Ngôn ngữ lập trình C

B2: Cấu trúc của chương trình C

Khoa Công nghệ thông tin

Chủ đề chính

- Chương trình C
  - Cấu trúc chương trình C
    - *Tập ký tự.*
    - *Định danh.*
    - *Từ khóa.*
    - *Câu lệnh.*
    - *Chú thích.*
  - Biên dịch chương trình C
  - Bài tập thực hành

Chương trình máy tính?

- Là một chuỗi các chỉ dẫn để máy tính thực hiện các nhiệm vụ.
- Các chỉ dẫn được bộ xử lý thực hiện một cách tuần tự.
- Mỗi chỉ dẫn được mã hóa bằng các mã máy là các số nhị phân (ngôn ngữ máy).

Ngôn ngữ lập trình?

- Đưa ra các chỉ dẫn (code) cho bộ vi xử lý bằng ngôn ngữ thân thiện với lập trình viên.
- Trình biên dịch (compiler): thông dịch từ code thành ngôn ngữ máy để bộ vi xử lý có thể hiểu được.
- Mỗi ngôn ngữ lập trình có một cấu trúc mà lập trình viên cần tuân theo.

Ngôn ngữ lập trình C

- Lịch sử phát triển:
  - Ngôn ngữ lập trìnhC (NNLT C) ra đời tại phòng thí nghiệm BELL của tập đoàn AT&T (HoaKỳ).
  - Do Brian W. Kernighan và Dennis Ritchie phát triển vào đầu 1970, hoàn thành 1972.
  - C dựa trên nền các ngôn ngữ BCPL (Basic Combined Programming Language) và ngôn ngữ B.
  - Tên là ngôn ngữ C như là sự tiếp nối ngôn ngữ B.

Ngôn ngữ lập trình C

- Đặc điểm của NNLT C:
  - Là một ngôn ngữ lập trình hệ thống mạnh, linh hoạt.
  - Có thế mạnh trong xử lý các dạng dữ liệu số, văn bản, cơ sở dữ liệu.
- Thường được sử dụng để viết:
  - Các chương trình hệ thống như hệ điều hành (VD Unix: 90% viết bằng C, 10% viết bằng hợp ngữ).
  - Các chương trình ứng dụng chuyên nghiệp có can thiệp tới dữ liệu ở mức thấp như xử lý văn bản, xử lí ảnh…

Ngôn ngữ lập trình C

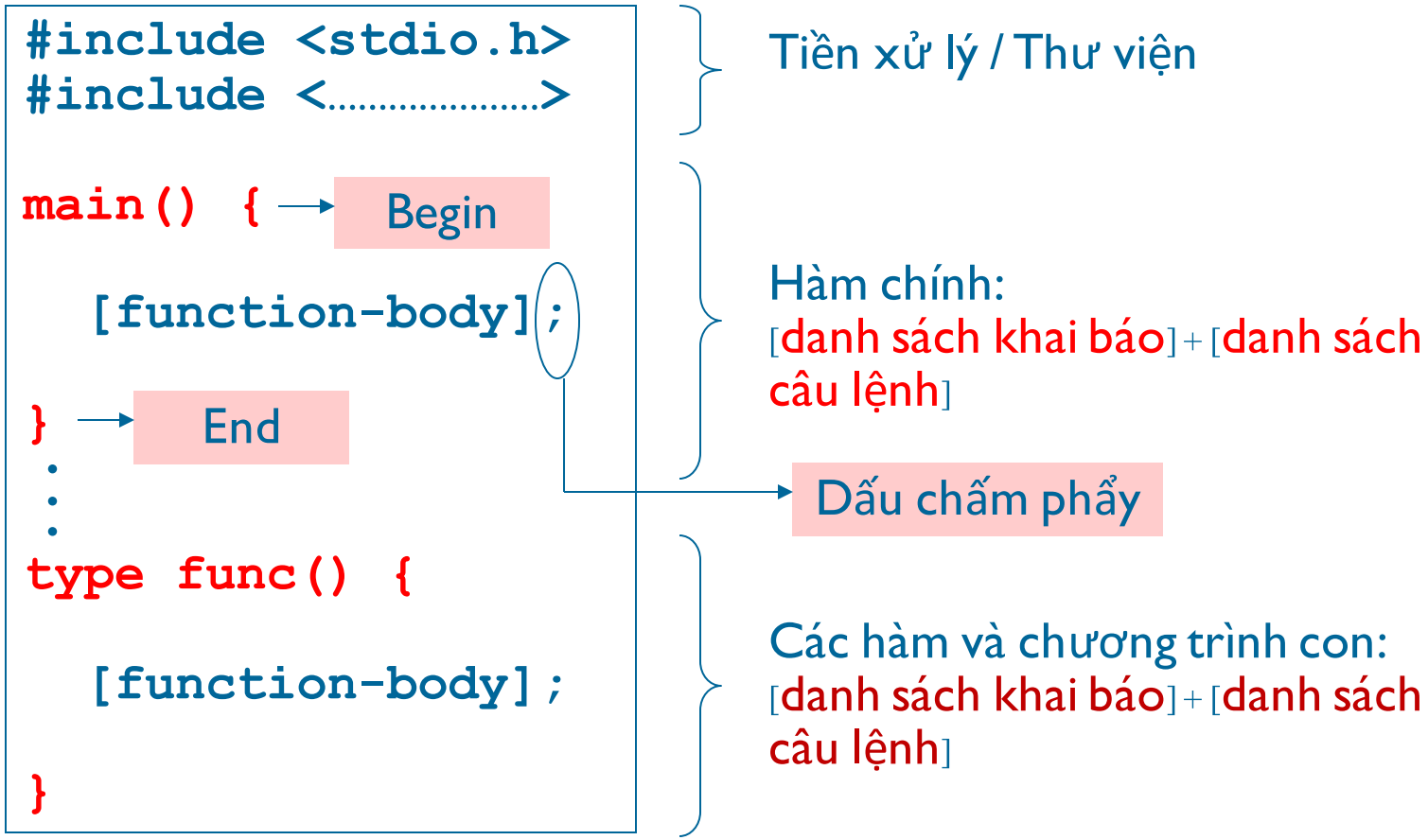
- 1978: C được giới thiệu trong phiên bản đầu của cuốn sách "The C programming language".
- Sau đó, C được bổ sung them những tính năng và khả năng mới → Đồng thời tồn tại nhiều phiên bản nhưng không tương thích nhau.
- Năm 1989, Viện tiêu chuẩn quốc gia Hoa Kỳ (American National Standards Institute-ANSI) đã công bố phiên bản chuẩn hóa của ngôn ngữ C: ANSI C hay C chuẩn hay C89.

Ngôn ngữ lập trình C

- Tất cả các phiên bản của ngôn ngữ C hiện nay đều tuân theo các mô tả đã được nêu ra trong ANSI C, sự khác biệt nếu có thì chủ yếu ở các thư viện bổ sung.
- Hiện nay cũng có nhiều phiên bản của ngôn ngữ C khác nhau, gắn liền với một bộ chương trình dịch cụ thể của ngôn ngữ C:
  - Turbo C++ và Borland C++ của BorlandInc.
  - MSC và VC của Microsoft Corp.
  - GCC của GNU project.
  - ICC của Intel Compiler.

Cấu trúc của ngôn ngữ C

- Định dạng chung:



Cấu trúc của ngôn ngữ C (tiếp)

- Chương trình C đầu tiên (hello.c)

```
#include <stdio.h>
int main() {
  printf("Hello World!\n");
   return 0;
}
```

Cấu trúc của ngôn ngữ C (tiếp)

- #include <stdio.h>
  - Khai báo sử dụng thư viện chuẩn vào-ra của C. Các thư viện khác của C: string.h, time.h, math.h, …
- int main()
  - Khai báo chương trình chính. Mỗi chương trình C chỉ có duy nhất một chương trình chính main()..
- { … }
  - Cú pháp đóng và mở của một khối lệnh.
- printf
  - hàm printf() đưa một chuỗi ký tự ra màn hình.
- return 0;
  - Kết thúc chương trình.

Cấu trúc của ngôn ngữ C (tiếp)

- Một ví dụ khác:

```
#include <stdio.h>
void main() {
    int sum;      /* khai báo biến */
                  /* sum là biến chứa tổng của hai
                     số nguyên */
    sum = 75 + 25;/* giá trị gán vào biến sum */
    printf("The sum of 75 and 25 is %d\n", sum);
}
```

```
The sum of 75 and 25 is 100
```

Các thành phần cơ bản trong NNLT C

- Tập ký tự:
  - Chương trình C được tạo ra từ các phần tử cơ bản là tập kí tự.
  - Các kí tự tổ hợp với nhau tạo thành các từ.
  - Các từ liên kết với nhau theo một quy tắc xác định để tạo thành các câu lệnh.
  - Từ các câu lệnh → tổ chức thành chương trình.

Các thành phần cơ bản trong NNLT C

- Tập ký tự (tiếp):

26 chữ cái hoa:	A B C . . . X Y Z
26 chữ cái thường:	a b c … x y z
10 chữ số:	0 1 2 3 4 5 6 7 8 9
Các kí hiệu toán	+ - * / = < >
Các dấu ngăn cách:	. ; , : space tab
Các dấu ngoặc:	( ) [ ] { }
Các kí hiệu đặc biệt:	_ ? $ & # ^ \ ! ' " ~

Các thành phần cơ bản trong NNLT C

- Từ khóa (keywords):
  - Là những từ có sẵn của ngôn ngữ và được sử dụng dành riêng cho những mục đích xác định.
  - Các từ khóa trong C được sử dụng để:
    - *Đặt tên cho các kiểu dữ liệu:* `int`, `float`, `double`, `char`, `struct`, `union`, …
    - *Mô tả các lệnh, các cấu trúc điều khiển:* `for`, `do`, `while`, `switch`, `case`, `if`, `else`, `break`, `continue`, …

Các thành phần cơ bản trong NNLT C

- Từ khóa (keywords): một số từ khóa hay dùng

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

Các thành phần cơ bản trong NNLT C

- Định danh / tên (identifier):
  - Là một dãy các kí tự dung để gọi tên các đối tượng trong chương trình.
  - Các đối tượng trong chương trình gồm có biến, hằng, hàm, kiểu dữ liệu, …
  - Có thể được đặt tên:
    - *Bởi ngôn ngữ lập trình (đó chính là các từ khóa)*
    - *Hoặc do người lập trình đặt.*

Các thành phần cơ bản trong NNLT C

- Định danh / tên (identifier):
  - Qui tắc đặt tên:
    - *Chỉ được gồm có: chữ cái, chữ số và dấu gạch dưới "_" (underscore).*
    - *Bắt đầu của định danh phải là chữ cái hoặc dấu gạch dưới, không được bắt đầu định danh bằng chữ số.*
    - *Định danh do người lập trình đặt không được trùng với từ khóa.*

Các thành phần cơ bản trong NNLT C

- Định danh / tên (identifier):
  - Ví dụ định danh/tên hợp lệ:

    i, x, y, a, b, _function, _MY_CONSTANT, PI, gia_tri_1
  - Ví dụ về định danh/tên không hợp lệ:

`1_a, 3d, 55x`	**bắt đầu bằng chữ số**
`so luong, ti le`	**có kí tự không hợp lệ (dấu cách – *space*) trong tên**
`int, char`	**trùng với từ khóa của ngôn ngữ C**

Các thành phần cơ bản trong NNLT C

- Định danh / tên (identifier):
  - Cách thức đặt định danh/tên:
    - Hằng số: chữ hoa
    - Các biến, hàm hay cấu trúc: Bằng chữ thường.
    - Nếu tên gồm nhiều từ thì ta nên phân cách các từ bằng dấu gạch dưới.
  - *Ví dụ*:

Định danh	Loại đối tượng
`HANG_SO_1, _CONSTANT_2`	hằng
`a, b, i, j, count`	biến
`nhap_du_lieu, tim_kiem, xu_li`	hàm
`sinh_vien, mat_hang`	cấu trúc

Các thành phần cơ bản trong NNLT C

- Các kiểu dữ liệu:
  - Là một tập hợp các giá trị mà một dữ liệu thuộc kiểu dữ liệu đó có thể nhận được.
  - Trên một kiểu dữ liệu ta xác định một số phép toán đối với các dữ liệu thuộc kiểu dữ liệu đó.
  - Ví dụ: Trong ngôn ngữ C có kiểu dữ liệu `int`. Một dữ liệu thuộc kiểu dữ liệu `int` thì:
    - *Là một số nguyên (integer)*
    - *Có thể nhận giá trị từ-32768 (-2^{15}) đến 32767 ($2^{15}-1$).*

Các thành phần cơ bản trong NNLT C

- Các kiểu dữ liệu:
  - Trên kiểu dữ liệu **int** ngôn ngữ C định nghĩa các phép toán số học đối với số nguyên như sau:
    - Đảo dấu: **-**
    - Cộng: **+**
    - Trừ: **-**
    - Nhân: *****
    - Chia lấy phần nguyên: **/**
    - Chia lấy phần dư: **%**
    - So sánh bằng: **= =**
    - So sánh lớn hơn: **>**
    - So sánh nhỏ hơn: **<**

Các thành phần cơ bản trong NNLT C

- Hằng số (constant):
  - Là đại lượng có giá trị không đổi trong chương trình.
  - Để giúp chương trình dịch nhận biết hằng ta cần nắm được cách biểu diễn hằng trong một chương trình C.
  - Ví dụ:
    - *Hằng dạng số: 2019, 3.14, …*
    - *Hằng ký tự ('A', 'c', …), hằng chuỗi ký tự ("Le Hoang Anh", "Hello world!"), ...*

Các thành phần cơ bản trong NNLT C

- Hằng số (constant): Hằng ký tự

Kí tự cần biểu diễn	Cách 1	Cách 2
Chữ cái A	'A'	65 hoặc 0101 hoặc 0x41
Dấu nháy đơn ‘	'\''	39 hoặc 047 hoặc 0x27
Dấu nháy kép “	'\"'	34 hoặc 042 hoặc 0x22
Dấu gạch chéo ngược \	'\\'	92 hoặc 0134 hoặc 0x5c
Kí tự xuống dòng	'\n'	
Kí tự NUL	'\0'	0 hoặc 00 hoặc 0x0

Các thành phần cơ bản trong NNLT C

- Biến (variable):
  - Là đại lượng mà giá trị có thể thay đổi trong chương trình.
  - Hằng và biến được sử dụng để lưu trữ dữ liệu, và phải thuộc một kiểu dữ liệu nào đó.
  - Tên biến và hằng được đặt theo quy tắc đặt tên cho định danh.

Các thành phần cơ bản trong NNLT C

- Câu lệnh (statement):
  - Diễn tả một hoặc một nhóm các thao tác trong giải thuật.
  - Chương trình được tạo thành từ dãy các câu lệnh.
  - Cuối mỗi câu lệnh đều có dấu chấm phẩy (;) để đánh dấu kết thúc câu lệnh.

Các thành phần cơ bản trong NNLT C

- Câu lệnh (statement):
  - Diễn tả một hoặc một nhóm các thao tác trong giải thuật.
  - Chương trình được tạo thành từ dãy các câu lệnh.
  - Cuối mỗi câu lệnh đều có dấu chấm phẩy (;) để đánh dấu kết thúc câu lệnh.

Các thành phần cơ bản trong NNLT C

- Câu lệnh (statement): Câu lệnh được chia thành 2 nhóm chính:
  - Nhóm các câu lệnh đơn:
    - *Không chứa câu lệnh khác.*
    - *Ví dụ: phép gán, phép cộng, phép trừ, …*
  - Nhóm các câu lệnh phức:
    - *Chứa câu lệnh khác trong nó.*
    - *Ví dụ: lệnh khối, các cấu trúc lệnh rẽ nhánh, cấu trúc lệnh lặp, …*
    - *Lệnh khối là một số các lệnh đơn được nhóm lại với nhau và đặt trong cặp dấu ngoặc nhọn {}*

Các thành phần cơ bản trong NNLT C

- Hàm (function):
  - Còn được gọi là chương trình con.
  - Những đoạn chương trình lặp đi lặp lại nhiều lần ở những chỗ khác nhau → Viết thành hàm để khi cần chỉ cần gọi ra chứ không phải viết lại toàn bộ.
  - Giải quyết một bài toán lớn thì chương trình của ta có thể rất lớn và dài → Chia thành các công việc nhỏ hơn được viết thành các hàm.

Các thành phần cơ bản trong NNLT C

- Chú thích (Comment):
  - Lời mô tả, giải thích vắn tắt cho một câu lệnh, một đoạn chương trình hoặc cả chương trình
  - Chỉ có tác dụng giúp chương trình viết ra dễ đọc và dễ hiểu hơn
  - Trình biên dịch sẽ tự động bỏ qua không dịch phần nội dung nằm trong phạm vi của vùng chú thích đó.
  - 2 cách chú thích
    - Trên 1 dòng: //
    - Trên nhiều dòng: /* */

Các thành phần cơ bản trong NNLT C

- Chú thích (Comment):
  - Cách 1:
    - Vùng bắt đầu từ // đến cuối dòng là vùng chú thích.
    - Ví dụ:

```
a = 5; b = 3; // Khoi tao gia tri cho cac bien nay
```

  - Cách 2:
    - Toàn bộ vùng bắt đầu nằm trong cặp kí hiệu /* */ là vùng chú thích.
    - Ví dụ:

```
/* Doan chuong trinh sau khai bao 2 bien nguyen va khoi tao
gia tri cho 2 bien nguyen nay */
int a, b;
a = 5; b = 3;
```

Các thành phần cơ bản trong NNLT C

- Tổng kết cấu trúc của một chương trình C:

Gồm 6 phần có thứ tự như sau:

Phần1: Khai báo tệp tiêu đề: #include
Phần 2: Định nghĩa kiểu dữ liệu mới: typedef ...
Phần 3: Khai báo các hàm nguyên mẫu
Phần 4: Khai báo các biến toàn cục
Phần 5: Hàm **main()**
Phần 6: Nội dung các hàm đã khai báo

Các thành phần cơ bản trong NNLT C

- Tổng kết cấu trúc của một chương trình C:
  - **Phần 1:** Khai báo tệp tiêu đề
    - *Thông báo cho chương trình dịch biết là chương trình có sử dụng những thư viện nào.*
    - *Ví dụ:* `#include<stdio.h>//thao tác vào ra.`
  - **Phần 2:** Định nghĩa các kiểu dữ liệu mới. Định nghĩa các kiểu dữ liệu mới (nếu cần) dung cho cả chương trình.
  - **Phần 3:** Khai báo các hàm nguyên mẫu giúp cho chương trình dịch biết được những thông tin cơ bản của các hàm sử dụng trong chương trình.
  - **Phần 4:** Khai báo các biến. Ví dụ:
    - `int a,b;`
    - `int tong, hieu, tich;`

Các thành phần cơ bản trong NNLT C

- Tổng kết cấu trúc của một chương trình C:
  - **Phần 5:** Hàm main()
    - *Khi thực hiện, chương trình sẽ bắt đầu bằng việc thực hiện các lệnh trong hàm main().*
    - *Trong hàm main() có thể có lệnh gọi tới các hàm khác.*
  - **Phần 6:** Nội dung của các hàm đã khai báo. Cài đặt (viết mã) cho các hàm đã khai báo nguyên mẫu ở phần 3.

Biên dịch chương trình với gcc

- gcc là GNU C Compiler
- Cho phép thực hiện trên HĐH Linux
- Ví dụ:
  - `gcc hello.c` (biên dịch file code `hello.c` thành file thực thi có tên `a.out`).
  - `gcc -o hello hello.c` (biên dịch file code `hello.c` thành file thực thi có tên `hello`).
  - `gcc -o hello hello.c other.c` (biên dịch file code `hello.c` và `other.c` thành file thực thi có tên `hello`).

Biên dịch chương trình với gcc

- Sử dụng hình thức trung gian: biên dịch các file code nguồn thành file object sau đó liên kết các file object thành file thực thi:

```
gcc –c hello.c
gcc –c other.c
gcc –o hello hello.o other.o
```

Bài tập 2.1

- Xem lại chương trình `hello.c` đã viết từ buổi trước và dùng `gcc compiler` để biên dịch thành file thực thi:

```
gcc hello.c
```

- Để xem chương trình thực thi như thế nào dùng lệnh:

```
./a.out
```

Bài tập 2.2: Debug và sửa lỗi chương trình

- Xóa một số ký tự trong file `hello.c` sau đó biên dịch lại để xem thay đổi.
- ***Thủ thuật***: nên mở 2 cửa sổ terminal và di chuyển đến cùng thư mục chứa file code:
  - 1 cửa sổ bật trình soạn thảo để soạn thảo code.
  - 1 cửa sổ để biên dịch và chạy file thực thi.

Nếu chương trình có lỗi

```
/* Your name – your class */
/* This is my first program in C */
#include <stdio.h>
main(                    không có ký tự ')'
{
  printf("Welcome to C Programming
  Introduction.\n");
}
```

- Khi biên dịch sẽ nhận được thông báo:
- `hello.c : in function 'main'`
- `hello.c:4: parse error before '}'`

Dòng có lỗi

Cách để sửa lỗi chương trình ?

- Mở lại file "`hello.c`";
- Tìm đến dòng báo lỗi và sửa lại cho đúng. *Lưu ý*: lỗi có thể phát sinh do các dòng lân cận;
- Lưu lại file;
- Biên dịch và chạy lại chương trình thực thi.

Bài tập 2.2

- Dùng `gcc compiler` để biên dịch file code `hello.c` trong bài tập trước thành một chương trình thực thi có tên `sayhello`

```
gcc –o sayhello hello.c
```

- Chạy file `sayhello`:

```
./sayhello
```

Bài tập 2.3

- Copy file hello.c thành hello1.c và sửa lại thành nội dung như sau:

```
/* Your name – your class */
/* This is my second program in C */

#include <stdio.h>
void main()
{
   printf("Welcome to C");
   printf("Programming  Introduction.\n");
}
```

- Biên dịch thành file thực thi có tên `hello1`.
- Chạy file thực thi `hello1` và so sánh sự khác biệt khi chạy file `sayhello` trước?

Bài tập 2.4

- Viết chương trình như dưới đây, biên dịch, chạy file thực thi và so sánh sự khác biệt với các chương trình trước:

```
/* Your name – your class */
/* This is my second program in C */

#include <stdio.h>
main()
{
   printf("Welcome to C\n");
   printf("Programming  Introduction.\n");
}
```

Bài tập 2.5

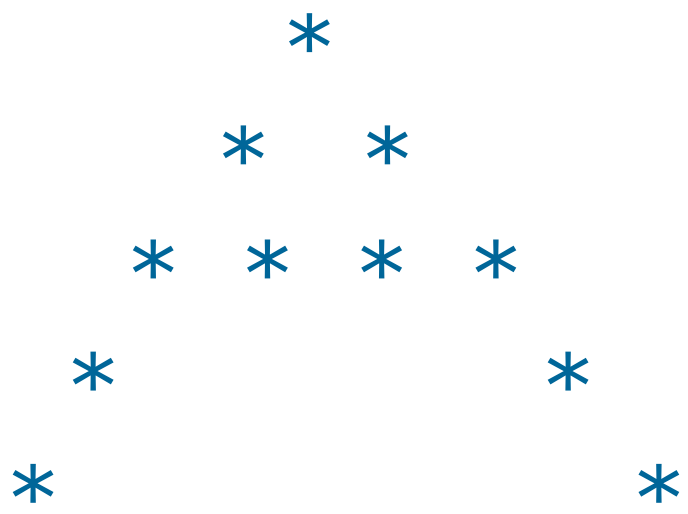
- Viết chương trình in ra màn hình thông tin giới thiệu bản thân.
- Ví dụ:

```
**************************
My name is Nguyen Van A.
Nice to meet you.
Hope you will have happy time

**************************
```

Bài tập 2.6

- Viết chương trình in ra màn hình có dạng như sau:



Bài tập 2.7

- Nhập lại đoạn chương trình dưới đây và lưu lại với tên `pi.c`. Biên dịch và chạy file thực thi để xem kết quả. Các bạn nghiên cứu và cho biết mục đích của chương trình này là để làm gì.

```
#include <stdio.h>
#define PI 3.142

main()
{
  double r, c, ac, as, v;
  r = 5.678;
  printf("Radius = %f\n", r);
  c = 2.0 * PI * r;
  printf("Circle's circumference = %f\n", c);
  ac = PI * r * r;
  printf("Circle's area = %f\n", ac);
  as = 4.0 * PI * r * r;
  printf("Sphere's area = %f\n", as);
  v = 4.0/3.0 * PI * r * r * r;
  printf("Sphere's volume = %f\n", v);
}
```

Bài tập về nhà 2.1

- Viết chương trình in ra màn hình thông tin thẻ sinh viên của bạn trên từng dòng.

Bài tập về nhà 2.2

- Nghiên cứu lại bài tập 2.7 ở trên lớp và thực hiện lại cho đối tượng là hình vuông và khối lập phương.
- Cạnh của hình vuông có kích thước: 10.

Bài tập về nhà 2.3

- Viết chương trình in ra màn hình hướng dẫn cách giải phương trình bậc 2.