", <các giá trị>);`
- Các ký hiệu định dạng thường dùng:

Ký hiệu	Kiểu	Ký hiệu	Kiểu
%f, %e, %g, %lf	double, float	%x	int (hex)
%d	int	%o	int (oct)
%c	char	%u	unsigned int
%s	chuỗi ký tự	%p	con trỏ

- Định dạng:
  - `%[flags] [width] [.precision]type`
  - Ví dụ: `%+15.5f`

Nhập dữ liệu từ bàn phím

- Cú pháp:
  - `scanf("Chuỗi định dạng", <địa chỉ biến>);`
- Ví dụ:
  - ```
    int tuoi;
    scanf("%d", &tuoi);
    ```
  - ```
    float can_nang;
    scanf("%f", &can_nang);
    ```
  - ```
    char ten[20];
    scanf("%s", ten);
    Hoặc
    fflush (stdin);
    gets(ten);
    ```
    ```

Nhập/xuất dữ liệu

Tầm quan trọng của các thao tác nhập/xuất dữ liệu.

- Cho phép chương trình nhập dữ liệu / xuất dữ liệu từ/đến các thiết bị vào/ra chuẩn
- **Các thao tác nhập dữ liệu** phải đáp ứng một số yêu cầu sau:

- Cho phép người sử dụng (NSD) có thể nhập đầy đủ các loại dữ liệu. Yêu cầu tối thiểu của NSD là các kiểu dữ liệu cơ bản như kiểu số, kiểu kí tự, kiểu chuỗi, kiểu logic.
- Cho phép NSD nhập dữ liệu từ nhiều nguồn khác nhau như từ bàn phím, từ chuột, từ các cổng vào,…
- Hỗ trợ NSD nhập dữ liệu một cách chính xác.
- Cho phép NSD nhập dữ liệu một cách nhanh chóng và dễ dàng.

Nhập/xuất dữ liệu

- **Giới thiệu các thao tác vào/ra trong C**
- Các thao tác vào/ra có thể chia làm hai nhóm:
  - Các thao tác vào/ra từng kí tự
  - Các thao tác vào/ra nhiều kí tự (từng chuỗi kí tự).
- Ngoài ra, các thao tác vào/ra còn có thể được chia làm hai loại:
  - + Các thao tác có định dạng :
  - + Các thao tác không có định dạng.
- Trong C, mỗi thao tác vào/ra được thực hiện bởi một hàm. Các hàm vào ra cơ bản đều được khai báo và định nghĩa trong hai tệp thư viện <stdio.h> và <conio.h>.

Các thao tác nhập dữ liệu

- Các thao tác nhập dữ liệu được chia làm hai nhóm:
- Các thao tác nhập từng kí tự (từng byte)
- Các thao tác nhập từng chuỗi kí tự (nhiều byte)
- Các thao tác nhập kí tự
  - *getchar()*
  - *getch()*
  - *getche()*

Các thao tác nhập dữ liệu

- Các thao tác nhập kí tự
- Một số điểm chung là:
  - Chúng đều không có tham số,
  - Các hàm này đều trả về một số nguyên (kiểu int) là mã ASCII của kí tự nhập vào.
  - Các hàm này đều nằm trong thư viện vào/ra chuẩn <stdio.h>
- Tuy nhiên, các hàm này cũng có một số điểm chi tiết khác nhau.

Các thao tác nhập dữ liệu

- *Bộ đệm (buffer) bàn phím*
- Là một vùng bộ nhớ dành riêng để lưu giữ các kí tự nhập vào từ bàn phím trước khi chúng được các chương trình lấy ra để xử lý.
- Được sinh ra để tăng tính độc lập và đồng bộ giữa các thao tác xử lý và các thao tác vào/ra dữ liệu, vì tốc độ giữa các thao tác này khác nhau rất nhiều.
- Làm việc theo nguyên tắc hàng đợi.

Các thao tác nhập dữ liệu

Các thao tác nhập chuỗi kí tự

Hàm *gets()*

- Cú pháp : *gets(char s[])* ;
- Ý nghĩa : hàm này đọc một chuỗi kí tự từ bàn phím cho đến khi bạn gõ ↵. Chuỗi kí tự đọc vào sẽ được gán cho biến *s*. Kí tự kết thúc chuỗi (‘\0’) cũng được tự động bổ sung vào cuối chuỗi s.

Các thao tác nhập dữ liệu

Hàm gets() – ví dụ

```
#include <stdio.h>
void main() {
    char  name[80] ;
    printf("Nhap ten :");
    gets(name) ;
    printf("Ten nhap vao
    la : %s",name);
}
```

Kết quả thực hiện chương trình:

Nhap ten : binh ↵

Ten nhap vao la : binh

Các thao tác nhập dữ liệu

Các thao tác nhập chuỗi kí tự

Hàm *scanf()*

- **Chức năng**: Là hàm nhập dữ liệu đa năng, có định dạng.
- **Cú pháp**:

  scanf(Hằng chuỗi định dạng, danh sách các biến);

- Trong đó :
  - **Hằng chuỗi định dạng** bao gồm 1 hoặc nhiều mã định dạng.
  - **Danh sách các biến** bao gồm một hoặc nhiều biến mà ta muốn nhập giá trị. Số lượng mã định dạng đúng bằng số lượng các biến. Mỗi mã định dạng tương ứng cho một biến vào để xác định kiểu dữ liệu và khuôn dạng dữ liệu nhập vào cho biến đó. Mỗi mã định dạng là một hằng chuỗi kí tự đặc biệt với kí tự đầu tiên luôn là %

2. Các thao tác nhập dữ liệu

Các thao tác nhập chuỗi kí tự

Hàm *scanf()*

- Nguyên tắc hoạt động
  - Khi ấn Enter, hàm sẽ kiểm tra số lượng DL đã nhập và số tham số cần nhập DL.
  - Nếu khớp nhau thì thực hiện chuyển đổi kiểu tương ứng cho các DL đã nhập rồi gán cho các tham số cần nhập.
  - Nếu không khớp thì sẽ tiếp tục đợi nhập tiếp.

Các thao tác xuất dữ liệu

- Trong C có các hàm xuất dữ liệu sau:
  - ***putchar()*** *:* hàm này để in một kí tự ra màn hình.
  - ***puts()*** *:* hàm này để in một chuỗi kí tự ra màn hình
  - ***printf()*** *:* in các loại dữ liệu ra màn hình và có định dạng.

Các thao tác xuất dữ liệu

Hàm **putchar()**

- Cú pháp : ***putchar(char c)*** ;
- Ý nghĩa : Hàm này in kí tự c ra màn hình ở vị trí hiện thời của con trỏ. Đây là hàm xuất dữ liệu không định dạng.

Hàm **puts()**

- Cú pháp : ***puts(const char *s)*** ;
- Ý nghĩa : Hàm này in chuỗi kí tự trỏ bởi con trỏ s ra màn hình, sau đó tự động đưa con trỏ xuống đầu dòng tiếp theo.

Các thao tác xuất dữ liệu

VD: sử dụng hàm putchar() để in các kí tự từ a-z

```
#include <stdio.h>
#include <conio.h>
void main(){
   int i='a';
   while (i<='z')
            { putchar(i++);putchar(' ');}
   getch();
}
```

3. Các thao tác xuất dữ liệu

Hàm ***printf()***

- **Giới thiệu:** Đây là hàm xuất dữ liệu ra màn hình phổ biến nhất. Nó cho phép in ra một hay nhiều biểu thức ở tất cả các kiểu dữ liệu cơ bản như kí tự, chuỗi, số và cho phép định dạng lại dữ liệu khi in ra.
- Việc định dạng dữ liệu như căn lề (trái hoặc phải), định kích thước vùng dữ liệu sẽ in ra (tính bằng số kí tự sẽ in ra, kích thước này có thể khác với kích thước thật của dữ liệu),…

Các thao tác xuất dữ liệu

Hàm ***printf()***

- Cú pháp :

  printf(*Hằng chuỗi định dạng[, danh sách các biểu thức]);*

- Ý nghĩa: hàm này sẽ in hằng chuỗi định dạng ra màn hình.

Hằng chuỗi này có 2 phần nội dung :

- Phần nội dung tĩnh :
- Phần nội dung động :

Hàm printf()

- **Phần nội dung tĩnh:**
  - Là phần nội dung cố định, được xác định bởi một hay một số hằng chuỗi cho trước.
  - Là phần bắt buộc phải có.

Hàm printf()

- **Phần nội dung động:**
  - Là phần nội dung không cố định, mà giá trị của nó được lấy từ giá trị các biểu thức trong *danh sách các biểu thức,* nên các biểu thức trong danh sách này được gọi là các *biểu thức thay thế.*
  - Phần nội dung này được quy định bởi các chuỗi kí tự đặc biệt được gọi là các *mã định dạng* (format code), với kí tự bắt đầu luôn là %.
  - Với mỗi nội dung động sẽ có một mã định dạng quy định khuôn dạng của nội dung và một biểu thức tương ứng trong danh sách các biểu thức để xác định giá trị cụ thể của nội dung.
  - Khuôn dạng của mỗi nội dung động được xác định lúc viết hàm printf. Còn giá trị của nội dung đó là giá trị của biểu thức thay thế tương ứng mà chỉ có thể xác định vào lúc chạy chương trình.
  - Phần nội dung động không bắt buộc phải có.

Hàm printf()

- VD : sử dụng hàm printf();

```
char  c = 'X' ;
int  i =10 ;
const PI=3.1415 ;
float  r=11.2345678 ;
printf("Kí tự %c có mã ASCII là
    %d \n",c,c) ;
printf("%d bình phương", i) ;
printf(" bằng %d ", i*i) ;
printf("Bán kính hình tròn là %f ",r)
    ;
printf(" có diện tích là %.3f
    \n",PI*r*r) ;
```

- Kết quả chạy chương trình :

Kí tự X có mã ASCII là 88

10 bình phương bằng 100

Bán kính hình tròn là 11.234568 có diện tích là 378.647

Hàm printf()

3.3.2. Mã định dạng

a) Cấu tạo: Phần mã định dạng có nhiệm vụ xác định kiểu dữ liệu và khuôn dạng dữ liệu sẽ được in ra. Mã định dạng là một dãy kí tự đặc biệt luôn bắt đầu bởi kí tự %. Mã định dạng có cấu tạo gồm hai phần như sau:

% [mã khuôn in] mã chuyển đổi

Trong đó:

- *Mã khuôn in* : phần này quy định khuôn dạng của dữ liệu sẽ được in ra như kích thước dữ liệu (tính theo số kí tự in ra), căn lề (trái hay phải),… Phần này không bắt buộc phải có. Khi không có phần này, hệ thống sẽ sử dụng khuôn in mặc định cho từng kiểu dữ liệu. Chi tiết về mã khuôn in sẽ được trình bầy trong phần sau.
- *Mã chuyển đổi* : Đây là phần bắt buộc phải có. Phần này quy định kiểu dữ liệu sẽ được in ra.

Hàm printf()

b) Mã khuôn in

- - Cấu tạo mã khuôn in: **[-][+][0][m | *][.n | .*]**

  Với m, n là các số nguyên không âm. Trong đó:
- Dấu '-' : xác định căn lề bên trái. Không có dấu này là căn lề bên phải
- Dấu '+' : áp dụng cho kiểu dữ liệu số (số nguyên hoặc thực), quy định số dương in ra sẽ có dấu +, số âm sẽ có dấu - .

Hàm printf()

- Giá trị *m :* là một hằng số nguyên xác định kích thước tối thiểu của dữ liệu sẽ in ra (tính theo số kí tự).
  - Nếu *m* lớn hơn kích thước thật sự của dữ liệu thì các kí tự trắng (hoặc số 0) sẽ được tự động thêm vào bên trái hoặc bên phải.
  - Trái lại, nếu *m* nhỏ hơn hoặc bằng kích thước dữ liệu thật thì kích thước thật của dữ liệu sẽ được in ra.
  - Ta có thể thay hằng số *m* bởi dấu *. Khi đó, giá trị kích thước tối thiểu sẽ được tính bởi giá trị của một biểu thức thay thế tương ứng nằm trong danh sách các biểu thức.

Hàm printf()

b) Mã khuôn in (tiếp)

- Số 0 : lựa chọn này áp dụng cho kiểu dữ liệu số (số nguyên, số thực), khi căn lề bên phải và kích thước dữ liệu in ra lớn hơn kích thước dữ liệu thật. Khi đó, các số 0 sẽ được bổ sung vào phía trước số sẽ được in ra cho đủ kích thước tối thiểu của dữ liệu in ra. Chế độ mặc định là chèn các kí tự trắng. Việc chèn số 0 chỉ chèn vào phía trước của các kiểu dữ liệu số.

Hàm printf()

- Giá trị n : luôn đi sau một dấu ‘.’. Lựa chọn này áp dụng cho các kiểu dữ liệu khác nhau với các vai trò khác nhau. Nếu áp dụng cho số thực, giá trị này xác định độ chính xác (số chữ số sau dấu phẩy) của số thực đưa ra. Độ chính xác mặc định cho kiểu số thực là 6 chữ số thập phân. Nếu áp dụng cho số nguyên, nó xác định số chữ số tối thiểu của số nguyên sẽ được in ra. Khi n lớn hơn số chữ số thực của số nguyên thì các số 0 sẽ được tự động chèn vào đầu cho đủ n chữ số. Nếu áp dụng cho kiểu chuỗi, giá trị này xác định số kí tự tối đa của chuỗi sẽ được in ra. Khi n nhỏ hơn hoặc bằng độ dài thực của chuỗi kí tự thì chỉ n kí tự đầu tiên của chuỗi được in ra. Giá trị hằng số n cũng có thể được thay thế bởi kí hiệu ‘*’.

Hàm printf() – Ví dụ áp dụng

```
main(){
  char ch='A';
  int i=1234,j=-1234;
  float x=11.23456789;
  printf("%-5c,%5c\n",ch,ch);
  printf("%d,%-10d,%+10d,
    %10.6d,%010d\n",i,i,i,i,j);
  printf("%f,%12f,%-
    12.4f,%+012.4f\n",x,x,x,x);
  printf("%*f\n",12,x);
  printf("%*.*f",12,3,x);
  getch();
  return;
}
```

Kết quả chạy chương trình:

A^^^^,^^^^A
1234,1234^^^^^^,^^^^^+1234,
^^^^001234,-0000001234
11.234568,
^^^11.234568,11.2346^^^^^,
+000011.2346
^^^11.234568
^^^^^11.235

Quy ước các dấu '^' biểu diễn các kí tự trắng để dễ theo dõi.

Bảng mã định dạng

Mã chuyển đổi	Ý nghĩa	Mã chuyển đổi	Ý nghĩa
%c	in ra một kí tự	%ld	in ra một số nguyên dài (long) ở hệ thập phân
%s	in ra một chuỗi kí tự	%lu	in ra một số nguyên dài không dấu ở hệ thập phân
%d,%i,%hd	in ra một số nguyên ở hệ thập phân	%f,%lf	in ra một số thực ở dạng dấu phẩy động
%o	in ra một số nguyên ở hệ cơ số 8	%e,%E	in ra một số thực ở dạng khoa học
%x,%X	in ra một số nguyên ở hệ cơ số 16	%g,%G	in ra một số thực ở dạng dấu phẩy động hoặc dạng khoa học
%u	in ra một số nguyên không dấu ở hệ thập phân		

Bài tập

1. Dùng toán tử sizeof() in ra màn hình độ dài các kiểu dữ liệu cơ bản và phức hợp
2. Nhập góc α và tính giá trị các hàm lượng giác
3. Nhập dữ liệu cho cấu trúc SinhVien (tên, năm sinh, khoá) và in lại giá trị ra màn hình
4. Viết một chương trình, khai báo hai biến x (char) và y (unsigned char). Gán -1 vào x, sau đó thực hiện chuyển kiểu và gán giá trị của x vào y. In kết quả của y ra màn hình. Giải thích kết quả.
5. Khai báo một kiểu dữ liệu miêu tả các thông tin của một chiếc ôtô có các thuộc tính: model, khối lượng, màu sơn, 4 bánh trong đó mỗi bánh có thuộc tính: chủng loại, bán kính, khối lượng
6. Vẫn bài trên, thêm việc nhập và in dữ liệu ra màn hình