TÓM LƯỢC BÀI GIẢNG
NHẬP MÔN LẬP TRÌNH

(Vũ Quốc Hoàng, vqhoang@fit.hcmus.edu.vn, FIT-HCMUS, 2021)

BÀI 2
TÍNH TOÁN

Chủ đề

- Tính toán
- Số nguyên và số thực
- Toán tử và biểu thức
- Biến và lệnh gán
- Dùng hàm và module
- Mã giả

Tài liệu

[1] Vũ Quốc Hoàng, *Bí kíp luyện Lập trình C (Quyển 1)*, hBook, 2017.

[2] Tony Gaddis, *Starting out with C++ From Control Structures through Objects*, Pearson, 8th edition, 2015.

[3] Vũ Quốc Hoàng, *Bí kíp luyện Lập trình nhập môn với Python*, hBook, 2020.

- Đọc kĩ: Bài 1.2, 1.3 [1]; Bài 2, 3 [3]
- Đọc thêm: Bài 1.4 [1], Phần 1.4, 1.5, 1.6, 2.3, 2.4, 2.5, 2.6, 2.9, 2.14, 2.15, 2.17 [2]

Kiến thức

- **Tính toán** (computation) là việc thao tác trên các đối tượng số hoặc không phải số nhằm tạo ra các kết quả mong đợi. **Điện toán** (computing) là việc tính toán tự động bằng **máy tính** (computer). **Tính toán số học** (arithmetic calculation) là việc tính toán trên các **số** (number).
- **Dữ liệu** là những thứ mà chương trình thao tác, xử lý, tính toán. **Kiểu dữ liệu** (data type) xác định dạng dữ liệu và các thao tác hay **phép toán** (operation) có thể thực hiện trên dữ liệu. Kiểu dữ liệu cơ bản nhất trên máy là **số nguyên** (integer) và **số thực** (real number). Các số thường được **biểu diễn ở dạng thập phân** (decimal representation).
- Các phép toán số học cơ bản là **phép đối** (negation), **phép cộng** (addition), **phép trừ** (subtraction), **phép nhân** (multiplication) và **phép chia** (division). Phép chia thường gồm 3 dạng là: chia thông thường (được số thực), **chia nguyên** (floor division) và **chia lấy dư** (modulo). Các phép toán cơ bản thường được mô tả bằng các **toán tử** (operator).
- Các phép toán số học nâng cao hơn là **tính căn** (square root, *n*th root), **tính mũ** (exponentiation), **tính log** (logarithm), **lượng giác** (trigonometry), ... thường được hỗ trợ bằng các **hàm** (function).
- Hàm là các **dịch vụ** (service) tính toán (và xử lý) mà chương trình có thể dùng qua lời **gọi hàm** (calling function) với **tên hàm** (function name) xác định hàm, các **đối số** (argument) cung cấp dữ liệu cho hàm và nhận **giá trị trả về** (return value) từ hàm. Cú pháp chung của lời gọi hàm là:

```
<Tên hàm>(<các đối số>)
```

- **Biểu thức** (expression), gồm các **toán hạng** (operand), các toán tử, các lời gọi hàm, mô tả một **giá trị** (value) là kết quả có được sau khi biểu thức được **lượng giá** (evaluate).
- **Độ ưu tiên** (precedence) và **tính kết hợp** (associativity) của các toán tử, cùng với các cặp ngoặc tròn xác định thứ tự thực hiện các phép toán trong biểu thức.
- Số toán hạng mà toán tử cần (hay số đối số mà hàm cần) được gọi là **số ngôi** (arity) của toán tử (hay của hàm). Đa số các toán tử (cộng, trừ, …) là **toán tử 2 ngôi** (binary operator); toán tử đối là **toán tử 1 ngôi** (unary operator).
- **Biến** (variable) là phương tiện để nhớ, tức là để lưu trữ dữ liệu của chương trình. Dữ liệu có thể được "đưa vào" hay gán cho biến bằng **lệnh gán** (assignment statement) và được dùng lại trong các biểu thức bằng tên biến. Cú pháp chung của lệnh gán là:

  `<Tên biến> = <Biểu thức>`
- Biến chứa kết quả tính toán (thao tác) trung gian được gọi là **biến tạm** (temporary variable).
- Dữ liệu trong biến có thể thay đổi (biến có thể được đặt lại để chứa dữ liệu khác). Để phân biệt, các số hay chuỗi cố định trong chương trình được gọi là các **hằng** (literal) vì chúng mô tả các dữ liệu cố định. Hằng cũng có thể được dùng trong biểu thức như biến.
- Tên biến và tên hàm là các **tên** (name) hay **định danh** (identifier). Chúng có chung qui tắc đặt tên, hơn nữa, ta thường đặt tên biến bằng danh từ (mô tả ý nghĩa của biến, biến đó chứa gì?) và đặt tên hàm bằng động từ (mô tả thao tác của hàm, hàm đó làm gì?).
- **Module** là tập hợp các hàm (và các thứ khác) được tổ chức theo nhóm, cung cấp các dịch vụ liên quan. Tập các module cung cấp sẵn cùng ngôn ngữ được gọi là **thư viện chuẩn** (standard library) của ngôn ngữ (như thư viện chuẩn Python, thư viện chuẩn C/C++). Các ngôn ngữ thường yêu cầu khai báo (hoặc lệnh) dùng module trước khi dùng các hàm trong module.
- **Mã giả** (pseudocode) là mô tả công việc (các bước làm) được viết ngắn gọn, rõ ràng bằng ngôn ngữ tự nhiên cô đọng và/hoặc các kí hiệu, công thức Toán. Mã giả gần với con người hơn là máy và không có qui chuẩn nên máy không làm theo được (do đó nó được gọi là mã giả). Muốn máy làm theo được, ta phải viết cụ thể hơn nữa bằng ngôn ngữ qui chuẩn như các ngôn ngữ lập trình, khi đó, ta có "mã thật".

Kĩ năng

- Dùng được các chức năng cơ bản của IDLE và Dev C++
- Thành thạo các bước phát triển một chương trình
- Thành thạo việc dùng toán tử, hàm (có sẵn), biểu thức, biến và lệnh gán
- Viết được các chương trình tính toán/xử lý trên số nguyên và số thực
- Diễn đạt được các bước làm các việc đơn giản bằng mã giả
- Ý thức và rèn luyện được việc viết chương trình đẹp, ngăn nắp, rõ ràng

Mã nguồn minh họa

Mã nguồn 1. (Khuôn chương trình "nhập - tính toán/xử lý - xuất")

Tiếng Việt	Python

A: Bạn sinh năm nào?
B: 2002.
A: (tính toán: 2021 – 2002 được 19)
A: Bạn 19 tuổi.
	`print("Bạn sinh năm nào?")`
	`năm = int(input())`
	`tuổi = 2021 - năm`
	`print("Bạn %d tuổi" % tuổi)`

C	**C++**

```c
#include <stdio.h>


int main()
{
	int year, age;

	printf("Ban sinh nam nao? ");
	scanf("%d", &year);
	age = 2021 - year;
	printf("Ban %d tuoi", age);

	return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main()
{
	int year, age;

	cout << "Ban sinh nam nao? ";
	cin >> year;
	age = 2021 - year;
	cout << "Ban " << age << " tuoi";

	return 0;
}
```

Giải thích và bình luận thêm:

- Tính toán là công việc cơ bản, thường xuyên và phổ biến nhất của con người; thuật ngữ này có nghĩa từ hẹp (tính toán số) đến rất rộng (thao tác, xử lý bất kì)
- Python, C/C++ đều dùng "tên" `int` cho kiểu số nguyên; nhớ rằng, không cần khai báo biến trong Python nhưng phải khai báo trong C/C++
- Python, C/C++ đều dùng chung các kí hiệu toán tử phổ biến là + (cộng), - (trừ), * (nhân), / (chia); toán tử đối cũng được kí hiệu là - (1 ngôi, phân biệt với toán tử trừ, 2 ngôi)
- Trong C, biến dùng trong `scanf` phải có dấu và (&) đằng trước (không có dấu & trước biến trong `printf`)
- Trong Python, toán tử % khi dùng với chuỗi sẽ giúp "định dạng" chuỗi
- Trong C++, << là **toán tử xuất** (output operator), >> là **toán tử nhập** (input operator)
- Một số ngôn ngữ (như Python) hỗ trợ số nguyên có độ lớn bất kì; một số ngôn ngữ (như C/C++) chỉ hỗ trợ số nguyên trong một phạm vi; kiểu `int` của C/C++ hỗ trợ các số nguyên từ khoảng -2 tỉ đến 2 tỉ.
- Ta gọi con số nguyên mô tả trong chương trình như `2021` ở trên là **hằng số nguyên** (integer literal) vì nó mô tả con số nguyên cố định (số 2021); các biến nguyên như `year`, `age` chứa số nguyên nhưng có thể thay đổi để chứa số nguyên khác
- `int` là viết tắt của integer; `scanf` là viết tắt của "scan formatted"; chữ d trong %d là viết tắt của decimal (thập phân)
- Rõ ràng, để chương trình trên chạy được mọi lúc (không chỉ năm 2021), chương trình cần có cách nào đó để tính (tìm/lấy) thời gian hiện tại

Mã nguồn 2. (Khuôn chương trình "tính toán phức tạp")

Mã giả (Pseudocode)	**Python**

Nhập 3 cạnh (hợp lệ) của tam giác a, b, c
 	
Tính nửa chu vi: $s = \frac{a+b+c}{2}$
Tính diện tích: $A = \sqrt{s(s-a)(s-b)(s-c)}$
 	
Xuất diện tích A
	`import math`
	
	`a = float(input("Nhập cạnh a: "))`
	`b = float(input("Nhập cạnh b: "))`
	`c = float(input("Nhập cạnh c: "))`
	
	`s = (a + b + c)/2`
	`A = math.sqrt(s*(s - a)*(s - b)*(s - c))`
	
	`print("Diện tích tam giác là %.2f" % A)`

C	C++

`#include <stdio.h>`
`#include <math.h>`
 	
`int main()`
`{`
`  double a, b, c;`
 	
`  printf("Nhap 3 canh tam giac: ");`
`  scanf("%lf %lf %lf", &a, &b, &c);`
 	
`  double s = (a + b + c)/2;`
`  double A = sqrt(s*(s - a)*(s - b)*(s - c));`
 	
`  printf("Dien tich la %.2lf", A);`
 	
`  return 0;`
`}`
	`#include <iostream>`
	`#include <cmath>`
	`#include <iomanip>`
	`using namespace std;`
	
	`int main()`
	`{`
	`  double a, b, c;`
	
	`  cout << "Nhap 3 canh tam giac: ";`
	`  cin >> a >> b >> c;`
	
	`  double s = (a + b + c)/2;`
	`  double A = sqrt(s*(s - a)*(s - b)*(s - c));`
	
	`  cout << "Dien tich la ";`
	`  cout << setprecision(2) << fixed << A;`
	
	`  return 0;`
	`}`

Giải thích và bình luận thêm:

- Python chỉ có kiểu số thực `float`; C/C++ có 2 kiểu số thực là `float` và `double` nhưng nên dùng kiểu `double` để có độ chính xác cao hơn; (kiểu `float` của Python thực ra là `double`!)
- Các ngôn ngữ lập trình đều dùng dấu chấm (`.`) để phân cách **phần nguyên** (integer part) và **phần lẻ** (fractional part) của số thực (chứ không dùng dấu phẩy (`,`) như Việt Nam ta hay dùng)
- Trong Python, toán tử `/` kí hiệu phép chia thông thường (được số thực), toán tử `//` kí hiệu phép chia nguyên, toán tử `%` kí hiệu phép chia lấy dư (`%` cũng là toán tử định dạng chuỗi); trong C/C++, toán tử `/` thực hiện phép chia nguyên nếu cả 2 toán hạng là số nguyên, ngược lại thì chia thường, toán tử `%` kí hiệu phép chia lấy dư
- Python hỗ trợ toán tử mũ (`**`); C/C++ thì không (dùng hàm `pow` trong module `math`)
- Độ ưu tiên các toán tử thường là đối, mũ, nhân/chia, cộng/trừ; đa số các toán tử có tính kết hợp trái (làm từ trái qua phải khi cùng độ ưu tiên); toán tử mũ có tính kết hợp phải (làm từ phải qua trái)
- Có thể tính căn trong Python bằng toán tử mũ (`** 0.5`) hoặc dùng hàm `sqrt` của module `math`
- Chuỗi định dạng `%.2f` (hay `%.2lf`) yêu cầu xuất số thực lấy 2 chữ số lẻ sau dấu chấm thập phân; để đặt định dạng trong C++ thì "phiền phức" hơn (dùng `setprecision` và `fixed`, …)
- Python phân biệt các **hàm dựng sẵn** (built-in function) và các hàm trong module; hàm dựng sẵn thì được dùng mà không cần `import`
- Trước khi dùng module, Python yêu cầu nạp module bằng lệnh nạp (`import`); C/C++ yêu cầu khai báo dùng module (`#include`); lưu ý, `#include` không là lệnh mà là **chỉ thị tiền xử lý** (preprocessor directive) (sẽ học kĩ sau); chỉ thị `#include` trong C thường có đuôi `.h` sau tên module
- Các hàm dựng sẵn của Python đã biết tới giờ là: `print` (dùng để xuất), `input` (dùng để nhập), `int` (dùng để tính số nguyên từ chuỗi biểu diễn thập phân), `float` (dùng để tính số thực từ chuỗi biểu diễn thập phân)
- Các hàm đã biết của C/C++ tới giờ là: `printf`, `gets`, `scanf` (trong `stdio.h`) và các hàm trong `math.h` (`cmath`)
- Trong C++, `cout`, `cin`, `endl`, `fixed` là các "biến đối tượng" được định nghĩa trước

- Module `math` cung cấp các hàm tính toán số nâng cao như căn (`sqrt`), mũ (`pow`), logarithm (`log`), lượng giác (`sin`, `cos`, ...), …
- Hầu hết các ngôn ngữ lập trình (kể cả Python, C, C++) đều làm việc với số thực "khá tệ"
- Ta gọi con số thực mô tả trong chương trình như `0.01` hay `1e-2` là **hằng số thực** (floating point literal)
- `float` là viết tắt của floating point number; `math` là viết tắt của mathematics; `sqrt` là viết tắt của square root; `double` là viết tắt của "double-precision floating-point format"; `iostream` là viết tắt của "input output stream"; `iomanip` là viết tắt của "input output manipulator"; chữ `f` trong `%f` là viết tắt của float; chữ `lf` trong `%lf` là viết tắt của "long float"
- Làm sao kiểm tra 3 cạnh nhập vào là 3 cạnh hợp lệ của một tam giác?

Bài tập

1. Sử dụng các chức năng cơ bản của các IDE: IDLE, Dev C++.
2. Làm các bài tập sau bằng C, C++: 1.2.4, 1.3, 1.4.2, 1.4.3 [1].
3. Làm thêm các bài tập sau bằng Python: Bài 2, 3 [3].