**Kỹ Thuật Lập Trình**

**Học kỳ 2 – Năm học 2017-2018**

**Lab 4 – Cấu trúc điều khiển lặp**

# MỤC TIÊU CỦA BÀI THỰC HÀNH

* Hiểu và vận dụng các cấu trúc điều khiển lặp để giải quyết trọn vẹn 1 bài toán hay một phần các bài toán có cần đến dạng cấu trúc này
* Luyện kỹ năng giải quyết vấn đề thông qua quá trình phân rã bài toán thành các công việc con được lặp đi lặp lại.

# BÀI TẬP BẮT BUỘC

**Câu 1:**

Viết chương trình in ra bảng mã ASCII của các ký tự chữ và số.

**Hướng dẫn:**

1. **Dữ liệu:** Chương trình phải có biến ô nhớ để chứa ký tự cần in ra giá trị trong bảng mã ASCII:

* **ch**: kiểu char

1. **Giải thuật:**

Sử dụng vòng lặp để lặp đầy đủ các trường hợp ký tự sau:

* Ký tự số : ‘0’ – ‘9’
* Ký tự chữ cái thường: ‘a’ – ‘z’
* Ký tự chữ cái in hoa: ‘A’ – ‘Z’

Tìm hiểu cách ép kiểu để xuất giá trị trong bảng mã ASCII của các ký tự

Gợi ý: Ban đầu gán biến ô nhớ **ch** là ký tự **‘0’**. Từ ‘0’ đến ‘9’có 10 ký tự, do đó đây là vòng lặp xác định trước số lần lặp. Ta dùng vòng lặp **for** lặp 10 lần để lần lượt in ký tự ch và giá trị tương ứng trong bảng mã ASCII, tiếp đến tăng **ch** lên 1 đơn vị để chuẩn bị cho lần in tiếp theo. SV làm tương tự cho chữ cái thường và in hoa.

**Câu 2:**

Viết chương trình cho phép:

1. Nhập số N (nguyên dương), nhập cho đến khi đúng thì thôi.
2. Tính tổng bình phương của các số lẻ từ 1 đến N bằng cách lặp

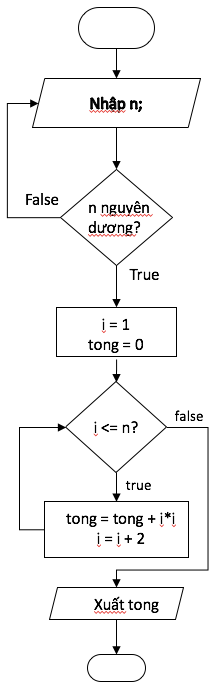
In ra N và kết quả - chọn cách in cho thẩm mỹ

**Hướng dẫn:**

1. **Dữ liệu:** chương trình phải có biến (ô nhớ) để chứa các dữ liệu sau đây:

* Biến lưu số n nguyên dương: (có thể) sử dụng unsigned long
* Biến lưu tổng bình phương: (có thể) sử dụng unsigned long long

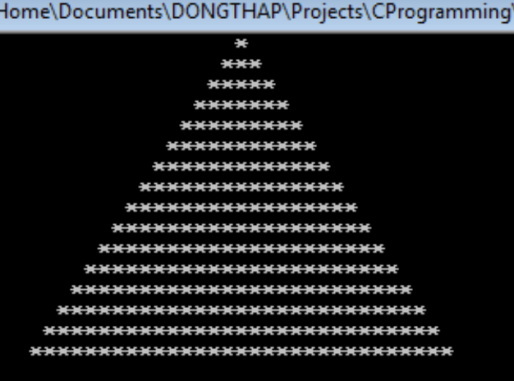
1. **Giải thuật:**

****

* Vì số lần lặp đã cho trước → sử dụng cấu trúc lặp for cho việc tính tổng bình phương.
* Để kiểm tra giá trị nhập có phải là số nguyên hay không, sử dụng cin.fail().

**Câu 3:**

Viết chương trình in ra tam giác cân rỗng và đặc

**Hướng dẫn:**

1. **Dữ liệu:** Ta cần một biến N để biểu diễn độ dài một cạnh của tam giác cần in và chiều cao H của tam giác.
2. **Giải thuật:**

Nhận thấy số dòng phải in chính là chiều cao của tam giác, dựa vào N chúng ta có thể tính được độ dài đường cao H. Điều này cho ta biết rằng ta có thể sử dụng vòng lặp để in ra từng dòng cho tam giác. Vì chúng ta đã biết được số lần lặp cố định dựa vào kích thước cạnh tam giác, vòng lặp for sẽ thích hợp trong trường hợp này.

Quan sát tam giác cân đặc, dòng đầu tiên, kí tự chính giữa sẽ là dấu ‘\*’ cách một khoảng so với lề là N/2-1 khoảng trắng, các kí tự ‘\*’ dòng 2 sẽ cách lề N/2-2 khoảng trắng, tương tự cho các dòng tiếp theo. Và độ dài một dòng được biểu diễn bằng số dấu ‘\*’ sẽ tăng đều từ 1, 3, 5, đến N. Sinh viên có thể sử dụng hàm setw kết hợp với 2 vòng lặp lồng nhau để in ra tam giác đặc.

Với trường hợp tam giác rỗng. Mỗi hàng trừ hàng đầu tiên và hàng cuối cùng thì chỉ có 2 ký tự ‘\*’. Hàng đầu tiên chỉ có một ký tự ‘\*’ và hàng cuối cùng sẽ có N ký tự ‘\*’. Trong trường hợp này ta chỉ cần sử dụng một vòng lặp để in ra phần thân tam giác, mỗi hàng in 2 kí tự ‘\*’ sao cho đúng vị trí, hàng trên cùng và hàng dưới cùng sẽ xử lý riêng. Sử dụng quan sát đối với tam giác đặc phía trên và hàm setw để hoàn chỉnh bài in tam giác rỗng.

**Câu 4:**

Viết chương trình tính số hạng thứ n của dãy Fibonaci

F(0) = 1

F(1) = 1

F(n) = F(n-1) + F(n-2), n>=2

**Hướng dẫn:**

1. **Dữ liệu:** Chương trình phải có các biến (ô nhớ) để chứa các dữ liệu:

* **n**: kiểu float (để kiểm tra yêu cầu nhập vào chính xác n phải là một số nguyên không âm)
* **previous, current, next**: kiển int

1. **Giải thuật:**

* Kiểm tra điều kiện nhập vào của giá trị n nhập vào. Yêu cầu phải kiểm tra được n là số nguyên không âm. Mọi giá trị nhập khác (chẳng hạn ký tự, số âm, ký tự,…) đều phải báo lỗi
* Khởi tạo các biến previous, current, next với giá trị khởi tạo thích hợp
* Lặp từ 0 đến n và cập nhật giá trị các biến nhớ previous, current, next thích hợp
* Xuất kết quả giá trị của số Fibonacci thứ n. Lưu ý đảm bảo kiểm tra

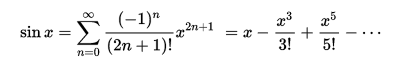
F(0) = F(1) = 1

**Câu 5:**

Viết chương trình tính sin(x)

**Hướng dẫn:**

Hàm sin(x) có thể được xấp xỉ theo Taylor:



1. **Dữ liệu:** x là số thực, vì vậy ta có thể chọn kiểu dữ liệu cho nó là: float hoặc double.
2. **Giải thuật:**

Xác định giá trị Epsilon (một số rất nhỏ), ví dụ Epsilon = 10-10.

Nhận thấy phép tổng bên vế phải của công thức có thể phân tách thành n toán hạng / phần tử.

Để tính sin(x) ta cần cộng tích lũy các phần tử bằng cách dùng vòng lặp, mỗi lần lặp ta sẽ tính giá trị phần tử và cộng tiếp vào biến kết quả. Vòng lặp sẽ dừng khi giá trị của phần tử đủ nhỏ (so sánh với epsilon). Ta nhận thấy rằng giá trị của epsilon càng nhỏ thì độ chính xác tính toán xấp xỉ của ta càng cao.

Mỗi phần tử cần tính toán các thành phần sau:

* A = x2n+1
* B = (2n + 1)!
* C = (-1)n

Phần tử = C \* A / B

Tổng tất cả các phần tử ở mỗi lần lặp chính là sin(x) cần tính.

*Lưu ý*: góc x trong công thức đơn vị là radian, vì vậy sinh viên cần đổi x từ radian sang đơn vị độ trước khi tiến hành các vòng lặp tính toán.

**Câu 6: Dùng phương pháp xác suất tính giá trị gần đúng của số π**

Phương pháp này là phương pháp tính toán với nền tảng là lý thuyết xác suất thống kê. Thiết lập mối liên hệ giữa bài toán và mô hình xác suất, dùng máy tính thực hiện việc lấy mẫu, để nhận được nghiệm gần đúng của bài toán.



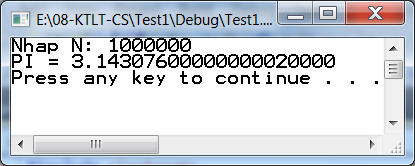
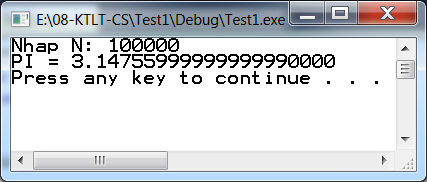


Giả sử có một đường tròn bán kính 1, thì diện tích của hình tròn là π, còn diện tích phần gạch chéo là π/4. Như vậy chỉ cần tính ra diện tích phần gạch chéo là có thể tính được giá trị π.

Cách tính cụ thể như sau:

Hình vuông ở góc trên bên phải có diện tích là 1, phần gạch chéo có diện tích là π/4. Dùng vòng lặp for để sinh ra *N* điểm ngẫu nhiên (cho phép người sử dụng nhập vào số lần lặp *N*). Tọa độ *x* và *y* của các điểm có giá trị nằm trong đoạn [0, 1]. Mỗi khi sinh ra một điểm thì kiểm tra xem điểm này có nằm trong vùng gạch chéo hay không. Thống kê tổng số điểm rơi vào vùng gạch chéo và lưu vào biến *sum*. Vì các điểm được tạo ra ngẫu nhiên nên chúng sẽ phân bố đều trong hình vuông, hay nói cách khác, ta có công thức sau: *sum*/*N* = diện tích phần gạch chéo/diện tích hình vuông, hay *sum*/*N* = π/4 → π = 4\**sum*/*N*.

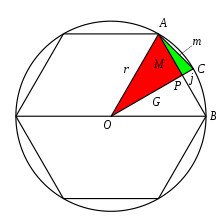
Ví dụ: Hình dưới minh họa kết quả chạy chương trình với số điểm lấy mẫu là 100000 và 1000000, chương trình cho hai kết quả khác nhau. Thậm chí, ở hai lần chạy khác nhau, nhập số lượng điểm lấy mẫu giống nhau, thì chương trình cũng cho kết quả khác nhau.



**Câu 7: Dùng chu vi đa giác đều để tính** **giá trị gần đúng của số π**

Như đã thấy ở trên, dùng phương pháp xác suất để tính giá trị số π, thì kết quả không ổn định. Người xưa đã sử dụng phương pháp sau để tính giá trị π: dùng chu vi của hình đa giác đều để xấp xỉ chu vi của hình tròn ngoại tiếp đa giác đều này. Archimedes dùng đa giác đều 96 cạnh và tính được giá trị của π chính xác đến 3 con số ở phần thập phân. Lưu Huy (người Trung Quốc) dùng đa giác đều 3072 cạnh và tính được giá trị của π chính xác đến 5 con số ở phần thập phân. Ludolph Van Ceulen dùng đa giác đều 262 cạnh và tính được giá trị của π chính xác đến 35 con số ở phần thập phân.

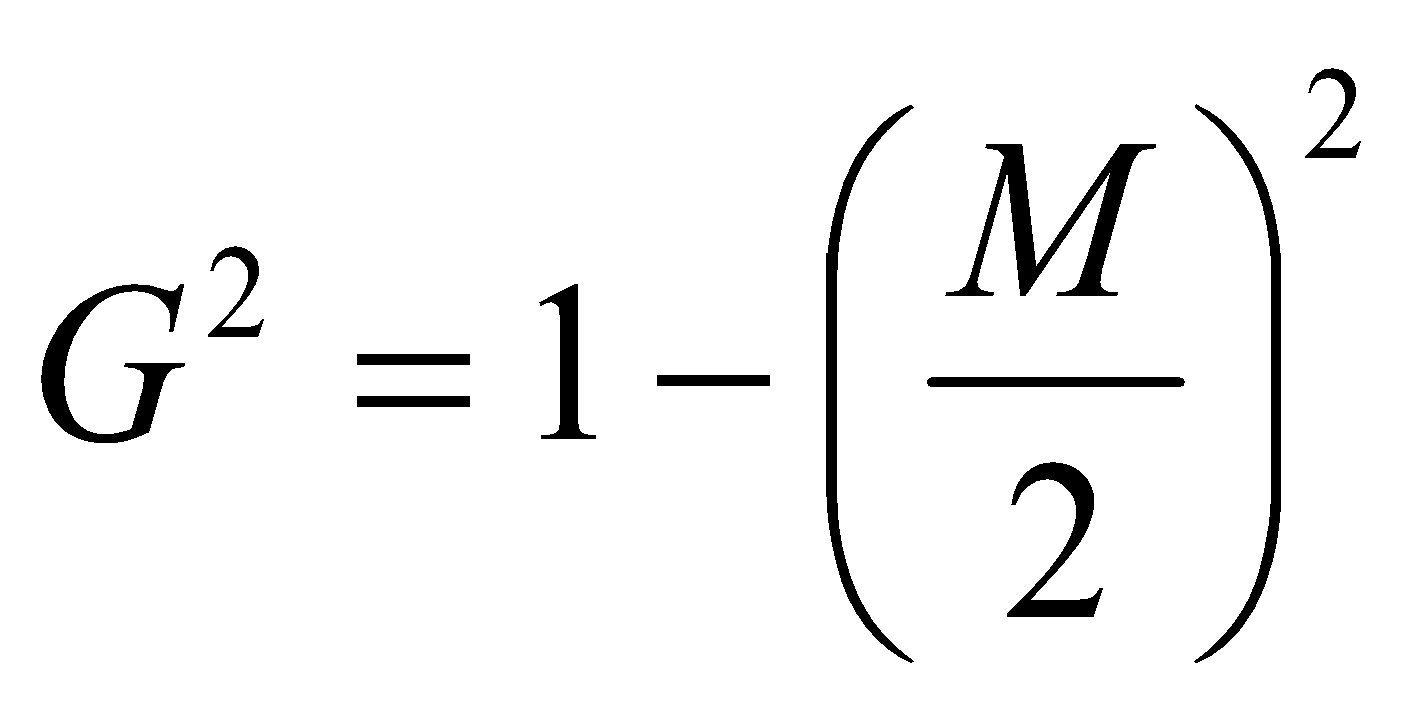
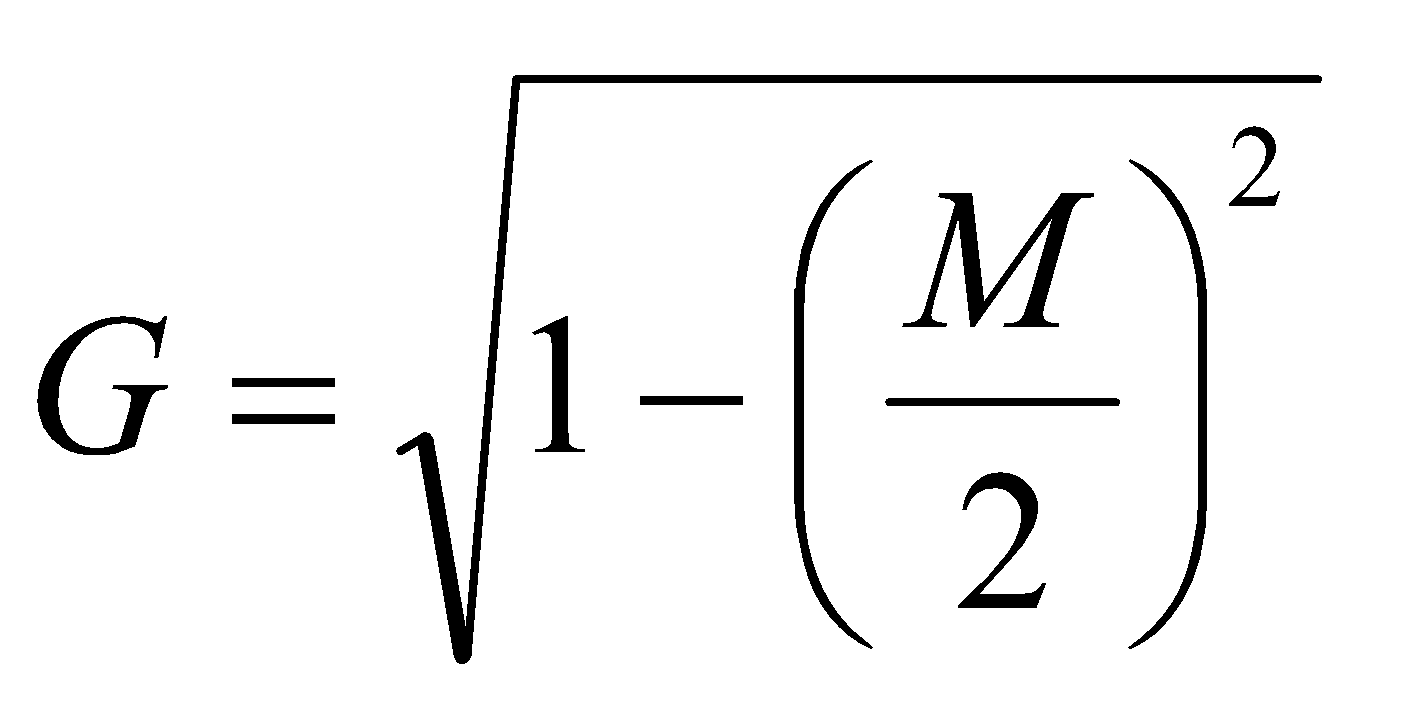
Giả sử có một đường tròn bán kính là 1, và một hình lục giác đều nội tiếp trong đường tròn này, như vậy cạnh của hình lục giác này bằng 1 và chu vi của nó là 6. Dùng chu vi hình lục giác này xấp xỉ chu vi đường tròn tức là 6 = 2\* π → π = 3.

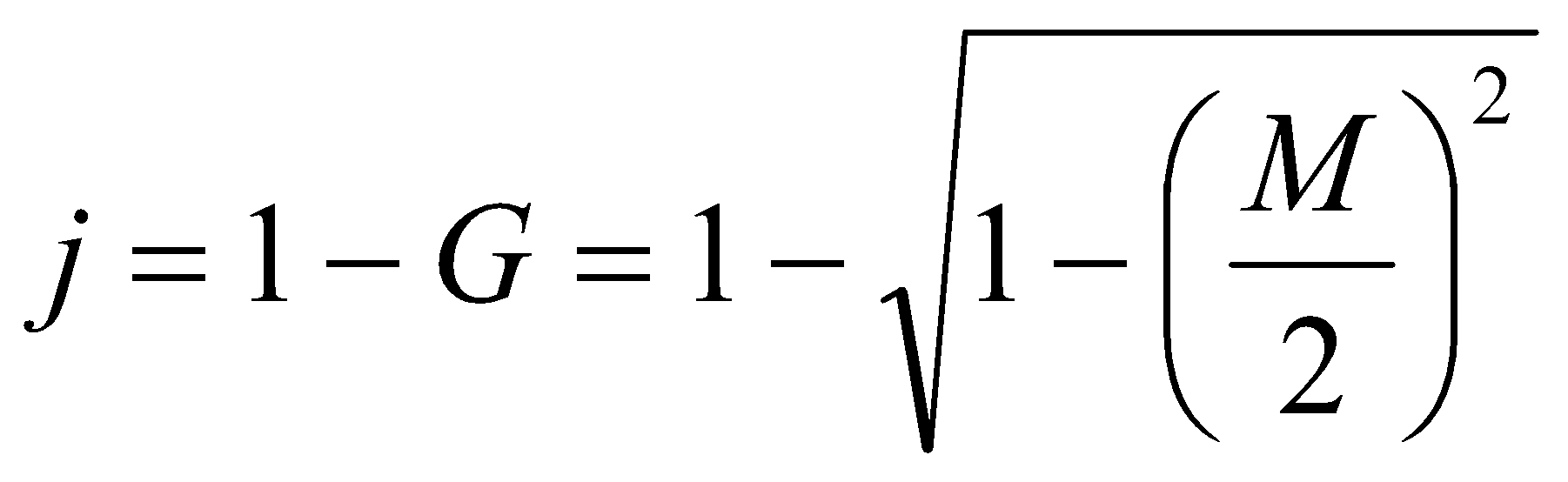


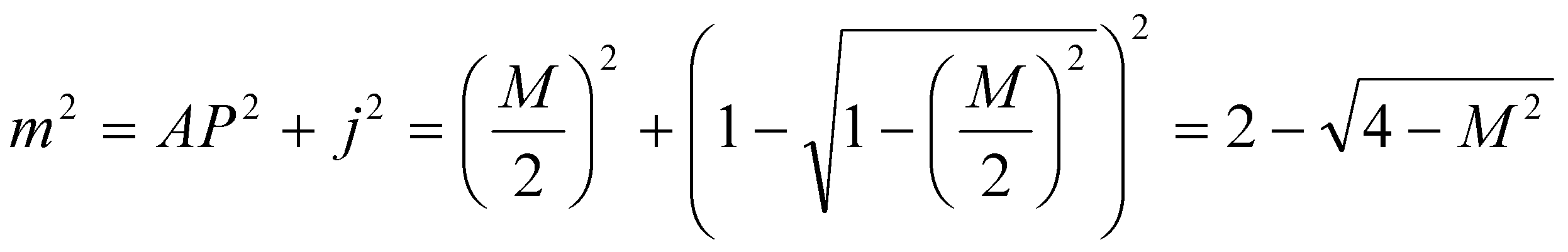
Bây giờ, cho hình đa giác đều 12 cạnh nội tiếp bên trong hình tròn (xem hình vẽ trên)

Đặt OP = G, PC = j, AB = M (M là chiều dài cạnh đa giác đều 6 cạnh), AC = m (m là chiều dài đa giác đều 12 cạnh).

Tam giác AOP, APC là tam giác vuông, theo định lý Pitago ta có:

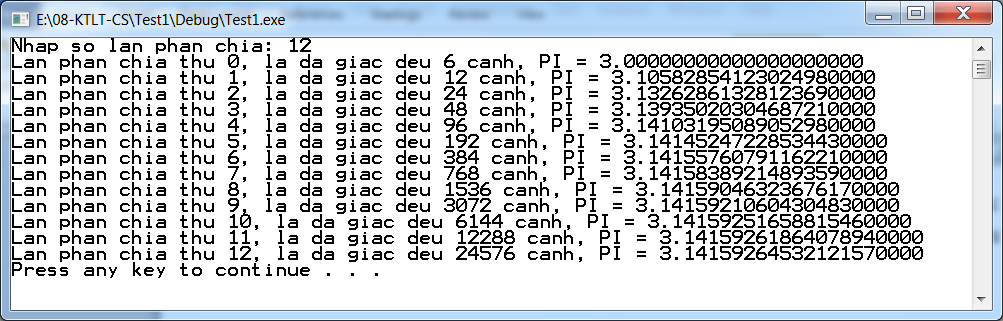
→ 



 (1)

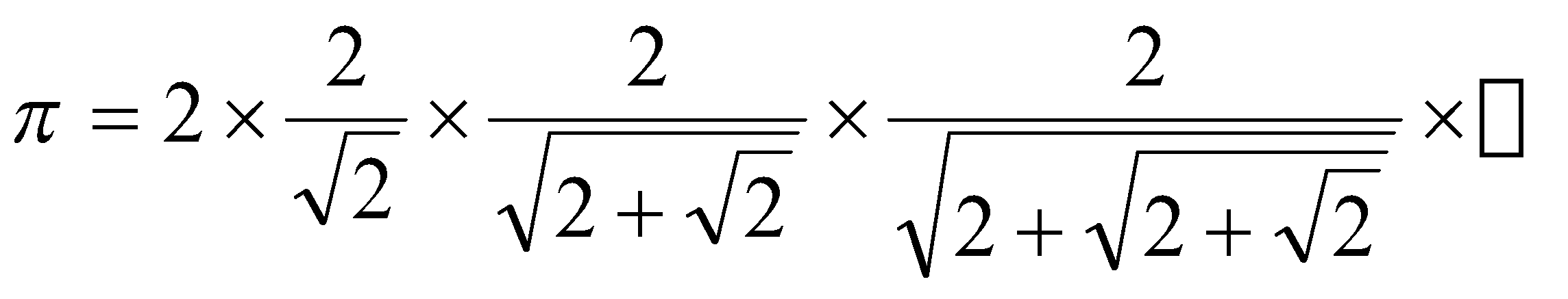
Công thức (1) cho thấy quan hệ giữa chiều dài cạnh đa giác đều 6 cạnh và đa giác đều 12 cạnh. Trường hợp tổng quát, công thức này phản ánh quan hệ giữa chiều dài đa giác đều 6\**n* cạnh và đa giác đều 6\*(*n*+1) cạnh.

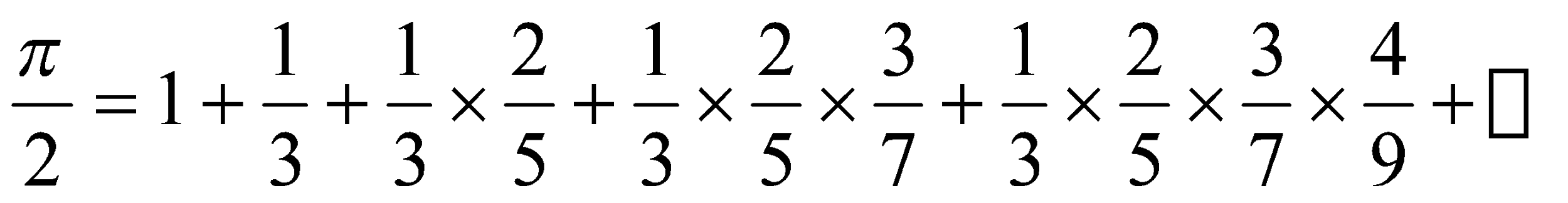
Hãy viết chương trình cho phép người sử dụng nhập vào số lần phân chia bắt đầu từ đa giác đều 6 cạnh. Sau mỗi lần phân chia, số cạnh của đa giác đều lại tăng lên gấp đôi và giá trị số π tính được lại càng chính xác.

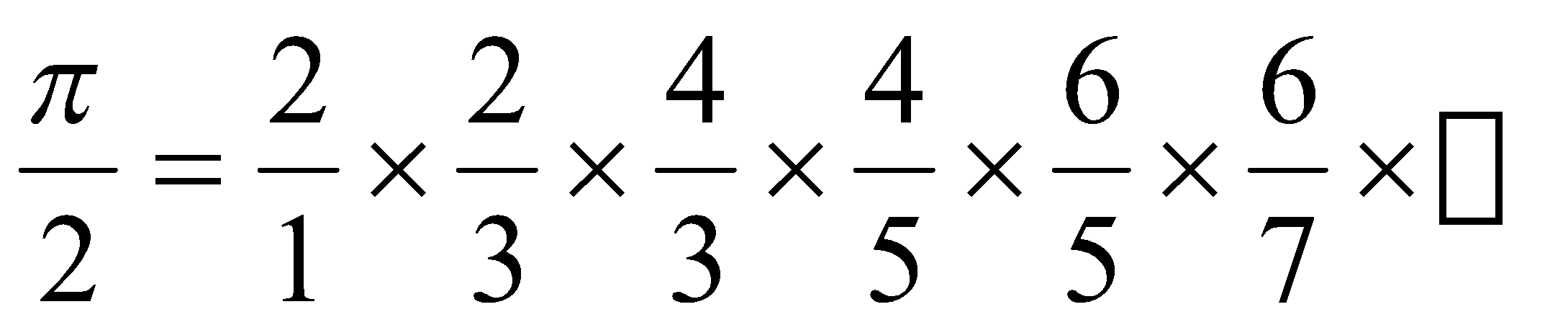


**Câu 8: Sử dụng công thức để tính giá trị số π**

Có khá nhiều công thức để tính giá trị số π. Trong bài thực hành này, sinh viên sử dụng ba công thức sau đây để tính giá trị của số π.





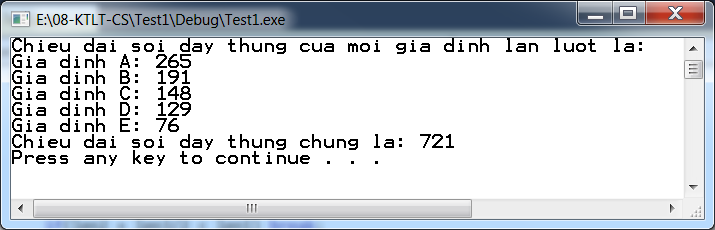


**Câu 9: Sử dụng vòng lặp lồng nhau**

Có 5 gia đình A, B, C, D, E ở vùng núi cùng dùng chung 1 giếng nước, mỗi nhà góp 1 đoạn dây thừng rồi buộc lại với nhau để thành một sợi dây dài múc nước giếng lên. Biết rằng:

* Sợi dây thừng chung có chiều dài **bằng** chiều dài sợi dây thừng gia đình B **cộng với** hai lần chiều dài sợi dây thừng của gia đình A.
* Sợi dây thừng chung có chiều dài **bằng** chiều dài sợi dây thừng gia đình C **cộng với** ba lần chiều dài sợi dây thừng của gia đình B.
* Sợi dây thừng chung có chiều dài **bằng** chiều dài sợi dây thừng gia đình D **cộng với** bốn lần chiều dài sợi dây thừng của gia đình C.
* Sợi dây thừng chung có chiều dài **bằng** chiều dài sợi dây thừng gia đình E **cộng với** năm lần chiều dài sợi dây thừng của gia đình D.
* Sợi dây thừng chung có chiều dài **bằng** chiều dài sợi dây thừng gia đình A **cộng với** sáu lần chiều dài sợi dây thừng của gia đình E.
* Chiều dài của các sợi dây thừng đều là số nguyên đơn vị tính là inch.

Viết chương trình để tìm ra chiều dài sợi dây thừng của mỗi gia đình và chiều dài của sợi dây thừng chung.



**3 BÀI TẬP KHÔNG BẮT BUỘC**

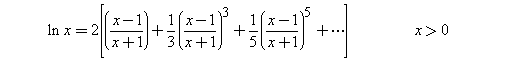
**Câu 1:**

Viết chương trình tính các hàm khác trong trong slide bài giảng: cos(x), tan(x), log(x)

**Hướng dẫn:**

Hàm cos(x) và ln(x) có thể được xấp xỉ theo Taylor:





1. **Dữ liệu:** x là số thực, vì vậy ta có thể chọn kiểu dữ liệu cho nó là: float hoặc double.
2. **Giải thuật:**

Sinh viên tiến hành phân tích tương tự như câu 10 để tính toán các các toán hạng / phần tử ở mỗi lần lặp.

Việc nhận thấy quy luật biến đổi của mỗi phần tử trong các lần lặp là rất quan trọng.

Tan(x) ta có thể tính bằng cách lấy sin(x) / cos(x).

*Lưu ý*: góc x trong công thức cos(x) đơn vị là radian, vì vậy sinh viên cần đổi x từ radian sang đơn vị độ trước khi tiến hành các vòng lặp tính toán.

**Câu 2:**

Viết chương trình tính giá trị đa thức bậc N

**Hướng dẫn:**

Dạng đa thức tính toán có thể là: A0 + A1X + A2X2 + A3X3 + A4X4 + …

1. **Dữ liệu:** Người dùng cần nhập:

* Bậc N của đa thức: có thể chọn kiểu dữ liệu là số nguyên dương.
* Giá trị của biến X: có thể chọn kiểu dữ liệu là số thực.
* N lần hệ số của mỗi đơn thức: có thể chọn kiểu dữ liệu là số thực.

1. **Giải thuật:**

Xét đa thức sau: anxn + an-1xn-1 + an-2xn-2 + … + a1x + a0

Chương trình yêu cầu người dùng nhập các hệ số dưới dạng chuỗi “an an-1 an-2 … a1 a0”. Sinh viên dùng stringstream để đọc các số trong đó.

Tiếp theo chương trình yêu cầu nhập x, nhập đến khi nào đúng là số thực thì thôi.

Cách tính đa thức:

* Giải thuật dùng cách biểu diễn sau cho đa thức bậc n:

[[an\*x + an-1]\* x + an-2]\*x + …

* Do đó, chúng ta tích luỹ giá trị của biểu thức bởi biểu thức sau:

px = px\*x + hệ\_số

* Khởi động, px = 0 => Biểu thức trên lần được đánh giá cho ra “hệ số” bậc n
  + Lần kế tiếp, sẽ là an\*x + an-1
  + Lần nữa là, [an\*x + an-1]\*x = an\*x2 + an-1\*x
  + … cuối cùng sẽ có biểu thức mong muốn.

Ví dụ cụ thể cho giải thuật:

Đa thức cần tính: 3x2 + 4x + 5

Người dùng nhập: 3 4 5

Người dùng nhập x = 2

Khởi động p\_x = 0

Lần lặp 1:

* Đọc hệ số // coeff = 3
* p\_x = p\_x\*x + coeff // p\_x = 0\*2 + 3 = 3

Lần lặp 2:

* Đọc hệ số // coeff = 4
* p\_x = p\_x\*x + coeff // p\_x = 3\*2 + 4 = 10

Lần lặp 3:

* Đọc hệ số // coeff = 5
* p\_x = p\_x\*x + coeff // p\_x = 10\*2 + 5 = 25

Ta tính được giá trị cần tìm là p\_x = 25