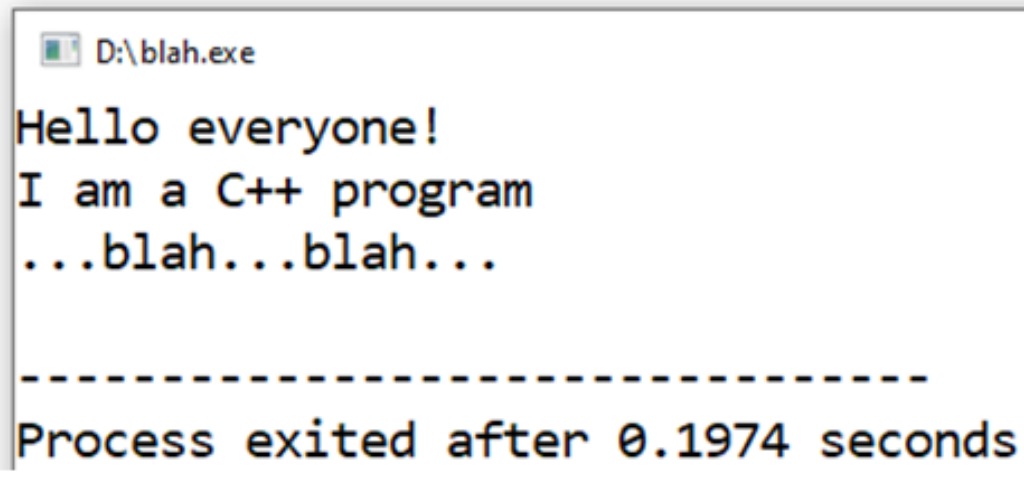


# NHẬP MÔN LẬP TRÌNH (23CLC05)

Buổi 2 – 09/10/23

# Khuôn chương trình 1 có thú vị không?

Tiếng Python	Kết quả thực thi
<pre>print("Hello everyone!") print("I am a Python program") print("...blah...blah...")</pre>	<pre>&gt;&gt;&gt; ===== RESTART: D:/blah.py ===== Hello everyone! I am a Python program ...blah...blah... &gt;&gt;&gt;</pre>
Tiếng C++	
<pre>#include &lt;iostream&gt; using namespace std;  int main() {     cout &lt;&lt; "Hello everyone!\n";     cout &lt;&lt; "I am a C++ program\n";     cout &lt;&lt; "...blah...blah...\n"; }</pre>	 <pre>D:\blah.exe Hello everyone! I am a C++ program ...blah...blah...  ----- Process exited after 0.1974 seconds</pre>

# Khuôn chương trình 2!

Tiếng Anh	Python
A: What's your name? B: Hoang. A: Hello Hoang.	<pre># A: chương trình, B : người dùng print("Bạn tên gì?") tên = input() print("Chào", tên)</pre>
C	C++
<pre>// A: program, B: user #include &lt;stdio.h&gt;  int main() {     char name[100];      printf("What's your name? ");     gets(name);     printf("Hello %s", name); }</pre>	<pre>// A: program, B: user #include &lt;iostream&gt; using namespace std;  int main() {     string name;      cout &lt;&lt; "What's your name? ";     cin &gt;&gt; name;     cout &lt;&lt; "Hello " &lt;&lt; name; }</pre>

# Ghi chú là gì?

- **Ghi chú** (comment) là những chú thích, giải thích, mô tả thêm trong mã nguồn
- Ghi chú không tham gia vào chương trình mà giúp việc đọc hiểu mã nguồn dễ hơn
- Ghi chú trên một dòng
  - Python: `# . . .`
  - C/C++: `// . . .`
- Ghi chú trên nhiều dòng
  - C/C++: `/* . . . */`

# Chương trình cần tương tác với người dùng!

- Chương trình tương tác với **người dùng** (user) khi chạy
- Chương trình nhận thông tin **đầu vào** (input) từ người dùng
- Chương trình xuất kết quả **đầu ra** (output) cho người dùng

# Chương trình nhận thông tin đầu vào từ người dùng bằng cách nào?

- Chương trình cần biết cách **hỏi** người dùng
- Và **nhớ** thông tin người dùng cung cấp

# Chương trình ghi nhớ bằng cách nào?

- **Biến** (variable) là phương tiện để ghi nhớ, tức là để lưu trữ **thông tin** (information) hay **dữ liệu** (data), của chương trình
- Chương trình dùng biến để chứa dữ liệu nhập từ người dùng (và các dữ liệu khác)

# Khai báo biến là gì?

- **Khai báo biến** (variable declaration) là việc báo cho biết **tên biến** (variable name) và **kiểu dữ liệu** (data type) mà biến sẽ chứa
- Nhiều ngôn ngữ như C, C++ **yêu cầu phải khai báo biến** trước khi biến được dùng; một số ngôn ngữ như Python thì không cần
- Cú pháp khai báo biến trong C/C++:

**<kiểu dữ liệu> <tên biến>;**

- Cú pháp khai báo **biến chuỗi** (string variable) trong C/C++:

**char <tên biến>[<số kí tự tối đa>;**

- Cú pháp khai báo biến chuỗi trong C++:

**string <tên biến>;**



# Cần lưu ý gì về tên biến?

- Tên biến phải được đặt đúng qui tắc và nên ngắn gọn, rõ ràng, gợi nhớ đến ý nghĩa của biến
- Đa số các ngôn ngữ (như Python, C, C++) **phân biệt chữ hoa chữ thường** (case sensitive)

# Chương trình hỏi người dùng bằng cách nào?

- **Lệnh nhập** (input statement) yêu cầu máy nhập một chuỗi nào đó từ người dùng (người dùng gõ chuỗi từ bàn phím và nhấn Enter để nhập)
- **Chuỗi nhập** (input string) sau đó có thể được “chuyển đổi” thành các dạng dữ liệu khác (nếu cần)
- Cú pháp của lệnh nhập
  - Python: `<biến> = input()`
  - C: `gets(<biến chuỗi>);`
  - C++: `cin >> <biến chuỗi>;`

# Chương trình xuất kết quả đầu ra cho người dùng bằng cách nào?

- **Lệnh xuất** (output statement) yêu cầu máy xuất ra một chuỗi (hoặc một giá trị) nào đó
- Cú pháp của lệnh xuất (đã học)
  - Python: `print(...)`
  - C: `printf(...);`
  - C++: `cout << ...;`
- Xuất chuỗi chứa trong biến chuỗi
  - Python: `print(<biến chuỗi>)`
  - C: `printf("... %s ...", <biến chuỗi>);`
  - C++: `cout << <biến chuỗi>;`

# Cộng điểm

- Làm bài tập 1.2.1 (Tài liệu C)
- Làm bài tập 1.2.2 (dùng Python)

Có lập trình viên nào mới viết đã được ngay chương trình chạy đúng không?

- Không!

# Lỗi chương trình là gì?

- **Lỗi cú pháp** (syntax error) là lỗi chương trình không viết đúng theo qui định của ngôn ngữ
  - Chương trình không chạy được khi có lỗi cú pháp
  - Các IDE hỗ trợ việc kiểm tra lỗi cú pháp và thông báo trợ giúp sửa lỗi
- Chương trình chạy được (không có lỗi cú pháp) vẫn có thể có lỗi lúc chạy, gọi là **lỗi thực thi** hay **lỗi lúc chạy** (runtime error)
- **Lỗi logic** (logical error) là lỗi chương trình mà không có “biểu hiện” gì
- Lỗi thực thi và lỗi logic thường được gọi chung là **bug**

# Viết chương trình đúng đã đủ chưa?

- Ta không chỉ cần viết chương trình cho đúng mà nên viết cho đẹp, rõ ràng và dễ hiểu!
- Nên viết chương trình theo **phong cách viết** (writing style, programming style, coding style) chung mà đa số các lập trình viên hay dùng
- Cần **ng nghiêm túc rèn luyện** việc này khi mới học lập trình để tránh các thói quen xấu khó sửa sau này
  - chẳng hạn, bằng cách bắt chước các mã nguồn minh họa trong môn học
- Đặc biệt, **ên dùng tiếng Anh càng nhiều càng tốt!**

# Công việc quan trọng nhất của máy tính và chương trình là gì?

- **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)



# Khuôn chương trình 3!

Tiếng Việt	Python
A: Bạn sinh năm nào? B: 2002. A: (tính toán: 2021 – 2002 được 19) A: Bạn 19 tuổi.	<pre>print("Bạn sinh năm nào?") năm = int(input()) tuổi = 2021 - năm print("Bạn %d tuổi" % tuổi)</pre>
C	C++
<pre>#include &lt;stdio.h&gt;  int main() {     int year, age;      printf("Ban sinh nam nao? ");     scanf("%d", &amp;year);     age = 2021 - year;     printf("Ban %d tuoi", age); }</pre>	<pre>#include &lt;iostream&gt; using namespace std;  int main() {     int year, age;      cout &lt;&lt; "Ban sinh nam nao? ";     cin &gt;&gt; year;     age = 2021 - year;     cout &lt;&lt; "Ban " &lt;&lt; age &lt;&lt; " tuoi"; }</pre>

# Dữ liệu là gì?

- **Dữ liệu** (data) 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 biến nguyên

- Cú pháp khai báo biến nguyên trong C/C++:

```
int <tên biến nguyên>;
```

- Nhập giá trị vào biến nguyên trong C/C++:

```
scanf("%d", &<biến nguyên>);
```

- Nhập giá trị vào biến nguyên trong Python:

```
<biến> = int(input())
```

- Xuất giá trị chứa trong biến nguyên trong C/C++:

```
printf("...%d...", <biến nguyên>);
```

# Các phép toán số học nào được xem là cơ bản?

- 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):  $-$ ,  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $//$ ,  $\%$

# Có cần giải lao không?

- Giải lao đến 10h00

# Biểu thức là gì?

- **Biểu thức** (expression) là dãy kí hiệu gồm các hằng, biến, các toán tử (và 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** (operand) 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)

# Các toán tử số học trong Python

<b>Độ ưu tiên (Precedence)</b>	<b>Toán tử (Operator)</b>	<b>Số ngôi (Arity)</b>	<b>Tính kết hợp (Associativity)</b>
1 (cao nhất)	mũ (**)	2	Phải
2	đổi (-)	1	(Phải)
3	nhân (*), chia (/) chia nguyên (//) chia lấy dư (%)	2	Trái
4	cộng (+), trừ (-)	2	Trái

# Hằng với biến khác nhau thế nào?

- Các số hay chuỗi cố định trong chương trình được gọi là các **hằng** (literal) vì chúng mô tả các dữ liệu cố định
- Biến chứa dữ liệu có thể thay đổi (biến có thể được đặt lại để chứa dữ liệu khác)
- Biến và hằng được xem là các biểu thức đơn giản nhất (biểu thức đơn) để tạo nên các biểu thức phức tạp hơn
  - Giá trị của biến là dữ liệu đang chứa trong biến
  - Giá trị của hằng là dữ liệu cố định được mô tả bởi hằng



# Lệnh gán là gì?

- 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

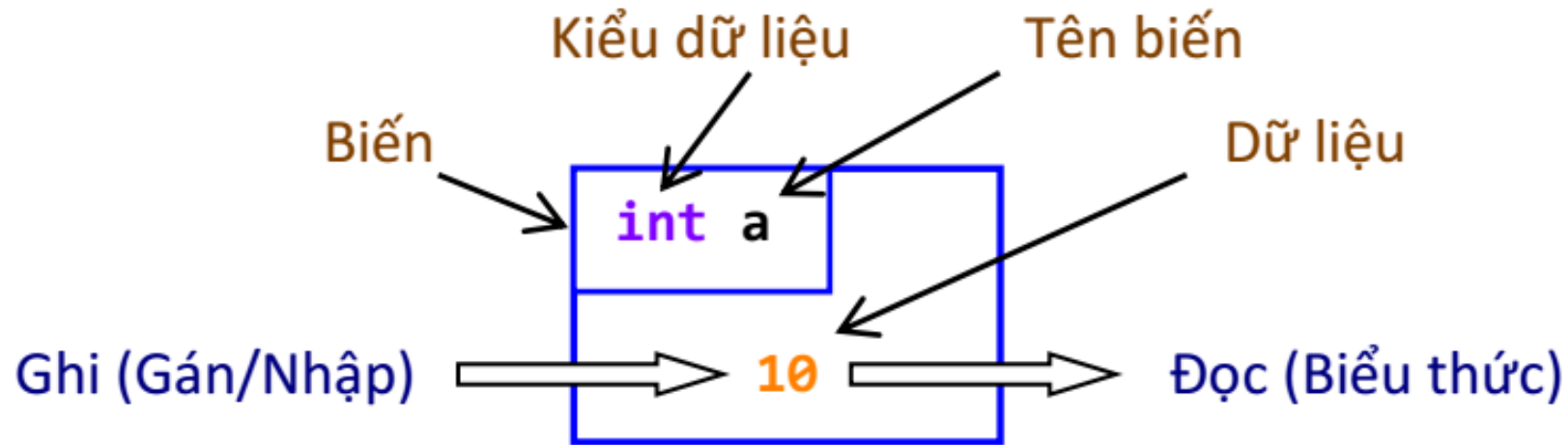
- Cú pháp chung của lệnh gán là:

**<tên biến> = <biểu thức>**

- Lệnh gán được xem là một lệnh đơn (do đó cần kết thúc bằng dấu ; trong C/C++)

# Biến như là “hộp chứa dữ liệu”!

Hình 1.2.1 – Biến như là “hộp chứa dữ liệu”



# Cộng điểm

- Làm bài tập 1.2.4 (Tài liệu C)

# Các phép toán số học nào được xem là nâng cao?

- Các phép toán số học nâng cao là **tính căn** (square root,  $n$ th root), **tính mũ** (exponentiation), **tính log** (logarithm), **lượng giác** (trigonometry), ...

# Khuôn chương trình 4! (1)

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

# Mã giả là gì?

- **Mã giả** (pseudocode) là bản mô tả công việc (các bước làm) được viết ngắn gọn, rõ ràng bằng ngôn ngữ tự nhiên cô đọng và 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 (“mã thật”)

# Khuôn chương trình 4! (2)

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

# Các biến thực

- Cú pháp khai báo biến thực trong C/C++:

```
double <tên biến thực>;
```

- Nhập giá trị vào biến thực trong C/C++:

```
scanf("%lf", &<biến thực>);
```

- Nhập giá trị vào biến thực trong Python:

```
<biến> = float(input())
```

- Xuất giá trị chứa trong biến thực trong C/C++:

```
printf("...%lf...", <biến thực>);
```



# Hàm là gì?

- Hàm là các **dịch vụ** (service) tính toán (và xử lý) mà chương trình có thể dùng qua lời **gọi hàm** (calling function) và nhận **giá trị trả về** (return value) từ hàm
- Cú pháp chung của lời gọi hàm là:  

**<tên hàm>(<các đối số>)**
- **Tên hàm** (function name) xác định hàm, các **đối số** (argument) cung cấp dữ liệu cho hàm
- Các đối số là dãy các biểu thức phân cách bởi dấu phẩy (,)
- Các phép toán số học nâng cao thường được hỗ trợ bằng các hàm

# Module và thư viện là 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

# Các module nào hay dùng?

- Tra cứu các module hay dùng trong C
- Tra cứu các module hay dùng trong C++
- Tra cứu các module hay dùng trong Python

# Từ khóa với định danh khác nhau thế nào?

- Tên biến và tên hàm là các **tên** (name) hay **định danh** (identifier)
- Cần đặt tên theo đúng qui tắc của ngôn ngữ
- 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?)
- **Từ khóa** (keyword) là các tên đặc biệt, dành riêng mà chương trình không được phép dùng để đặt tên (cho biến, hàm, ...)
- Tra cứu danh sách từ khóa trong C/C++, Python

# Cộng điểm

- Làm bài tập 1.3.2 (Tài liệu C)
- Làm bài tập 1.3.3 (về nhà)
- Làm bài tập 1.4.3 (về nhà)