**ĐẠI HỌC QUỐC GIA TP HCM**

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

🙡★🙣



BÁO CÁO ĐỒ ÁN

MÔN LẬP TRÌNH SYMBOLIC TRONG TRÍ TUỆ NHÂN TẠO

NGÀNH KHOA HỌC MÁY TÍNH

**CHƯƠNG TRÌNH GIẢI TOÁN ĐẠO HÀM, TÍCH PHÂN, GIỚI HẠN**

**GIẢNG VIÊN HƯỚNG DẪN:**

Nguyễn Đình Hiển

**NHÓM SINH VIÊN THỰC HIỆN:**

1. Đinh Trọng Hậu - MSSV: 15520204
2. Huỳnh Mai Minh Hiếu - MSSV: 15520222

TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2019

**MỤC LỤC**

[**CHƯƠNG 1.** **TỔNG QUAN** 2](#_Toc11147538)

[**1.1.** **Phạm vi đề tài** 2](#_Toc11147539)

[**1.2.** **Mục tiêu, yêu cầu** 2](#_Toc11147540)

[**1.3.** **Nền tảng công nghệ, công cụ** 3](#_Toc11147541)

[**CHƯƠNG 2.** **PHƯƠNG PHÁP GIẢI BÀI TOÁN** 4](#_Toc11147542)

[**2.1.** **Giải toán Calculus trong Maple** 4](#_Toc11147543)

[**2.2.** **Xây dựng package cho chương trình** 5](#_Toc11147544)

[**2.2.1.** **Hàm giải chi tiết từng bước** 5](#_Toc11147545)

[**2.2.2.** **Hàm chuyển đổi lời giải thích (hint) sang tiếng Việt** 6](#_Toc11147546)

[**2.2.3.** **Hàm xuất kết quả ra tập tin** 8](#_Toc11147547)

[**2.2.4.** **Đóng gói package thành thư viện** 10](#_Toc11147548)

[**CHƯƠNG 3.** **THIẾT KẾ, CÀI ĐẶT CHƯƠNG TRÌNH** 12](#_Toc11147549)

[**3.1.** **Thiết kế giao diện** 12](#_Toc11147550)

[**3.1.1.** **Giao diện chính** 12](#_Toc11147551)

[**3.1.2.** **Thông tin và Cài đặt** 12](#_Toc11147552)

[**3.2.** **Kết nối Maple với C#** 14](#_Toc11147553)

[**3.3.** **Hiển thị ký tự toán học lên giao diện** 17](#_Toc11147554)

[**CHƯƠNG 4.** **THỬ NGHIỆM VÀ KẾT LUẬN** 21](#_Toc11147555)

[**4.1.** **Thử nghiệm** 21](#_Toc11147556)

[**4.1.1.** **Trường hợp sai cú pháp** 21](#_Toc11147557)

[**4.1.2.** **Đạo hàm** 22](#_Toc11147558)

[**4.1.3.** **Tích phân** 29](#_Toc11147559)

[**4.1.4.** **Giới hạn** 38](#_Toc11147560)

[**4.2.** **Kết luận** 41](#_Toc11147561)

[**TÀI LIỆU THAM KHẢO** 42](#_Toc11147562)

1. **TỔNG QUAN**
   1. **Phạm vi đề tài**

Đề tài hướng đến việc giải quyết các bài toán của nhánh Calculus trong Toán học. Các bài toán Calculus là các bài toán đi tìm lời giải cho 3 dạng toán chính sau: đạo hàm, tích phân và giới hạn. Đây đều là những dạng toán phổ biến trong chương trình đại số và giải tích của THPT. Cụ thể đối với đề tài, ta sẽ giải quyết các dạng toán sau:

* Tính đạo hàm của một hàm theo một biến hoặc nhiều biến cho trước
* Tính nguyên hàm, tích phân của một hàm
* Tính nguyên hàm, tích phân kép của một hàm
* Tính nguyên hàm, tích phân bội ba của một hàm
* Tính giới hạn của một hàm, bao gồm cả giới hạn trái và giới hạn phải
  1. **Mục tiêu, yêu cầu**

Việc xây dựng một chương trình giải toán Calculus hi vọng sẽ giúp ích cho các học sinh và giáo viên THPT trong quá trình học tập và giảng dạy Đại số - Giải tích cũng như các môn học khác như Hình học, Vật lý, Hóa học,… Đồng thời mong rằng các đối tượng khác có nhu cầu giải quyết các dạng toán này sẽ cảm thấy hữu ích và hài lòng khi sử dụng chương trình.

Như ta đã biết, đạo hàm, tích phân và giới hạn không phải là những dạng toán lạ, do đó đã có rất nhiều chương trình có khả năng giải quyết nhanh gọn các bài toán này, thậm chí được tích hợp vào các chương trình tính toán mặc định trên nhiều hệ điều hành của máy tính và điện thoại di động. Do đó, yêu cầu của chương trình không chỉ dừng lại ở việc tìm kiếm đáp án như ở các chương trình trên, mà phải đi sâu hơn bằng việc đưa ra lời giải chi tiết cho một bài toán, cũng như giải thích sơ bộ cho mỗi bước giải của chương trình. Các yêu cầu cụ thể của chương trình:

* Có giao diện đơn giản, trực quan, thân thiện, dễ sử dụng với tất cả người dùng, dễ dàng nhập dữ liệu vào chương trình theo những qui ước dễ hiểu và xuất ra lời giải quen thuộc với cách viết của người dùng (hiển thị chi tiết các ký tự toán học).
* Đưa ra lời giải chi tiết từng bước, có giải thích sơ bộ ở mỗi bước giải và dễ đọc, dễ hiểu với người dùng, nhất là với đối tượng học sinh, sinh viên.
* Giải quyết được hầu hết các dạng toán đạo hàm, tích phân, giới hạn với thời gian nhanh chóng hoặc có thể chấp nhận được.
  1. **Nền tảng công nghệ, công cụ**
* Nền tảng: Windows Form – Win32 Application
* Ngôn ngữ: Maple, C#
* Công cụ:
  + Maple 2016
  + Visual Studio (IDE)
  + Miktex (soạn thảo văn bản latex), với các package cần tải thêm: arabi, babel-vietnamese và vntex
  + ImageMagick (chuyển đổi văn bản sang dạng ảnh)
  + Ghostscript (hỗ trợ Miktex)

1. **PHƯƠNG PHÁP GIẢI BÀI TOÁN**
   1. **Giải toán Calculus trong Maple**

Trước khi đi vào từng bước giải chi tiết, ta cần quan tâm đến cách mà Maple thực hiện việc giải các bài toán Calculus như thế nào. Ta có cú pháp biểu thị cho đạo hàm, nguyên hàm, tích phân, giới hạn của Maple như sau:

Để tìm kết quả cho các biểu thức trên, ta chỉ việc thay các chữ hoa thành chữ thường trong tên hàm: Diff → diff, Int → int, Limit → limit.

VD: Diff (x^2, x) →

Int(x^2, x) = int(x^2, x) →

Như vậy, việc tìm đáp án cho một bài toán đạo hàm, tích phân, giới hạn là việc không hề khó trong Maple. Tuy nhiên, để hiển thị được lời giải chi tiết từng bước cho những bài toán trên lại là một vấn đề khác. Do đó, ta phải tìm kiếm một giải pháp khác được Maple hỗ trợ. Rất may mắn, trong Maple có một thư viện tên *Student[Calculus1]*, thư viện này sẽ hỗ trợ quá trình tính toán từng bước cho một bài toán Calculus. Có hai hàm trong thư viện mà ta cần quan tâm, đó là *Hint* và *Rule*.

Khi ta gọi hàm Hint cho một **biểu thức** expr (biểu thức này có dạng Diff(…), Int(…) hoặc Limit(…)), Maple sẽ tạo ra một bài toán (problem) và ghi nhận problem này vào bộ nhớ làm việc. Còn khi ta gọi hàm Hint cho một **đẳng thức** (là một bước giải của bài toán, có dạng: <problem> = <kết quả của bước i>), Maple sẽ tìm kiếm trong tất cả các bước giải của tất cả các problem đang giải để trả về kết quả. Kết quả mà hàm Hint trả về sẽ là một luật chung để giải quyết problem ở bước tại và tiến đến bước tiếp theo. Kế đến, khi ta muốn biết được kết quả của bài toán ở bước hiện tại, ta sử dụng hàm Rule. Chỉ số của hàm Rule là một luật – chính là kết quả lấy được từ hàm Hint, còn tham số của hàm Rule là một đẳng thức – chính là bước mà ta cần lấy kết quả. Kết quả mà hàm Rule trả về cũng là một đẳng thức và đây là đẳng thức của bước tiếp theo. Nếu ta nói tham số của Rule là (<problem> = <kết quả của bước i>), thì ta sẽ có kết quả trả về là (<problem> = <kết quả của bước i+1>). Khi ta dùng kết quả trả về này của hàm Rule làm tham số cho hàm Hint và thực hiện lại các bước trên cho đến khi không còn luật nào được sản sinh nữa thì ta sẽ có các bước giải chi tiết cho bài toán.

Ví dụ về Hint và Rule:

A:=Diff(x^2+x,x); →

Hint(A); → [sum]

B≔Rule[%](A); →

Hint(B); → [identity]

C:=Rule[%](B); →

* 1. **Xây dựng package cho chương trình**
     1. **Hàm giải chi tiết từng bước**

Như đã đề cập ở trên, quá trình giải chi tiết từng bước là một vòng lặp giữa hàm Hint và hàm Rule, ta có:

* **Input**: problem – bài toán cần giải
* **Output**: [steps, hints] – danh sách các bước giải và danh sách các hướng dẫn giải
* **Các bước thực hiện**: như sơ đồ

S

Đ

step ≔ problem

hint ≔ Hint (step)

hint = []

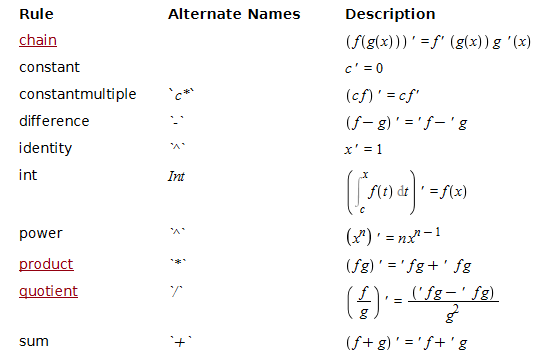
step ≔ Rule[hint] (step)

hints := [op(hints), [hint][1]]

steps := [op(steps), rhs(step)]

* + 1. **Hàm chuyển đổi lời giải thích (hint) sang tiếng Việt**

Lời giải thích của Maple được viết ngắn gọn bằng tiếng Anh, do đó ta cần đổi sang tiếng Việt và phải sử dụng các thuật ngữ quen thuộc trong Toán học để đảm bảo trải nghiệm cho người dùng. Trước tiên, ta cần xác định tất cả các luật (tên gọi của luật tương ứng như lời giải thích) được Maple sử dụng để giải các bài toán đạo hàm, tích phân, giới hạn. Các luật này được liệt kê đầy đủ trong Maple Help.



*Ảnh minh họa về bảng các luật dùng để giải đạo hàm trong Maple*

Sau khi thu thập và tổng hợp các luật giải đạo hàm, tích hợp và giới hạn xong, ta xây dựng bảng các luật như sau:

t ≔ table ([chain = "hàm hợp", change = "đổi biến", constant = "hằng số", constantmultiple = "nhân hằng số", dif = "đạo hàm", difference = "hiệu", dividebyzero = "chia cho 0", factor = "nhân tử", flip = "đảo cận", identity = "định nghĩa", int = "tích phân", join = "hợp lại", lhopital = "định lý L'Hospital", parts = "tích phân từng phần", partialfractions = "phân số đồng nhất", power = "lũy thừa", product = "nhân", quotient = "chia", revert = "trả lại", solve = "giải pt", split = "tách ra", sum = "tổng", rewrite = "viết lại"])

Tiếp theo, ta chỉ việc duyệt các hints và lần lượt thay thế sang từ tiếng Việt tương ứng dựa vào table t trên

* **Input**: hints – danh sách lời giải thích ở các bước
* **Output**: result – danh sách lời giải thích đã được xử lý
* **Các bước thực hiện**:
  + Bước 1: khai báo số lượng phần tử trong hints

n ≔ nops (hints);

* Bước 2: lần lượt duyệt và đưa phần tử tiếng Việt vào result

for i to n do

hint ≔ hints[i];

key ≔ hint[1];

value := t[key];

result ≔ [op(result), value];

end do;

* + 1. **Hàm xuất kết quả ra tập tin**

Ở đây, nhóm hiện thực hai hàm xuất kết quả ra tập tin, cả hai đều xuất vào file “input.tex” theo định dạng latex. Hàm đầu tiên có chức năng chỉ xuất đáp án ra file (dùng cho người dùng lựa chọn chế độ chỉ xem đáp án của bài toán). Do đó hàm này thiết kế khá đơn giản, chỉ việc chuyển đổi tham số đầu vào của hàm sang định dạng latex rồi xuất ra file.

* **Input**: result – đáp án của bài toán
* **Output**: đáp án đã được xuất ra tập tin
* **Các bước thực hiện**:
  + Bước 1: mở file

dir := FileTools:-JoinPath ([convert (currentdir(), string), "input.tex"]);

if not FileTools:-Exists (dir) then Export(dir, "") end if;

FileTools[Text]:-Open(dir, overwrite = true);

* + Bước 2: ghi kết quả ra file theo định dạng latex. Trong Maple có hỗ trợ hàm này. Tuy nhiên cần lưu ý, nhóm sử dụng option “**output = string**” làm tham số, đây là option chỉ có ở Maple 2016, do đó các phiên bản Maple cũ hơn sẽ không thể chạy được.

FileTools[Text]:-WriteString (dir, StringTools:-Join (["\\[", latex (result, output = string), "\\]"], ""));

* + Bước 3: đóng file

FileTools[Text]:-Close(dir);

Hàm thứ hai là hàm xuất ra lời giải đầy đủ cho người dùng, do đó đây là hàm xuất file mặc định được sử dụng trong chương trình. Hàm này cũng xuất ra một văn bản theo định dạng latex vào file “input.tex”.

* **Input**: problem – bài toán, solution – lời giải lấy được từ hàm giải chi tiết và showhints – biến xác định các hints có được xuất ra hay không
* **Output**: lời giải đã được xuất ra tập tin
* **Các bước thực hiện**:
  + Bước 1: mở file

dir := FileTools:-JoinPath([convert(currentdir(), string), "input.tex"]);

if not FileTools:-Exists(dir) then Export(dir, "") end if;

FileTools[Text]:-Open(dir, overwrite = true);

* Bước 2: ghi problem vào file

FileTools[Text]:-WriteString(dir, StringTools:-Join(["\\[", latex(problem, output = string), "\\]"], ""));

* Bước 3: khai báo steps là danh sách các bước giải, còn hints là danh sách các lời giải thích đã được chuyển sang tiếng Việt

steps := solution[1];

hints := DoiTiengViet(solution[2]);

* Bước 4: thực hiện quá trình lặp lần lượt ghi mỗi bước giải vào file.

n := nops(steps);

for i to n do

FileTools[Text]:-WriteString(dir, StringTools:-Join(["\\[=", latex(steps[i], output = string)], ""));

if showhints = true then

FileTools[Text]:-WriteString (dir, StringTools:-Join (["\\text{ (", hints[i], ")}"], "")) end if;

FileTools[Text]:-WriteString(dir, "\\]");

end do;

* Bước 5: đóng file

FileTools[Text]:-Close(dir);

* + 1. **Đóng gói package thành thư viện**

Đây là một bước khá quen thuộc khi làm việc trên Maple. Thư viện sau khi được đóng gói xong có đuôi “.mla” và sẽ được dùng để kết nối đến chương trình có giao diện để sử dụng. Chi tiết cách đóng gói:

packageDir := cat(currentdir(), kernelopts(dirsep), "DoAn.mla");

if FileTools:-Exists(packageDir) then FileTools:-Remove(packageDir) end if;

LibraryTools:-Create(packageDir); savelibname := packageDir;

savelib('cbrt', 'GiaiChiTiet', 'DoiTiengViet', 'XuatKetQua', 'XuatLoiGiai');

1. **THIẾT KẾ, CÀI ĐẶT CHƯƠNG TRÌNH**

Chương trình ứng dụng được thiết kế trên nền Windows Form, sử dụng ngôn ngữ lập trình C#. Với Maple làm phần nhân, C# chỉ đóng vai trò làm giao diện để nhập đề bài và xuất kết quả cho người dùng, tuy nhiên đây lại là khâu tác động lớn đến trải nghiệm của người dùng, yêu cầu ứng dụng viết ra phải đảm bảo tính dễ sử dụng, thân thiện và thẩm mỹ với mọi người dùng.

* 1. **Thiết kế giao diện**

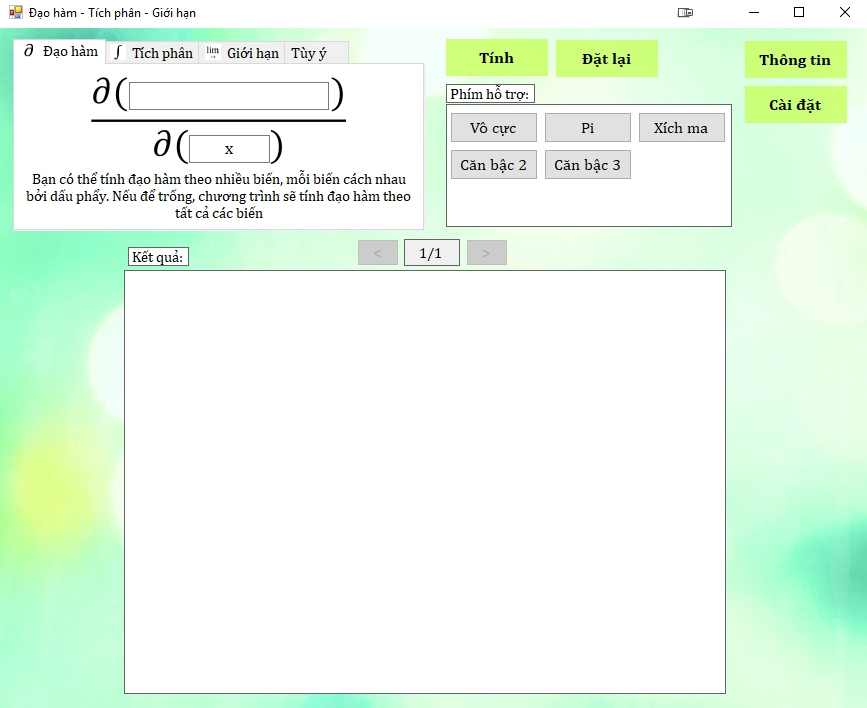
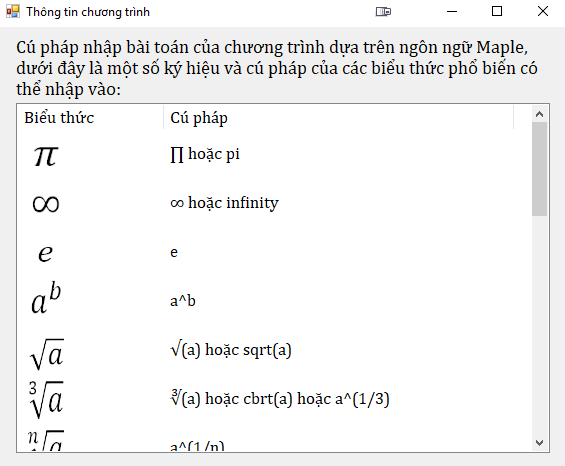
Chương trình được thiết kế với 3 giao diện: giao diện chính, giao diện cài đặt và giao diện thông tin; tương ứng với 3 form: MainForm, SettingsForm và InformationForm.

* + 1. **Giao diện chính**

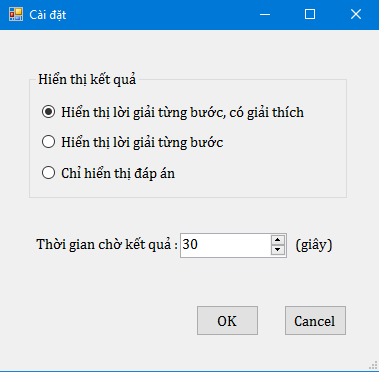
Dưới đây là giao diện chính của chương trình:

* + 1. **Thông tin và Cài đặt**

Giao diện thông tin là giao diện hiển thị một số thông tin về phần mềm, bao gồm tên chương trình, nhà phát triển, các yêu cầu cài đặt và quan trọng nhất là phần cú pháp của các biểu thức có thể nhập vào. Đối với những người dùng chưa quen với các cú pháp của Maple, họ có thể vào đây để xem và nhập đề bài.



Giao diện cài đặt gồm hai phần: cài đặt cách hiển thị kết quả và cài đặt thời gian chờ kết quả. Cài đặt cách hiển thị cho phép người dùng lựa chọn kết quả của bài toán sẽ được hiển thị lời giải đầy đủ có giải thích hay không có giải thích hoặc chỉ hiển thị đáp án. Mặc định thì chương trình sẽ hiển thị lời giải đầy đủ có giải thích. Cài đặt thời gian chờ kết quả là cài đặt mà khi chương trình tính toán vượt quá ngưỡng thời gian này, thông báo lỗi sẽ hiện ra (vì khi batch file gặp lỗi, chương trình sẽ không thể thoát khỏi tiến trình của file batch để chạy tiếp). Mặc định thì thời gian chờ là 30 giây, đối với những bài toán rất khó ta nên tăng thời gian này lên để đảm bảo xem được kết quả.



* 1. **Kết nối Maple với C#**

Phương pháp được nhóm sử dụng để kết nối Maple với C# là thông qua batch file. Sau khi có được package đã được đóng gói (“DoAn.mla”), ta tiến hành đưa package này vào thư mục chứa chương trình và tạo thêm một file batch cũng tại thư mục này có tên là “solve.bat” để gọi chương trình commandline của Maple. Nội dung của file batch:

"C:\Program Files\Maple 2016\bin.X86\_64\_WINDOWS\cmaple.exe" input.mpl

Trong đó, “input.mpl” là file chứa nội dung đề bài được nhập, cấu trúc của nó như sau:

packageDir := cat (currentdir(), kernelopts(dirsep) , "DoAn.mla"):

march('open', packageDir):

< Đề bài cần giải >

Trong đó, hai dòng đầu là để kết nối đến package mà ta đã bỏ vào trong thư mục của chương trình. Các dòng tiếp theo là đề bài mà ta cần giải. Do đề bài là không thể xác định trước nên “input.mpl” phải được tạo thông qua code C#.

Tổng hợp lại, ta có các xử lý dưới đây trong Visual Studio để thực hiện kết nối và giải một bài toán:

* **Input**: input – đề bài ở dạng Diff(…), Int(…), Limit(…), option – lựa chọn hiển thị lời giải của người dùng (tương ứng trong phần Cài đặt)
* **Output**: text – lời giải được xuất ra theo định dạng latex
* **Các bước thực hiện**:
  + Bước 1: tạo và mờ file “input.mpl”

writer = File.CreateText("input.mpl");

* Bước 2: ghi nội dung bao gồm cả đề bài vào file

writer.WriteLine("packageDir:= cat(currentdir(), kernelopts(dirsep) , \"DoAn.mla\"):");

writer.WriteLine("march('open', packageDir):");

switch (option)

{

case 0: // hiển thị lời giải đầy đủ có giải thích

writer.WriteLine("A := " + input + ";");

writer.WriteLine("S := GiaiChiTiet(A);");

writer.WriteLine("XuatLoiGiai(A, S, true);");

break;

case 1: // hiển thị lời giải đầy đủ

writer.WriteLine("A := " + input + ";");

writer.WriteLine("S := GiaiChiTiet(A);");

writer.WriteLine("XuatLoiGiai(A, S,false);");

break;

case 2: // chỉ hiển thị kết quả

string result = input.Replace("Diff", "diff")

.Replace("Int", "int")

.Replace("Limit", "limit");

writer.WriteLine("S := " + input + " = " + result + ";");

writer.WriteLine("XuatKetQua(S);");

break;

}

* Bước 3: đóng file

writer.Close();

* Bước 4: chạy file “solve.bat”

ProcessStartInfo processInfo = new ProcessStartInfo();

processInfo.FileName = "solve.bat";

processInfo.UseShellExecute = false;

processInfo.CreateNoWindow = true;

Process process = Process.Start(processInfo);

process.WaitForExit();

* Bước 5: mở file “input.tex” để lấy kết quả trả về của bài toán

reader = File.OpenText("input.tex");

string text = reader.ReadToEnd();

reader.Close();

return text;

* 1. **Hiển thị ký tự toán học lên giao diện**

Để hiển thị các ký tự toán học lên giao diện của chương trình, nhóm sử dụng các công cụ: Miktex (với các package tải thêm là arabi, babel-vietnamese và vntex), ImageMagick và Ghostscript. Phương pháp chung của các công cụ này là soạn thảo một văn bản latex, build thành file “.pdf”, sau đó chuyển từ văn bản pdf này sang dạng ảnh “.png”.

Đầu tiên, ta tạo một file batch có tên “display.bat”. Nội dung của file:

rem display.bat

rem %1 represents the file name with no extension

pdflatex -jobname=output %1

convert output.pdf output.png

File batch này có chức năng build một văn bản latex rồi chuyển đổi sang dạng hình ảnh của nó. Khi thực thi file này cần phải nhập một tham số vào, đó là tên file latex đầu vào mà ta muốn chuyển đổi. Trong trường hợp của chúng ta, đó là file “input.tex”, nên ta sẽ nhập vào là “input”. Sau khi batch file chạy xong, ta sẽ nhận được một file “output.pdf” và một hoặc nhiều file ảnh có đuôi “.png”. Cụ thể, nếu số trang của file pdf là 1 thì ta có một file “output.png”, còn nếu file pdf có nhiều trang (n trang) thì ta có các file “output-0.png”, “output-1.png”, “output-2.png”, … “output-n.png”.

Tổng hợp lại, ta thực hiện giải thuật sau để hiển thị kết quả lên giao diện chương trình:

* **Input**: text – lời giải nhận được ở [bước 5 phần 3.2](#text), beginDoc – đoạn mở đầu cho văn bản latex, endDoc – đoạn kết thúc cho văn bản latex, document – văn bản latex hoàn chỉnh.
* beginDoc = @"

\documentclass[12pt,a4paper]{report}

\usepackage{amssymb}

\usepackage{amsmath}

\usepackage[utf8]{inputenc}

\usepackage[vietnamese]{babel}

\usepackage[landscape,margin=0.5cm]{geometry}

\begin{document}

\pagestyle{empty}

\fussy

{\Huge";

* endDoc = @"}\end{document}";
* document = beginDoc;
* **Output**: hình ảnh được xuất lên giao diện
* **Các bước thực hiện**:
* Bước 1: ghi document hoàn chỉnh vào “input.tex”

document += text;

document += endDoc;

File.WriteAllText("input.tex", document);

* Bước 2: chạy file “display.bat”

Process process = new Process();

process.StartInfo.FileName = "display.bat";

process.StartInfo.Arguments = "input";

process.StartInfo.UseShellExecute = false;

process.StartInfo.CreateNoWindow = true;

process.Start();

process.WaitForExit(waitingTime);

* Bước 3: tìm các file có cấu trúc “output\*.png”

string pattern = "output\*.png";

string[] dirs = GetFiles (GetCurrentDirectory(), pattern);

* Bước 4: nếu không có file nào thì báo lỗi, nếu có một file thì hiển thị file “output.png”, nếu có nhiều file thì hiển thị file đầu tiên là “output-0.png” (đồng thời ta lưu lại chỉ số trang bằng biến imageIndex để thực hiện việc chuyển trang thông qua hai nút chuyển trang).

if (dirs.Length == 0)

{

resultPictureBox.Image = Properties.Resources.error;

imageIndex = -1;

imageCount = 0;

}

else if (dirs.Length == 1)

{

resultPictureBox.ImageLocation = "output.png";

imageIndex = 0;

imageCount = 1;

}

else

{

resultPictureBox.ImageLocation = "output-0.png";

imageIndex = 0;

imageCount = dirs.Length;

}

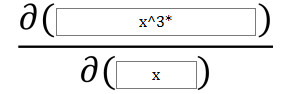
* Bước 5: quá trình hiển thị hoàn tất, gán lại document chuẩn bị cho lần hiển thị sau

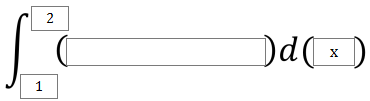
document = beginDoc;

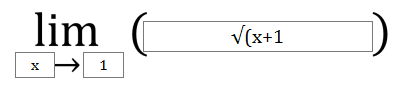
}

1. **THỬ NGHIỆM VÀ KẾT LUẬN**
   1. **Thử nghiệm**
      1. **Trường hợp sai cú pháp**

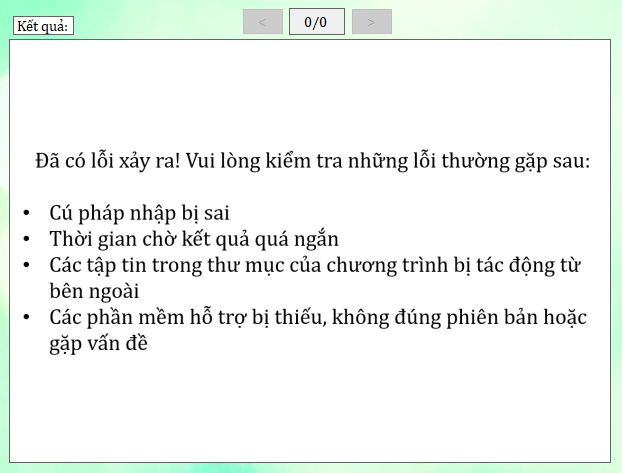
Đầu tiên, ta thử nghiệm một số trường hợp sai cú pháp như sau:







Kết quả, sau 30 giây (thời gian chờ kết quả mặc định) chương trình sẽ hiển thị thông báo lỗi như sau:



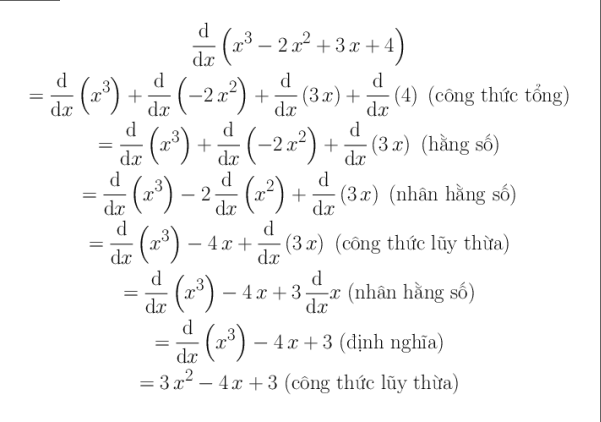
* + 1. **Đạo hàm**

Các bài toán thử nghiệm được lấy từ trang web: <http://toanhoc247.com/dao-ham-va-cac-bai-toan-lien-quan-day-du-chi-tiet-a5357.html>

* Lời giải được cung cấp:



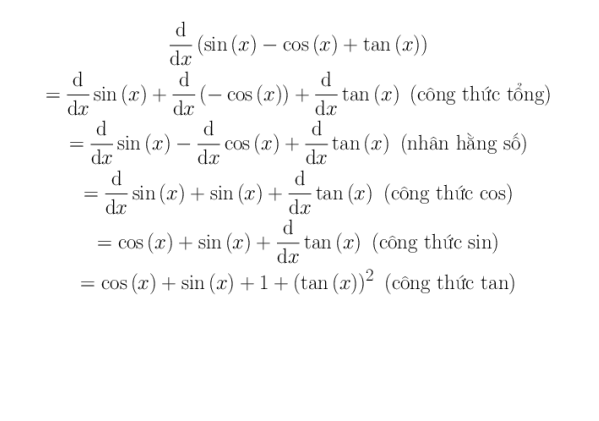
* Lời giải của chương trình



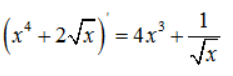
* Lời giải được cung cấp:



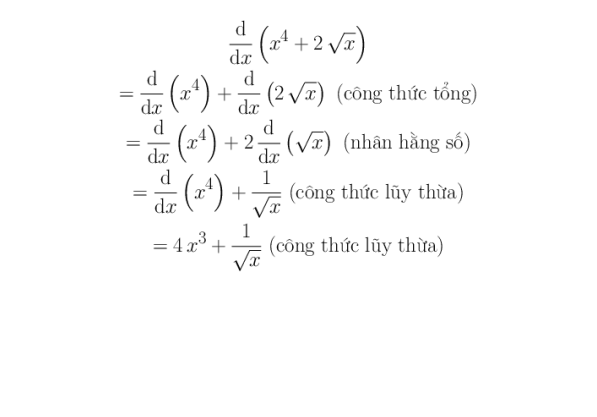
* Lời giải của chương trình



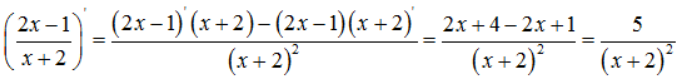
* Lời giải được cung cấp:



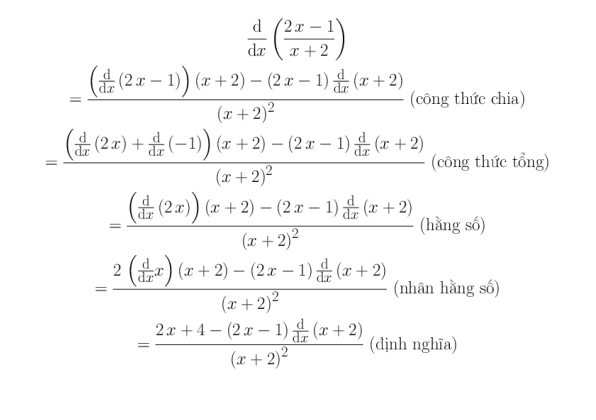
* Lời giải của chương trình

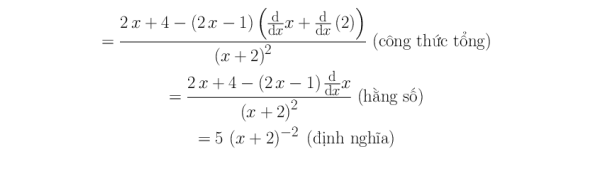


* Lời giải được cung cấp:

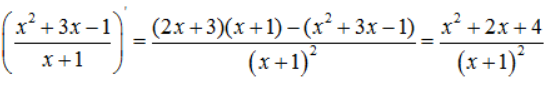


* Lời giải của chương trình

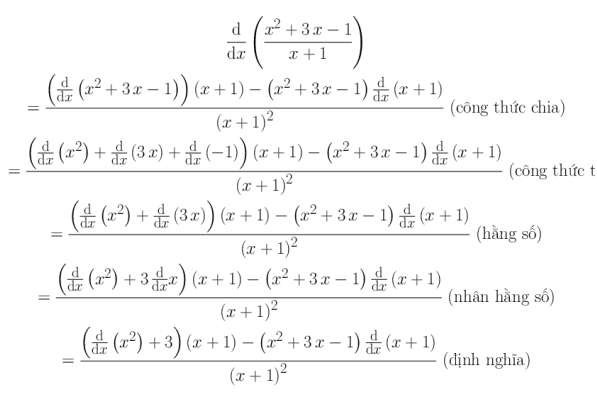


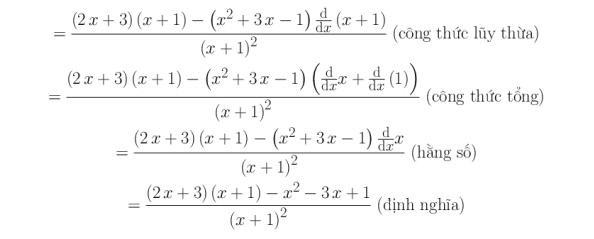


* Lời giải được cung cấp:



* Lời giải của chương trình

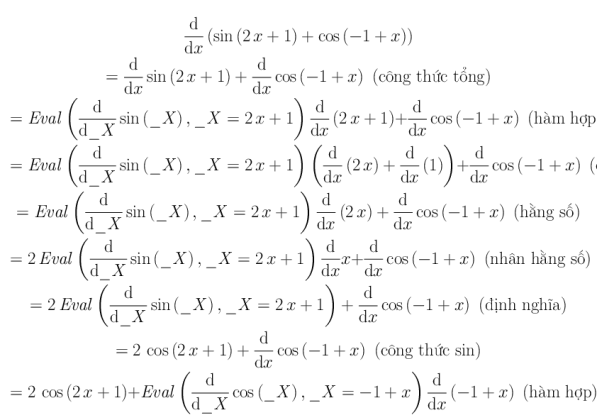


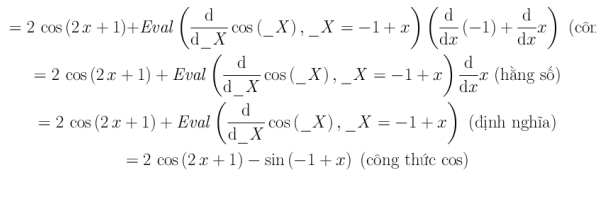


* Lời giải được cung cấp:

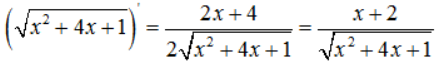


* Lời giải của chương trình

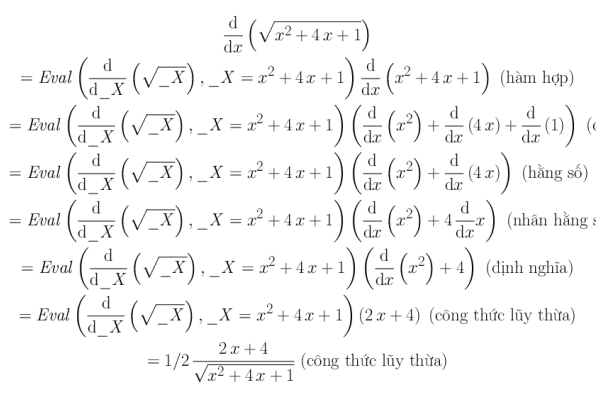




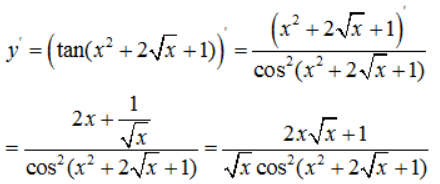
* Lời giải được cung cấp:



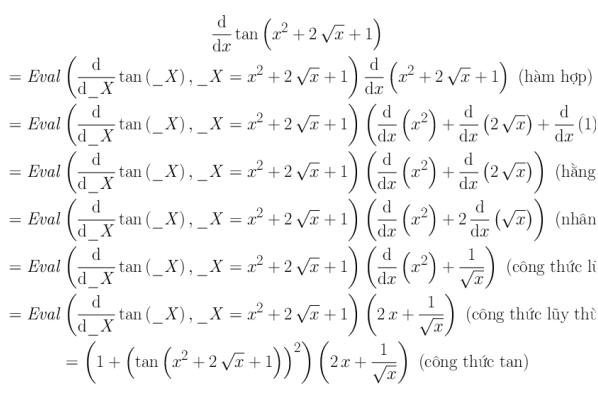
* Lời giải của chương trình



* Lời giải được cung cấp:



* Lời giải của chương trình

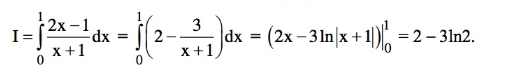


* + 1. **Tích phân**

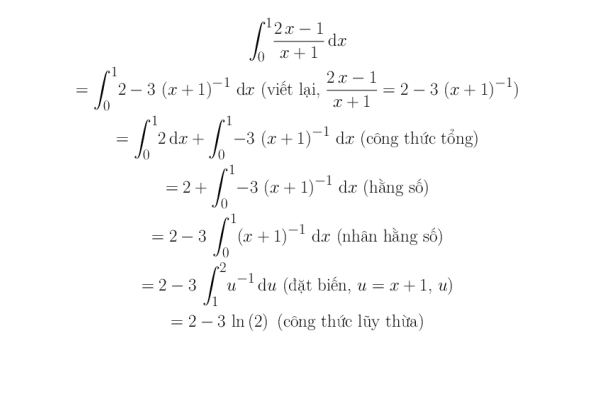
Các ví dụ thử nghiệm được lấy từ trang web <http://kenhtuyensinh.vn/phuong-phap-giai-bai-tap-tich-phan-va-de-thi-thu-chu-de-tich-phan> (bao gồm đề thi đại học một số năm) cộng với một số bài tự tạo.

1. dx

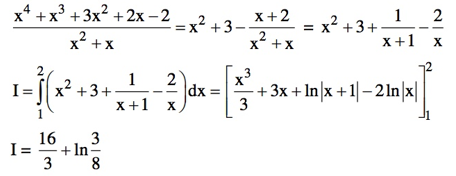
* Lời giải được cung cấp:



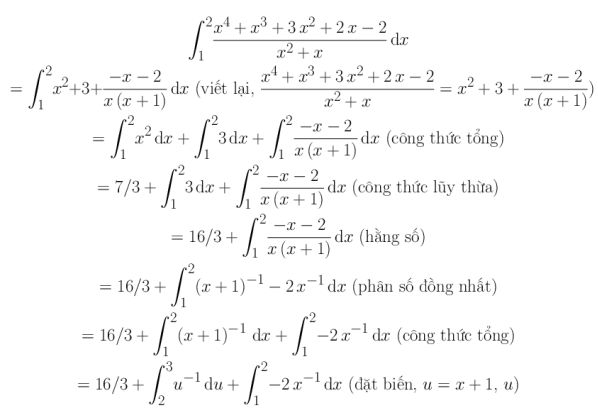
* Lời giải của chương trình:

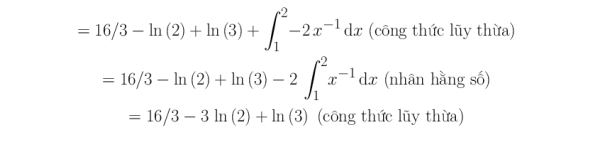


* Lời giải được cung cấp:

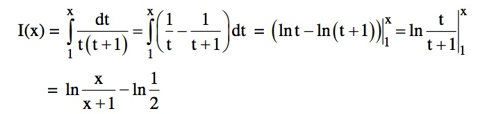


* Lời giải của chương trình:

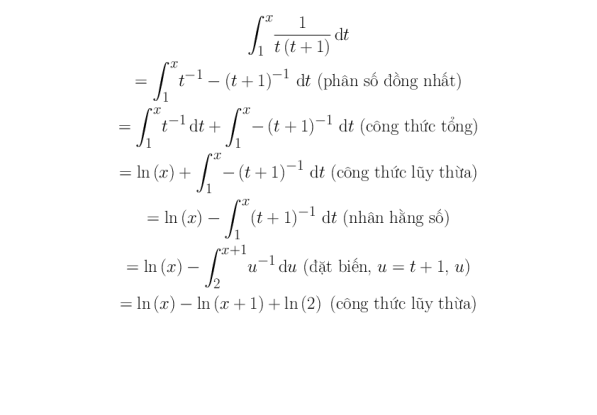




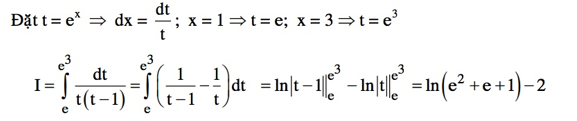
* Lời giải được cung cấp:



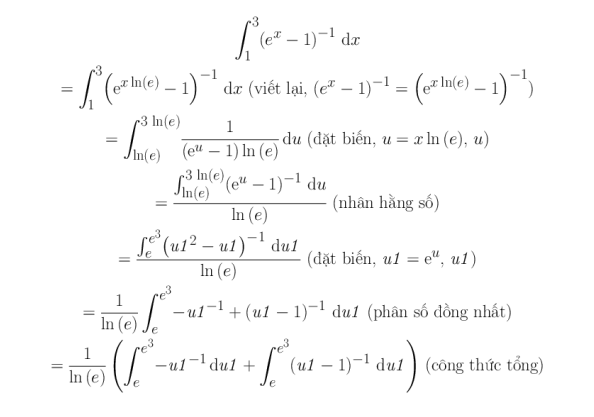
* Lời giải của chương trình:

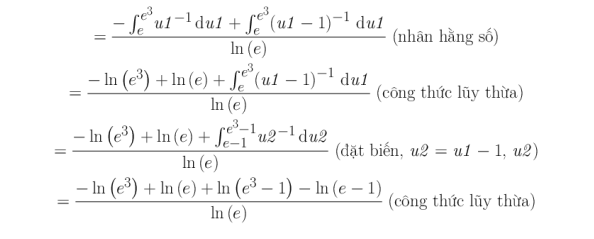


* Lời giải được cung cấp:

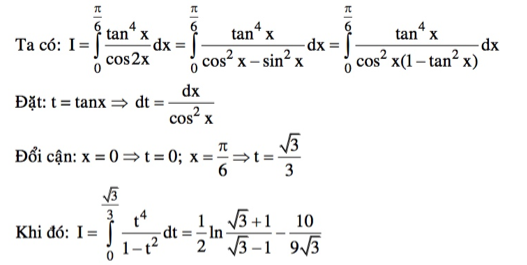


* Lời giải của chương trình:

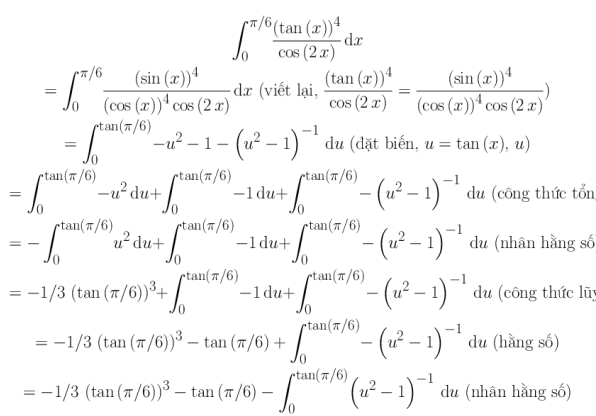


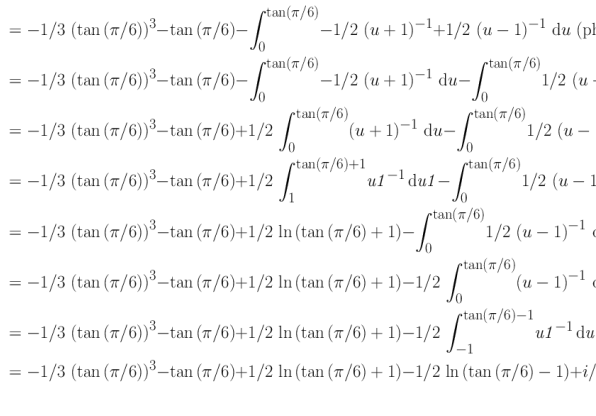


* Lời giải được cung cấp:

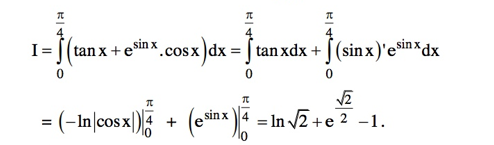


* Lời giải của chương trình:

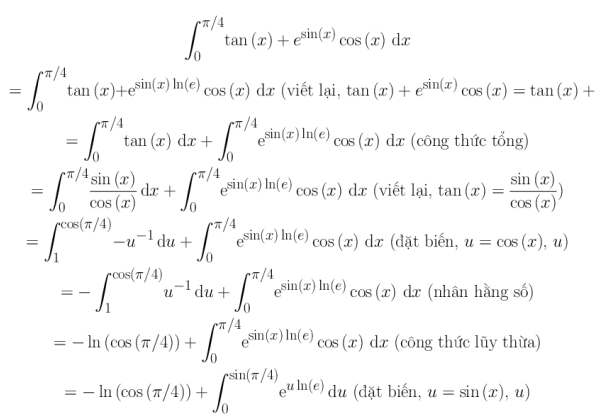


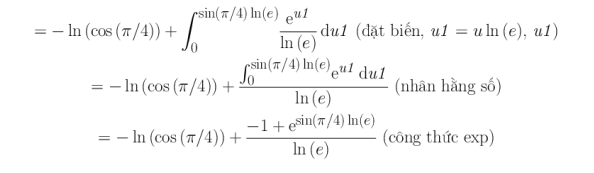


* Lời giải được cung cấp:

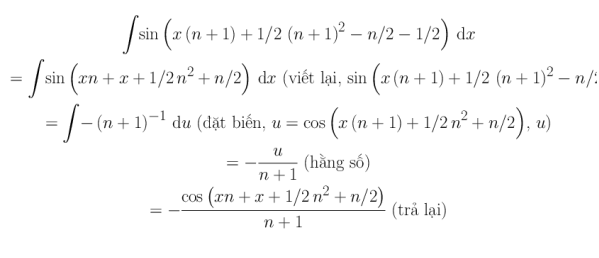


* Lời giải của chương trình:

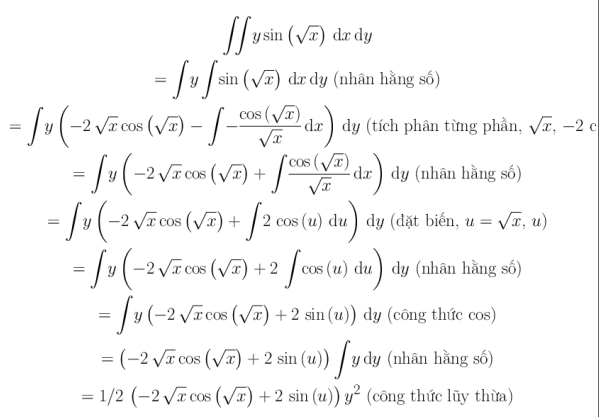




* Lời giải của chương trình:

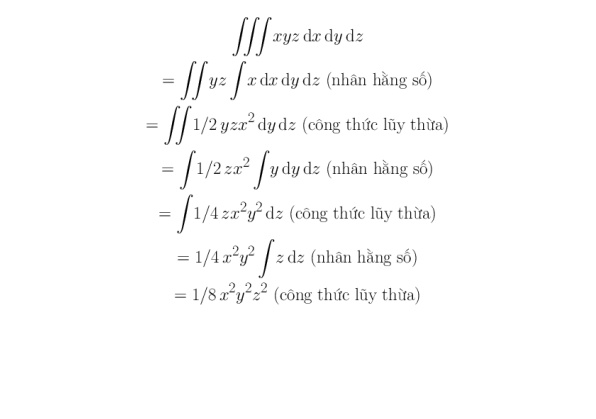


* Lời giải của chương trình:





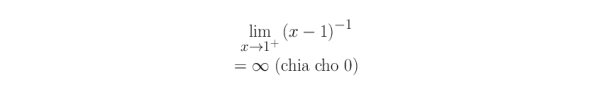
* Lời giải của chương trình:



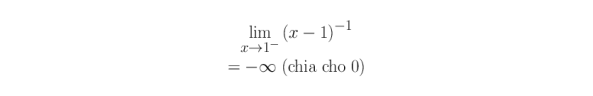
* + 1. **Giới hạn**

Các bài toán được lấy từ lấy từ trang web <http://tailieuhoctap.vn/chi-tiet-sach/394-nganh-cong-nghe-thong-tin/the-loai-khac/769430-bai-tap-kem-loi-giai-gioi-han-ham-so> và một số bài được nhóm tự tạo.

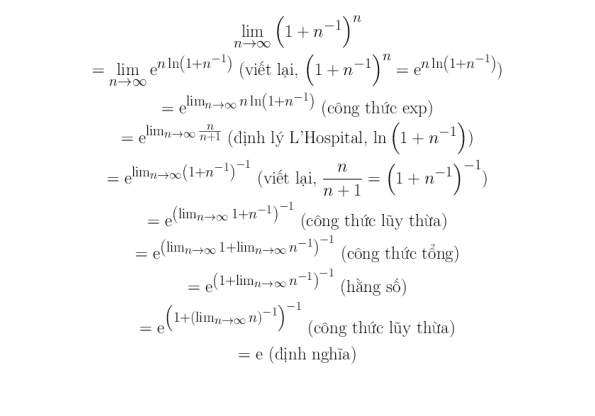
* Lời giải của chương trình:



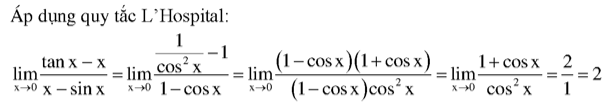
* Lời giải của chương trình:



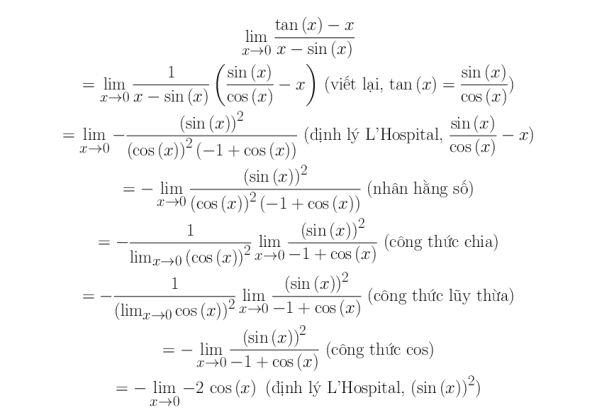
* Lời giải của chương trình:

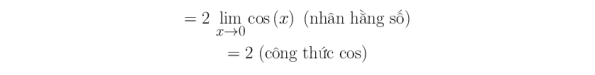


* Lời giải được cung cấp

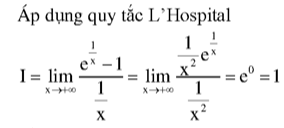


* Lời giải của chương trình:

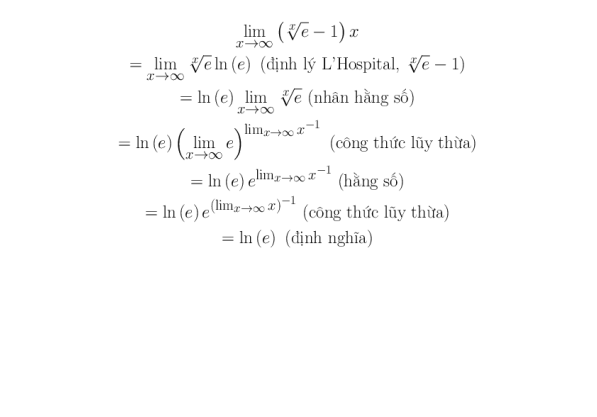




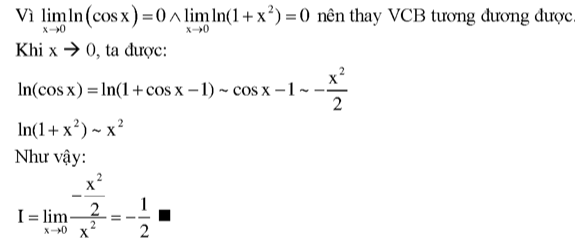
* Lời giải được cung cấp:



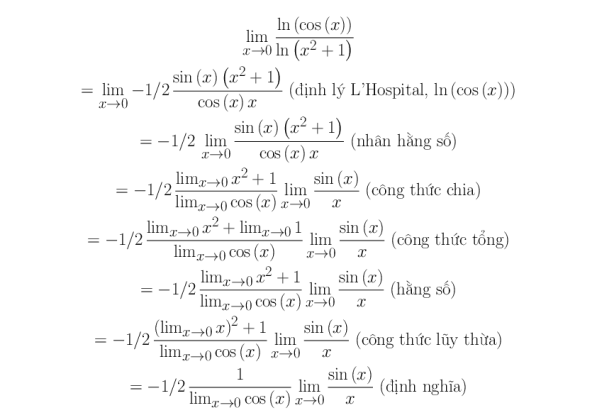
* Lời giải của chương trình:



* Lời giải được cung cấp:



* Lời giải của chương trình:





* 1. **Kết luận**

Sau khi thực hiện quá trình thử nghiệm với khá nhiều bài toán đưa ra, nhóm rút ra nhận xét là chương trình đã thực hiện đưa ra lời giải tốt và chính xác trong nhiều trường hợp. Thời gian thực thi nhanh với những bài toán cơ bản và chấp nhận được với những bài toán khó. Tuy nhiên, vẫn có trường hợp chương trình đưa ra lời giải không được tự nhiên, cách giải dài dòng không phù hợp với đối tượng học sinh sinh viên. Ngoài ra, cách trình bày lời giải cũng có chút khác biệt so với cách trình bày truyền thống của học sinh sinh viên Việt Nam nên sẽ gây bỡ ngỡ cho người dùng lúc mới sử dụng. Có trường hợp cá biệt, chương trình còn đưa ra đáp án bị sai (ví dụ 5 phần Giới hạn). Hy vọng trong tương lai, Maple sẽ cải thiện những khuyết điểm này trong hệ thống của mình, để giúp cho chương trình của nhóm và nhiều chương trình khác có sử dụng nhân Maple trở nên tốt và hoàn thiện hơn nữa.

# **TÀI LIỆU THAM KHẢO**

[1] Maple Help

[2] Đạo hàm và các bài toán liên quan: <http://toanhoc247.com/dao-ham-va-cac-bai-toan-lien-quan-day-du-chi-tiet-a5357.html>

[3] Phương pháp giải bài tập Tích Phân và đề thi thử chủ đề Tích Phân: <http://kenhtuyensinh.vn/phuong-phap-giai-bai-tap-tich-phan-va-de-thi-thu-chu-de-tich-phan>

[4] Bài Tập kèm lời giải - Giới Hạn Hàm Số: <http://tailieuhoctap.vn/chi-tiet-sach/394-nganh-cong-nghe-thong-tin/the-loai-khac/769430-bai-tap-kem-loi-giai-gioi-han-ham-so>