

ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN TIN



BÀI TẬP LỚN
Kỹ thuật lập trình

Giáo viên hướng dẫn: Nguyễn Thị Thanh Huyền

Sinh viên thực hiện: Trần Ngọc Bảo

20227086

Hà Nội, ngày 24 tháng 6 năm 2024

Lời mở đầu

Thông thường muốn tính tích phân xác định của một hàm số trên đoạn $[a, b]$, ta tìm nguyên hàm của hàm đó và thay cận từ a đến b . Tuy nhiên trong khoa học và kỹ thuật không phải lúc nào ta cũng tìm được nguyên hàm của một hàm cho trước, tức là đôi khi tồn tại nguyên hàm của một hàm nhưng chúng ta không thể biểu diễn nguyên hàm ấy dưới dạng các hàm sơ cấp. Chính vì vậy mà sẽ có một số giải pháp để tính tích phân những hàm như thế. Thêm vào đó với sự phát triển của công nghệ thông tin, thì sự hỗ trợ của máy tính là rất tuyệt vời, tuy nhiên để máy tính có thể hiểu được thì chúng ta phải có một thuật toán. Từ đó chúng ta sẽ thiết lập các phương pháp tính xấp xỉ mà nếu với sự hỗ trợ của máy tính ta sẽ có những kết quả xấp xỉ hoàn toàn chấp nhận được.

Trong báo cáo bài tập lớn này sẽ trình bày về việc thiết kế một chương trình cơ bản để tính tích phân của hàm một biến $y = f(x)$ trên một đoạn $[a, b]$ và các phương pháp số được dùng là phương pháp dựa trên công thức hình thang và công thức Simpson.

I. Xây dựng module chương trình

Nếu muốn máy tính thực hiện một ý tưởng của con người, trước tiên ta phải xác định rõ thực tế con người chúng ta cần làm gì, đôi khi những việc chúng ta lơ đi tưởng chừng là đơn giản nhưng để khiến một cỗ máy vật lý có thể làm là một điều khó khăn.

Ở bài toán của chúng ta, ta cần xây dựng một chuỗi các bước để giúp một chương trình làm theo các bước đó. Theo hướng tư duy ngược ta có thể xác định được ta cần các bước nhập liệu và tính toán. Tuy nhiên để một chương trình có tính mở rộng và bảo trì cao hơn, việc thiết kế một menu lựa chọn và thuật toán đi kèm việc thao tác menu là không thể thiếu.

1.1. Menu

Trong phạm vi một báo cáo môn học, việc tạo một menu cầu kì là không cần thiết nên trong báo cáo này, menu và các thao tác trên menu sẽ thực hiện ở trên Console, thứ đã quá quen thuộc với các lập trình viên. Như đã trình bày ở trên, chúng ta hướng tới một chương trình có tính mở rộng và bảo trì cao và không nên “lờ đi” các việc tưởng chừng đơn giản. Các việc đơn giản ở đây chính là “*Nhập chuỗi từ bàn phím*” và “*Nhập số thực từ bàn phím*”, “*In ra menu chính*”(bản chất của việc này là in ra màn hình các tên lựa chọn).

Ngoài menu chính ra chúng ta có và có thể sẽ có thêm các menu khác nữa nếu chúng ta muốn, ở trong báo cáo này, menu phụ dùng để chọn phương pháp hình thang hoặc Simpson. Và các việc không đơn giản lắm sẽ là “*Thao tác với các lựa chọn*” việc thao tác với các lựa chọn bản chất là gọi tới các hàm thực hiện các “lựa chọn” đó.

1.2. Tính giá trị $f(x)$ tại một điểm cho trước

Sau khi đã tạo được menu, chúng ta sẽ tới phần chính của chương trình của chúng ta - Tính tích phân hay chính xác hơn là tính gần đúng tích phân. Trước khi tính tích phân ta phải xác định được công thức của phương pháp ta cần thực hiện.

+ Đối với công thức hình thang

$$\int_a^b f(x) dx \approx \frac{h}{2} \left[f(x_0) + f(x_n) + 2 \sum_{i=1}^{n-1} f(x_i) \right]$$

với $h = \frac{b-a}{n}$

+ Đối với công thức Simpson

$$\int_a^b f(x) dx \approx \frac{h}{3} \left[f(x_0) + f(x_n) + 4 \sum_{i=1, \text{lẻ}}^{n-1} f(x_i) + 2 \sum_{i=1, \text{chẵn}}^{n-1} f(x_i) \right]$$

với $h = \frac{b-a}{n}$

Như đã thấy ở công thức, chúng ta lại cần làm một việc “đơn giản” nữa, đó chính là tính $f(x)$ tại một điểm trên đoạn $[a, b]$. Rõ ràng việc tính toán với chúng ta là quá đơn giản nhưng với máy tính là việc khác. Ta cần phải cho máy tính biết cách tính như thế nào. Đôi khi để ra cùng một kết quả thì các thao tác trong thực tế con người làm và thao tác trong máy tính là khác nhau. Việc tính $f(x)$ tại một điểm ở trong báo cáo này là một ví dụ cho sự khác nhau đó. Thay vì thực hiện đơn thuần từ trái phải, chúng ta sẽ thực hiện theo thứ tự các toán tử giữa các biểu thức. Như vậy, các công việc của chúng ta khi muốn tính $f(x)$ tại một điểm sẽ được chia thành các việc nhỏ như sau:

- + Phân tích và tính toán các biểu thức có các toán tử cộng (+) và trừ (-).
- + Phân tích và tính toán các biểu thức có các toán tử nhân (*) và chia (/).
- + Phân tích và tính toán các biểu thức có toán tử lũy thừa (^).
- + Phân tích và tính toán các đơn vị cơ bản trong biểu thức như số, biến và hàm.
- + Tính toán giá trị của các hàm cơ bản như hàm lượng giác, logarit,...
- + Tính toán giá trị của biến ẩn

Lưu ý: Vì việc kiểm tra tính đúng đắn của biểu thức $f(x)$ của một biểu thức trong máy tính là một điều khó hơn với thực tế ta làm bằng tay, nên trong báo cáo này ta giả định $f(x)$ luôn đúng đắn và cũng đảm bảo xác định, liên tục trên $[a, b]$.

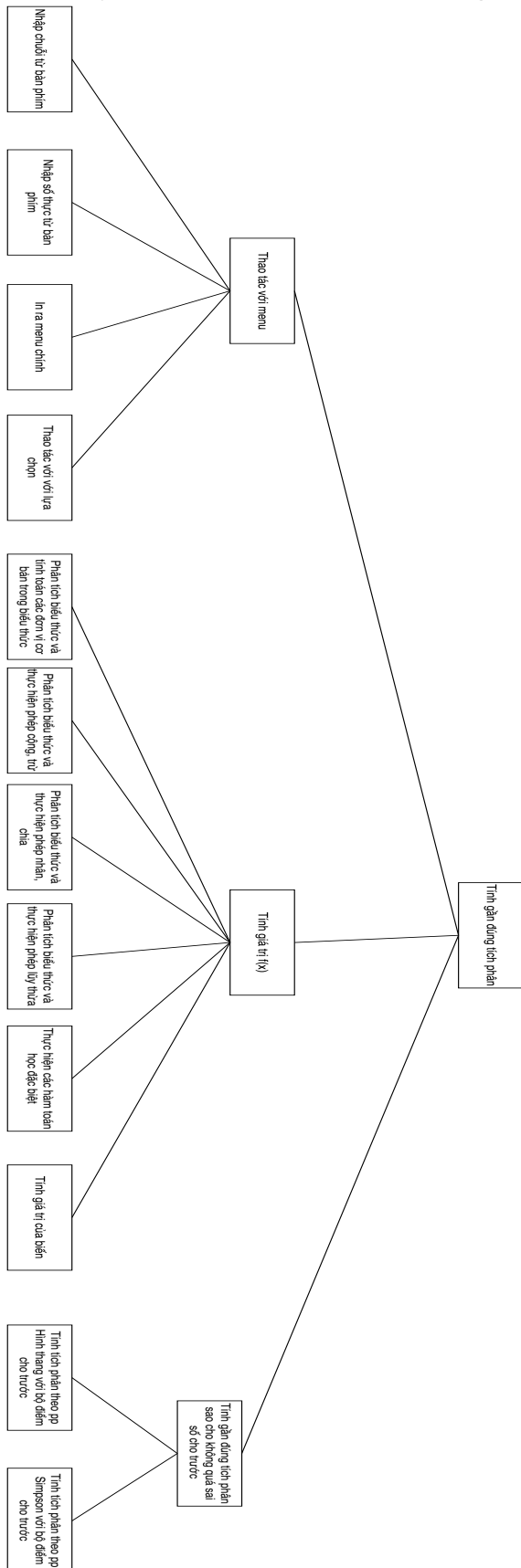
1.3. Tính tích phân

Ta có thể nhận thấy rằng sau khi tính được giá trị của $f(x)$ tại một điểm bất kì trên $[a, b]$, ta có thể tính được tích phân dựa theo công thức (1) hoặc (2) kể trên. Vấn đề còn lại chỉ là áp dụng công thức, tuy nhiên nếu chúng ta muốn có kết quả tính tốt hơn ta sẽ thực hiện đi thực hiện lại công thức (1) hoặc (2) với sau mỗi lần thực hiện ta sẽ tăng n lên 2 lần đến khi nào sai số giữa các kết quả tính không quá một hằng số epsilon nào đó. Ý tưởng của việc này là ta lặp đi lặp lại cách tính với n tăng 2 lần sau mỗi lần tính, ta sẽ được kết quả tốt hơn kết quả đã tính trước đó (một cách lý giải khác là sai số của phương pháp này sẽ giảm nếu số bước h giảm hay số n tăng lên).

Như vậy, việc tính tích phân của chúng ta nhìn chung sẽ được chia thành 2 công việc sau:

- + Tính tích phân theo bộ điểm (x, y) cho trước

+ Tính tích phân sao cho giá trị tích phân có sai số không vượt quá epsilon
Sau đây là sơ đồ module của chúng ta



1.4. Lưu ý khi đọc báo cáo

Các biến sau là biến toàn cục:

- + function: chuỗi biểu diễn $f(x)$
- + x: số thực x
- + a, b, e: 2 đầu mút a, b và sai số epsilon
- + precision: số chữ số sau dấu phẩy mong muốn
- + res_tichphan: kết quả tính tích phân
- + flag2: biến cờ cho biết việc đã tính toán giá trị $f(x)$ tại một điểm hay chưa
- + flag3: biến cờ cho biết việc đã tính toán tích phân hay chưa và tính bằng phương pháp nào. Ở bài tập của chúng ta, flag3=1 tức là tính bằng phương pháp hình thang, flag=2 tức là tính bằng phương pháp Simpson
- + biến var: biến biểu diễn cấu trúc biến ẩn x trong hàm $f(x)$, cấu trúc gồm tên biến và địa chỉ của biến

II. Thiết kế menu

Như đã nói ở phần trước, menu trong chương trình của chúng ta được “làm” trên console. Ở trong báo cáo này, với mục tiêu chương trình có thể thực hiện trên đa số hệ thống, các thao tác trên console sẽ sử dụng thư viện ncurses.

Đôi chút về ncurses:

Thư viện lập trình ncurses (new curses) cung cấp một giao diện lập trình ứng dụng (API) cho phép lập trình viên viết các giao diện người dùng dựa trên văn bản (TUI) một cách không phụ thuộc vào terminal. Đây là một bộ công cụ để phát triển phần mềm ứng dụng "giống GUI" chạy dưới một trình giả lập terminal. Nó cũng tối ưu hóa các thay đổi trên màn hình, nhằm giảm độ trễ khi sử dụng các shell từ xa.

Nhìn chung, các thao tác khi ta sử dụng thư viện trên chính là:

1. Khởi tạo ncurses: `initscr()`. Bước này khởi tạo thư viện ncurses, thiết lập các giá trị mặc định và phân bổ bộ nhớ cho các cấu trúc dữ liệu cần thiết. Nếu không thực hiện bước này, các hàm ncurses khác sẽ không hoạt động đúng.
2. Điều chỉnh chế độ nhập: `noecho()`, `cbreak()`. `noecho()` tắt chế độ echo, nghĩa là ký tự nhập vào không được in ra màn hình. `cbreak()` tắt buffering dòng, nghĩa là tất cả các ký tự nhập vào được truyền đi ngay lập tức, không chờ đến khi có dấu xuống dòng. Cả hai bước này giúp kiểm soát cách nhập liệu từ người dùng.

3. Xóa màn hình: `clear()`. Bước này xóa màn hình, chuẩn bị cho việc in ra nội dung mới.
4. In ra nội dung: `printw()`, `mvprintw()`, v.v. Các hàm này in ra nội dung lên màn hình. Bạn có thể điều chỉnh vị trí in bằng cách sử dụng các hàm như `mvprintw()`.
5. Cập nhật màn hình: `refresh()`. Bước này cập nhật màn hình, hiển thị tất cả các thay đổi từ lần cập nhật cuối cùng. Nếu không thực hiện bước này, các thay đổi sẽ không được hiển thị.
6. Kết thúc ncurses: `endwin()`. Bước này giải phóng bộ nhớ đã phân bổ cho ncurses và trả lại điều khiển cho terminal. Nếu không thực hiện bước này, terminal có thể không hoạt động đúng sau khi chương trình kết thúc.

Và đó là đôi nét về thư viện chúng ta sẽ sử dụng trong báo cáo này để tạo menu, tiếp theo chúng ta sẽ làm rõ các module trong việc thiết kế menu

2.1. In ra menu

Ở trong báo cáo này, menu chính sẽ được triển khai ở dạng “2 chiều”. Cột đầu tiên lần lượt sẽ biểu hiện cho các chức năng như sau “Nhập $f(x)$ từ bàn phím”, “Tính $f(x)$ tại một điểm”, “Tính gần đúng tích phân”, “Làm sạch”, “Thoát chương trình”. Cột thứ 2 chỉ có “Nhập hàm $f(x)$ từ một file txt”.

Ý nghĩa của các chức năng khá rõ ràng. Nếu muốn tính được tích phân $f(x)$ là gì ta cần phải cho chương trình biết được $f(x)$ là gì trước đã, vậy nên ta cần phải nhập $f(x)$, nếu $f(x)$ là hàm phức tạp việc nhập sẵn ở file là cần thiết. Nếu muốn kiểm tra nghi ngờ gì ở việc tính tích phân có đúng hay $f(x)$ có xác định trên đoạn $[a, b]$ hay không, thì ta cần tính $f(x)$ tại điểm mà ta nghi ngờ. Nếu chúng ta muốn tiếp tục việc thao tác tính toán, việc làm sạch là cần thiết, làm sạch ở đây có nghĩa như một công việc reset lại chương trình. Và một chương trình cũng phải có kết thúc - một điều hiển nhiên. Cuối cùng, việc tính tích phân là phần chính nên chắc chắn phải có nó.

Để có tính linh hoạt cho menu, việc in ra menu sau khi thao tác điều hướng là thật sự cần thiết nên chúng ta sẽ triển khai hàm `print_menu` như sau:

Đầu vào: 2 số nguyên, mảng hai chiều menu

Ý tưởng: In ra menu và highlight thao tác đã điều hướng tới

1. Tính số lượng lựa chọn là kích thước của mảng menu chia cho kích thước của một phần tử trong mảng
2. Khởi tạo ncurses
3. Xóa màn hình
4. Tắt echo
5. Tắt buffering dòng, truyền tất cả

6. Duyệt qua tất cả các lựa chọn trong menu:

6.1. Duyệt qua tất cả các cột trong mỗi lựa chọn:

Nếu hàng và cột hiện tại là hàng và cột được highlight:

1. Bật chế độ đảo ngược màu sắc
2. In lựa chọn hiện tại với độ rộng 30 ký tự
3. Tắt chế độ đảo ngược màu sắc

Ngược lại:

In lựa chọn hiện tại với độ rộng 30 ký tự

6.2. In một dòng mới

7. Cập nhật màn hình

Lưu ý:

- Việc tắt echo ở đây có ý nghĩa như đã giải thích khi ta đã nói qua về ncurses. Nó ngăn cho ký tự nhập không được in ra màn hình. Có thể chúng ta sẽ nghĩ là chúng ta không gọi tới hàm nào yêu cầu nhập từ bàn phím thì sao lại có thao tác này???. Lý giải cho việc này là console không hoạt động theo chúng ta nghĩ. Việc chúng ta nhận một phím bất kỳ cũng hiển thị ra các ký tự tương ứng, kể cả các phím “không biểu diễn cho một chữ cái hay số”.

2.2. Nhập một chuỗi từ bàn phím

Như đã nói trước đó, chúng ta đang làm việc với ncurses và đang thiết kế chương trình định hướng cho người sử dụng biết làm gì sau các thao tác chọn. Việc nhắc nhở đang “làm gì” trước khi thực sự “làm gì” là cần thiết, sau đây chính là hàm nhập từ bàn phím(`get_input_string`)

Đầu vào: chuỗi nhắc (prompt), con trỏ chuỗi buffer, số nguyên biểu diễn cỡ tối đa của buffer(`buffer_size`)

Ý tưởng: In ra prompt yêu cầu người dùng nhập chuỗi buffer, bật tắt echo trước và sau khi thực hiện nhập chuỗi

1. In ra chuỗi prompt
2. Bật chế độ echo
3. Nhận chuỗi đầu vào từ người dùng, chuỗi này sẽ được lưu vào buffer, (`buffer_size - 1`) sẽ là số ký tự tối đa sẽ nhận(điều này đảm bảo dành chỗ cho ký tự NULL kết thúc chuỗi)
4. Tắt echo

2.3. Nhập một số thực từ bàn phím

Tương tự với `get_input_string`, ta có hàm `get_input_float`.

Đầu ra: Chuỗi nhắc prompt

Đầu ra: Giá trị đã nhập

Ý tưởng: Để tiện cho việc tương thích với thư viện `ncurses`, thay vì gọi trực tiếp các hàm nhập `scanf` hay `fscanf` ta sẽ dùng cách khác là chuyển chuỗi đã nhập từ bàn phím thành số (hiển nhiên chuỗi này chính là số), các thao tác như in ra prompt và bật tắt echo tương tự như `get_input_string`

1. Khởi tạo một mảng ký tự buffer có kích cỡ 50 (con số có thể thay đổi nếu muốn)
2. Bật echo
3. Nhận chuỗi đầu vào từ người dùng, lưu vào buffer, (50-1) là ký tự tối đa sẽ nhận
4. Tắt echo
5. Chuyển đổi chuỗi trong buffer thành số thực, lưu vào một biến tên là `value` được khai báo trước đó
6. Trả về `value`

Lý giải kĩ hơn cho việc “đi đường vòng” phải chuyển chuỗi thành số là:

Thư viện `ncurses` trong C được thiết kế cho màn hình ký tự và không cung cấp một hàm trực tiếp để nhập dữ liệu kiểu float hoặc bất kỳ kiểu dữ liệu nào không phải là chuỗi.

Nếu ta muốn đọc một số float trực tiếp, ta thường sẽ sử dụng một hàm như `scanf` trong C. Tuy nhiên, `scanf` không hoạt động tốt với `ncurses` vì `ncurses` yêu cầu kiểm soát đầu vào và đầu ra của terminal để hoạt động đúng, trong khi `scanf` lại bỏ qua sự kiểm soát này.

Vì vậy, mặc dù có thể về mặt kỹ thuật sử dụng `scanf` hoặc các hàm tương tự khi `ncurses` đang hoạt động, nhưng điều này thường không được khuyến nghị vì có thể dẫn đến hành vi không mong muốn. Thực hành tốt nhất khi sử dụng `ncurses` là đọc đầu vào dưới dạng chuỗi bằng cách sử dụng các hàm của `ncurses` như `getnstr`, và sau đó chuyển đổi sang kiểu dữ liệu mong muốn khi cần.

2.4. Thao tác với các lựa chọn

Vì chương trình của chúng ta mong muốn thiết kế cho việc thuận tiện để bảo trì và mở rộng, nên ở đây chúng ta thay vì tạo một switch case đơn giản để cho chương trình chính thực hiện các thao tác gọi tới, chúng ta sẽ làm một cách trông linh hoạt hơn một chút. Đó là tạo một mảng con trỏ hàm trỏ tới các hàm thực hiện ứng với các lựa chọn. Việc này khá hữu ích khi chúng ta muốn sửa các hàm thực hiện hành động và nếu muốn dùng tới chỉ cần truy vấn tới một phần tử của mảng hành động đó.

Vì việc thực hiện các thao tác lựa chọn kia không phải là một hàm nên chúng ta không có các bước xác định đầu ra vào hay thuật toán như các phần trước.

2.5. Các hàm thực hiện ứng với các lựa chọn

Lưu ý:

- Các việc tạm dừng chương trình ở trong mục này có mục đích làm cho chương trình “động” hơn.

2.5.1. Nhập $f(x)$ từ bàn phím

Function `handle_option_1`

1. Gọi tới hàm `get_input_string` với đầu vào là chuỗi biểu diễn $f(x)$ và kích cỡ tối đa mà ta định trước của chuỗi $f(x)$
2. Tạm dừng thực thi trong khoảng 1s

2.5.2. Tính giá trị của $f(x)$ tại một điểm

Function `handle_option_2`

1. Loại bỏ ký tự xuống dòng (nếu có) khỏi cuối chuỗi function bằng cách tìm vị trí của ký tự xuống dòng và thay thế nó bằng ký tự kết thúc chuỗi (`\0`).
2. Kiểm tra xem chuỗi function có rỗng không (tức là không chỉ chứa ký tự kết thúc chuỗi). Nếu function không rỗng:

2.1. Yêu cầu người dùng nhập vào một số thực, được lưu vào biến `x`, thông qua hàm `get_input_float` với thông điệp "Nhập điểm cần tính: ".

2.2. Sử dụng biến `expression` để lưu trữ địa chỉ của chuỗi function.

2.3. Tính toán kết quả của biểu thức lưu trong function tại điểm `x` bằng hàm `parse_expression`, và lưu kết quả vào biến `result`.

2.4. In kết quả ra màn hình, hiển thị biểu thức và giá trị của nó tại điểm `x`.

2.5. Gọi hàm `refresh` để cập nhật màn hình với thông tin mới in ra.

2.5. Tạm dừng thực thi chương trình trong 1 giây bằng hàm.

2.6. Đặt giá trị của biến toàn cục `flag2` thành 1, có thể để theo dõi rằng một phép tính đã được thực hiện thành công.

3. Nếu chuỗi function rỗng (tức là chưa có biểu thức nào được nhập vào):

3.1. In ra màn hình thông điệp "Ban chưa nhập ham số" để thông báo cho người dùng biết rằng họ cần nhập một biểu thức trước khi thực hiện tính toán.

3.2. Gọi hàm refresh để cập nhật màn hình với thông điệp mới.

3.3. Tạm dừng thực thi chương trình trong 1 giây bằng hàm

2.5.3. Tính tích phân với sai số cho trước

Function `handle_option_3`

1. Kiểm tra xem chuỗi function (được giả định là biểu thức hàm số) có rỗng không. Nếu không rỗng, thực hiện các bước tiếp theo; nếu rỗng, hiển thị thông điệp "Ban chưa nhập ham số" và tạm dừng chương trình trong 1 giây.
2. Yêu cầu người dùng nhập vào các giá trị a, b (giới hạn dưới và giới hạn trên của khoảng tích phân), e (sai số cho phép), và precision (số chữ số thập phân độ chính xác) thông qua hàm `get_input_float`.
3. Khởi tạo biến expression để lưu trữ địa chỉ của chuỗi function.
4. Hiển thị submenu cho người dùng và cho phép họ chọn giữa các tùy chọn bằng cách sử dụng phím mũi tên lên và xuống. Người dùng xác nhận lựa chọn của mình bằng cách nhấn phím Enter (`\n`).
5. Dựa trên lựa chọn của người dùng (`sub_choice`), thiết lập giá trị cho biến `flag3` (1 hoặc 2) và gọi hàm `integrate` với các tham số tương ứng để tính toán kết quả tích phân. Kết quả được lưu vào biến `result` và `res_tichphan`.
6. Thoát khỏi vòng lặp khi người dùng nhấn Enter, tức là đã chọn xong tùy chọn và tính toán đã được thực hiện.
7. Hiển thị kết quả tích phân của hàm số trên khoảng $[a, b]$.
8. Cập nhật màn hình với thông tin mới và tạm dừng chương trình trong 1 giây

Lưu ý, việc in ra submenu tương tự như in ra menu chính và sẽ đề cập ở mục 2.6

2.5.4. Thoát chương trình

Function `exit_program`

1. Kết thúc chế độ ncurses (điều này tương đương với việc trả lại terminal cho hệ điều hành)
2. Gọi tới hàm kết thúc chương trình

2.5.5. Reset chương trình

Function reset

1. Gọi tới hàm xóa màn hình
2. Đặt lại function thành chuỗi rỗng
3. Đặt lại biến flag2, flag3 bằng 0
4. Gọi hàm cập nhật màn hình

2.5.6. Nhập f(x) từ một file

Function handle_option_6

1. Khai báo một mảng ký tự filename với kích thước 100 để lưu tên đầy đủ của file.
2. Gọi hàm get_input_string với thông điệp "Nhap ten file: ", mảng filename và kích thước của mảng này làm tham số.
3. Mở file với tên được lưu trong filename ở chế độ đọc.
4. Kiểm tra xem file có mở thành công không bằng cách kiểm tra giá trị của file. Nếu file là NULL, tức là không thể mở file, thực hiện các bước sau:
 - 4.1. Hiện thị thông điệp "Khong the mo file\n".
 - 4.2. Cập nhật màn hình.
 - 4.3. Tạm dừng chương trình trong 1 giây.
 - 4.4. Kết thúc hàm sớm bằng từ khóa return.
5. Nếu file mở thành công, đọc dữ liệu từ file vào biến function sử dụng hàm fgets. Hàm này đọc một dòng từ file và lưu vào function, với kích thước tối đa được xác định bởi sizeof(function).
6. Đóng file.

2.6. In ra menu con cho việc lựa chọn cách tính tích phân

Để đơn giản cho việc thiết lập chúng ta sẽ lựa chọn cách tạo menu này theo một chiều đơn giản

Đầu vào: số tự nhiên highlight

1. Tạo biến cục bộ menu là mảng chứa hai xâu "Phuong phap hinh thang", "Phuong phap Simpson"
2. Khởi tạo biến num_choices bằng số lựa chọn (ở đây là 2)
3. Khởi tạo môi trường ncurses
4. Xóa màn hình
5. Tắt echo
6. Tắt buffer dòng
7. Duyệt qua các phần tử của menu với chỉ số chạy i: 0->num_choices-1
 - 7.1. Nếu highlight = i+1:
 - 7.1.1. Bật đảo ngược màu

7.1.2. In ra phần tử thứ menu[i]

7.1.3. Tắt đảo ngược màu

7.2. ngược lại:

7.2.1 In ra menu[i]

8. Cập nhật màn hình

III. Tính giá trị $f(x)$ tại một điểm

Lưu ý:

- Tư tưởng của việc thực hiện tính giá trị $f(x)$ tại một điểm của chúng ta trong báo cáo này là việc tính lần lượt các biểu thức theo thứ tự ưu tiên: phép lấy mũ, phép nhân chia, phép cộng trừ. Tức là điều đầu tiên chúng ta sẽ gọi tới hàm tính cộng trừ, và trong hàm tính cộng trừ ấy chúng ta sẽ tính nhân chia, và cứ thế... Như chúng ta thấy, bản chất đây là đệ quy.
- Các hàm của chúng ta sẽ và ý tưởng của các hàm sẽ là:
 - + `parse_expression`: Hàm này phân tích cú pháp một biểu thức toán học, xử lý các phép cộng (+) và trừ (-). Nó đệ quy gọi `parse_term` để xử lý các phần của biểu thức và kết hợp kết quả.
 - + `parse_term`: Hàm này phân tích cú pháp các "term" trong biểu thức, xử lý các phép nhân (*) và chia (/). Nó đệ quy gọi `parse_factor` để xử lý các yếu tố cấu thành term và kết hợp kết quả.
 - + `parse_factor`: Hàm này phân tích cú pháp các "factor" (yếu tố), xử lý phép lũy thừa (^). Nó đệ quy gọi `parse_primary` để xử lý các yếu tố cơ bản và áp dụng phép lũy thừa nếu cần.
 - + `parse_primary`: Hàm này phân tích cú pháp các yếu tố cơ bản của biểu thức, bao gồm số, biến, hàm, và biểu thức con trong dấu ngoặc. Nó có thể gọi `parse_expression` để xử lý biểu thức con, `evaluate_function` để tính toán giá trị của một hàm, hoặc `evaluate_variable` để lấy giá trị của một biến.
 - + `evaluate_function`: Hàm này tính toán giá trị của một hàm toán học (ví dụ: sin, cos, log, v.v.) dựa trên tên và đối số của hàm đó.
 - + `evaluate_variable`: Hàm này tìm và trả về giá trị của một biến dựa trên tên của biến đó. Nó tìm kiếm trong một mảng các biến được truyền vào hàm.
- Vẫn với ý định là chương trình sẽ có tính mở rộng, thay vì chúng ta chỉ làm một hàm tính toán với một biến x, chúng ta sẽ thiết kế các hàm tính toán

với nhiều biến. Tuy nhiên với quy mô một báo cáo như thế này, ta sẽ không làm hẳn một chương trình tính tích phân của hàm nhiều biến.

3.1. Phân tích biểu thức và tính cộng trừ

Function `parse_expression`

Đầu vào: con trỏ hằng trỏ tới con trỏ trỏ tới chuỗi biểu thức $f(x)$, biến x đã thay số, số nguyên biểu diễn số biến của hàm $f(x)$

Một lẽ thông thường khi tính toán $f(x)$ tại một điểm ta chỉ cần hàm $f(x)$ và giá trị của x tại một điểm cần tính, với lý do vừa kể trên, ta sẽ cần tới một biến là số các biến trong $f(x)$.

Đầu ra: kết quả của $f(x)$ tại điểm x

Như đã bàn luận ở lưu ý trên, bản chất của chúng ta là đệ quy và đầu ra của việc này chính là điều cần tìm nên không bất ngờ gì về việc một hàm tính cộng trừ nhưng lại có ngay kết quả cuối cùng.

Ý tưởng: Nó phân tích cú pháp một biểu thức toán học, xử lý các phép cộng (+) và trừ (-). Nó đệ quy gọi `parse_term` (hàm tính toán phép nhân chia) để xử lý các phần của biểu thức và kết hợp kết quả.

1. Khởi tạo giá trị kết quả: Đầu tiên, hàm tính giá trị của một "term" (đơn vị cơ bản của biểu thức, có thể là một số, biến, hoặc một biểu thức con) bằng cách gọi hàm `parse_term` với các tham số là biểu thức, mảng các biến và số lượng biến. Giá trị này được lưu trữ trong biến `result`.
2. Lặp qua biểu thức: Hàm tiếp tục kiểm tra từng ký tự trong biểu thức. Nếu gặp ký tự '+' hoặc '-', hàm sẽ thực hiện các bước tiếp theo. Nếu không, vòng lặp kết thúc và hàm trả về giá trị `result`.
3. Xác định phép toán: Khi gặp ký tự '+' hoặc '-', hàm lưu trữ ký tự đó vào biến `op` và sau đó dịch chuyển con trỏ biểu diễn $f(x)$ để bỏ qua ký tự phép toán.
4. Tính giá trị term tiếp theo: Hàm gọi lại hàm `parse_term` để tính giá trị của term tiếp theo trong biểu thức.
5. Thực hiện phép toán: Dựa vào giá trị của `op` (phép toán là '+' hoặc '-'), hàm cộng hoặc trừ giá trị của term mới với giá trị `result` hiện tại.
6. Trả về kết quả: Sau khi đã xử lý hết các phép toán cộng và trừ trong biểu thức, hàm trả về giá trị cuối cùng của `result`.

3.2. Phân tích và tính toán các phép nhân chia

Function `parse_term`

Đầu vào và đầu ra tương tự với `parse_expression`

Ý tưởng: Phân tích cú pháp các "term" trong biểu thức, xử lý các phép nhân (*) và chia (/). Độ quy gọi `parse_factor` để xử lý các yếu tố cấu thành term và kết hợp kết quả.

1. Khởi tạo giá trị kết quả: Hàm bắt đầu bằng cách tính giá trị của một "factor" (yếu tố cơ bản của term, có thể là một số, biến, hàm, hoặc một biểu thức con) bằng cách gọi hàm `parse_factor` với các tham số là biểu thức, mảng các biến và số lượng biến. Giá trị này được lưu trữ trong biến `result`.
2. Lặp qua biểu thức: Hàm tiếp tục kiểm tra từng ký tự trong biểu thức. Nếu gặp ký tự '*' (nhân) hoặc '/' (chia), hàm sẽ thực hiện các bước tiếp theo. Nếu không, vòng lặp kết thúc và hàm trả về giá trị `result`.
3. Xác định phép toán: Khi gặp ký tự '*' hoặc '/', hàm lưu trữ ký tự đó vào biến `op` và sau đó dịch chuyển con trỏ tới chuỗi `f(x)` để bỏ qua ký tự phép toán.
4. Tính giá trị factor tiếp theo: Hàm gọi lại `parse_factor` để tính giá trị của factor tiếp theo trong biểu thức.
5. Thực hiện phép toán: Dựa vào giá trị của `op` (phép toán là '*' hoặc '/'), hàm nhân hoặc chia giá trị của factor mới với giá trị `result` hiện tại.
6. Trả về kết quả: Sau khi đã xử lý hết các phép toán nhân và chia trong biểu thức, hàm trả về giá trị cuối cùng của `result`.

3.3. Phân tích và tính toán các phép lũy thừa

Đầu vào và đầu ra tương tự với `parse_expression`

Ý tưởng: Phân tích cú pháp các "factor" (yếu tố), xử lý phép lũy thừa (^). Độ quy gọi `parse_primary` để xử lý các yếu tố cơ bản và áp dụng phép lũy thừa nếu cần.

1. Khởi tạo giá trị kết quả: Hàm bắt đầu bằng cách tính giá trị của một "primary" (yếu tố cơ bản nhất của factor, có thể là một số, biến, hàm, hoặc một biểu thức con trong dấu ngoặc) bằng cách gọi hàm `parse_primary` với các tham số là biểu thức, mảng các biến và số lượng biến. Giá trị này được lưu trữ trong biến `result`.
2. Lặp qua biểu thức: Hàm tiếp tục kiểm tra từng ký tự trong biểu thức. Nếu gặp ký tự '^' (phép lũy thừa), hàm sẽ thực hiện các bước tiếp theo. Nếu không, vòng lặp kết thúc và hàm trả về giá trị `result`.
3. Xử lý phép lũy thừa: Khi gặp ký tự '^', hàm dịch chuyển con trỏ tới chuỗi `f(x)` để bỏ qua ký tự phép lũy thừa.
4. Tính giá trị của số mũ: Hàm gọi lại `parse_primary` để tính giá trị của số mũ trong phép lũy thừa.
5. Thực hiện phép lũy thừa: Tính lũy thừa của `result` với số mũ vừa tính được, sau đó cập nhật lại giá trị của `result` với kết quả mới.

6. Trả về kết quả: Sau khi đã xử lý hết các phép lũy thừa trong biểu thức, hàm trả về giá trị cuối cùng của result.

3.4. Phân tích biểu thức và tính toán các đơn vị cơ bản trong biểu thức

Function `parse_primary`

Đầu vào, đầu ra tương tự hàm `parse_expression`

Ý tưởng: Phân tích cú pháp các yếu tố cơ bản của biểu thức, bao gồm số, biến, hàm, và biểu thức con trong dấu ngoặc. Có thể gọi `parse_expression` để xử lý biểu thức con, `evaluate_function` để tính toán giá trị của một hàm, hoặc `evaluate_variable` để lấy giá trị của một biến.

1. Xử lý biểu thức trong dấu ngoặc đơn: Nếu ký tự đầu tiên của biểu thức là một dấu ngoặc mở '(', hàm sẽ tăng con trỏ biểu thức để bỏ qua dấu ngoặc này và gọi `parse_expression` để tính giá trị của biểu thức bên trong dấu ngoặc. Sau khi biểu thức bên trong được xử lý, nếu gặp dấu ngoặc đóng ')', con trỏ biểu thức sẽ được tăng lên để bỏ qua dấu ngoặc đóng.
2. Xử lý biến hoặc hàm: Nếu ký tự đầu tiên là một chữ cái, hàm sẽ xác định đây là tên của một biến hoặc một hàm. Hàm tiếp tục đọc các ký tự tiếp theo cho đến khi gặp một ký tự không phải là chữ cái hoặc số, sau đó lưu tên này vào một biến tạm thời.

2.1. Nếu tên này theo sau là dấu ngoặc mở '(': Điều này chỉ ra rằng đây là một hàm. Hàm sẽ tăng con trỏ biểu thức để bỏ qua dấu ngoặc mở, sau đó gọi `parse_expression` để tính giá trị của đối số truyền vào hàm. Nếu gặp dấu ngoặc đóng ')', con trỏ biểu thức sẽ được tăng lên để bỏ qua nó. Cuối cùng, hàm `evaluate_function` được gọi để tính giá trị của hàm với đối số đã cho.

2.2. Nếu không gặp dấu ngoặc mở sau tên: Điều này chỉ ra rằng đây là một biến. Hàm `evaluate_variable` được gọi để lấy giá trị của biến.

3. Xử lý số: Nếu ký tự đầu tiên không phải là dấu ngoặc mở và không phải là chữ cái, hàm giả định rằng đây là một số. Hàm `strtod` được sử dụng để chuyển đổi chuỗi ký tự thành một số thực và cập nhật con trỏ biểu thức để bỏ qua phần đã được chuyển đổi.
4. Trả về kết quả: Hàm trả về giá trị đã được tính toán, dựa trên loại phân tử cơ bản được xử lý: biểu thức trong dấu ngoặc, biến, hàm, hoặc số.

3.5. Tính toán giá trị của các hàm cơ bản như hàm lượng giác, logarit,..

Function `evaluate_function`

Đầu vào: con trỏ hằng trỏ tới tên của chuỗi tên các hàm(name), số thực arg

Đầu ra: Trả về giá trị của hàm muốn tính với tham số arg

1. Kiểm tra tên hàm: Hàm bắt đầu bằng việc so sánh tên hàm được truyền vào (qua tham số name) với các tên hàm toán học đã biết như "sin", "cos", "tan", "log", "exp", và "sqrt"
2. Tính toán và trả về kết quả:
 - 2.1. Nếu tên hàm khớp với "sin", hàm trả về giá trị của hàm sin của đối số arg sử dụng hàm sin.
 - 2.2. Nếu tên hàm khớp với "cos", hàm trả về giá trị của hàm cos của đối số arg sử dụng hàm cos.
 - 2.3. Nếu tên hàm khớp với "tan", hàm trả về giá trị của hàm tan của đối số arg sử dụng hàm tan.
 - 2.4. Nếu tên hàm khớp với "log", hàm trả về giá trị của hàm logarit tự nhiên của đối số arg sử dụng hàm log.
 - 2.5. Nếu tên hàm khớp với "exp", hàm trả về giá trị của hàm lũy thừa e mũ arg sử dụng hàm exp.
 - 2.6. Nếu tên hàm khớp với "sqrt", hàm trả về giá trị của hàm căn bậc hai của đối số arg sử dụng hàm sqrt.
3. Trả về 0.0 cho các hàm không xác định: Nếu tên hàm không khớp với bất kỳ tên hàm toán học nào đã biết, hàm trả về giá trị 0.0, biểu thị cho việc hàm không được xác định hoặc không tồn tại.

Lưu ý: Bước 4 trả về 0.0 để thuận tiện cho các ngôn ngữ như C, yêu cầu định nghĩa rõ kiểu giá trị hàm trả về. Nhược điểm của điều này là không tính được chính xác nếu hàm số có vấn đề. Tuy nhiên như đã đề cập trước đó, chúng ta mặc định hàm $f(x)$ là xác định trên khoảng cần tính nên việc trả về 0.0 có ý nghĩa kiểm tra để phục vụ việc sự nghi ngờ đã đề cập từ trước (ý nghĩa của việc tính $f(x)$ tại một điểm ngoài phục vụ tính tích phân)

3.6. Tính toán giá trị của biến ẩn

Function evaluate_variable

Đầu vào: con trỏ trỏ tới chuỗi tên biến ẩn, biến ẩn, số lượng biến ẩn

Đầu ra: giá trị của biến ẩn

Ý tưởng: Hàm này tìm và trả về giá trị của một biến dựa trên tên của biến đó. Nó tìm kiếm trong một mảng các biến được truyền vào hàm.

1. Duyệt qua danh sách các biến: Hàm bắt đầu bằng việc lặp qua mỗi biến trong mảng vars, với var_count là số lượng biến trong mảng.

2. So sánh tên biến: Trong mỗi lần lặp, hàm so sánh tên của biến hiện tại (được truy cập thông qua `vars[i].name`) với tên biến được truyền vào hàm (`name`) sử dụng hàm `strcmp`.
3. Trả về giá trị của biến nếu tên khớp: Nếu tên biến khớp (tức là `strcmp` trả về 0), hàm sẽ trả về giá trị của biến đó. Giá trị này được truy cập thông qua con trỏ `vars[i].address` và trả về giá trị mà con trỏ này trỏ tới.
4. Trả về 0.0 nếu không tìm thấy biến: Nếu hàm kết thúc vòng lặp mà không tìm thấy biến nào có tên khớp với `name`, hàm sẽ trả về giá trị 0.0, biểu thị rằng biến không được xác định hoặc không tồn tại trong danh sách.

Lưu ý:

- Như đã đề cập ở trước đó, chúng ta đang thiết kế một chương trình “nửa vờ” khi đáng lẽ có thể tính toán giá trị của hàm `f` theo nhiều biến nhưng chúng ta chỉ tính theo một biến để làm đơn giản chương trình.
- Việc trả về 0.0 có ý nghĩa tương tự như hàm `evaluate_function`, mặc định chương trình của chúng ta sẽ không chạy tới lệnh đó.

IV. Tính tích phân

Như đã xác định rõ ở mục lớn 1, chúng ta sẽ có các hàm và ý nghĩa của chúng như sau:

1. `trapezoidal_rule`: Hàm này thực hiện tính toán tích phân của một hàm số dựa trên công thức hình thang dựa trên bộ điểm cho trước.
2. `simpson_rule`: Hàm này thực hiện tính toán tích phân của một hàm số dựa trên công thức Simpson dựa trên bộ điểm cho trước.
3. `integrate`: Hàm này để tính tích phân (theo phương pháp hình thang hoặc simpson) của một hàm số cho trước trong một khoảng xác định từ `a` đến `b` với một sai số chấp nhận được `epsilon`. Hàm này điều chỉnh số lượng điểm được sử dụng để tính toán tích phân cho đến khi sai số giữa hai lần tính toán liên tiếp nhỏ hơn `epsilon`.

4.1. Tính tích phân dùng công thức hình thang với bộ điểm cho trước

Function `trapezoidal_rule`

Đầu vào: số thực `a`, `b`, con trỏ trỏ tới mảng giá trị của bộ `x(x_values)`, con trỏ trỏ tới mảng giá trị của bộ `y(y_values)`, số mốc `n`

Đầu ra: kết quả tích phân

Ý tưởng: theo ý tưởng của việc tính tích phân theo công thức hình thang đã đề cập trước đó

1. Tính kích thước bước (h): Đầu tiên, hàm tính kích thước bước h bằng cách lấy khoảng cách giữa a và b (điểm đầu và điểm cuối của khoảng tích phân) và chia cho n, số lượng phân đoạn mà khoảng này được chia thành.
2. Khởi tạo tổng (s): Tổng s được khởi tạo bằng tổng giá trị của hàm tại điểm đầu tiên ($y_values[0]$) và điểm cuối cùng ($y_values[n]$).
3. Tính tổng các giá trị hàm ở các điểm trung gian: Hàm sau đó lặp qua mỗi điểm trung gian (từ $i = 1$ đến $i = n-1$) và cộng dồn vào s gấp đôi giá trị của hàm tại mỗi điểm đó ($2 * y_values[i]$).
4. Tính và trả về kết quả: Cuối cùng, hàm nhân tổng s với $h/2$ để tính diện tích tổng cộng dưới đồ thị của hàm số, theo quy tắc hình thang, và trả về giá trị này.

4.2. Tính tích phân dùng công thức Simpson với bộ điểm cho trước

Function `simpson_rule`

Đầu vào, đầu ra như hàm `trapezoidal_rule`

Ý tưởng: theo ý tưởng của việc tính tích phân theo công thức simpson 1/3 đã đề cập trước đó

1. Tính kích thước bước (h): Đầu tiên, hàm tính kích thước bước h bằng cách lấy khoảng cách giữa a và b (điểm đầu và điểm cuối của khoảng tích phân) và chia cho n, số lượng phân đoạn mà khoảng này được chia thành.
 2. Khởi tạo tổng (s): Tổng s được khởi tạo bằng tổng giá trị của hàm tại điểm đầu tiên ($y_values[0]$) và điểm cuối cùng ($y_values[n]$). Điều này tương ứng với việc lấy giá trị của hàm tại hai đầu mút của khoảng tích phân.
 3. Tính tổng các giá trị hàm ở các điểm trung gian: Hàm sau đó lặp qua mỗi điểm trung gian (từ $i = 1$ đến $i = n-1$) và cộng dồn vào s giá trị của hàm tại mỗi điểm đó, nhưng với một quy tắc đặc biệt: nếu i là số chẵn, giá trị của hàm tại điểm đó được nhân với 2; nếu i là số lẻ, giá trị của hàm tại điểm đó được nhân với 4.
 4. Tính và trả về kết quả: Cuối cùng, hàm nhân tổng s với $h/3$ để tính diện tích tổng cộng dưới đồ thị của hàm số, theo quy tắc Simpson, và trả về giá trị này.
1. Tính kích thước bước (h): Đầu tiên, hàm tính kích thước bước h bằng cách lấy khoảng cách giữa a và b (điểm đầu và điểm cuối của khoảng tích phân) và chia cho n, số lượng phân đoạn mà khoảng này được chia thành.
 2. Khởi tạo tổng (s): Tổng s được khởi tạo bằng tổng giá trị của hàm tại điểm đầu tiên ($y_values[0]$) và điểm cuối cùng ($y_values[n]$).

3. Tính tổng các giá trị hàm ở các điểm trung gian: Hàm sau đó lặp qua mỗi điểm trung gian (từ $i = 1$ đến $i = n-1$) và cộng dồn vào s giá trị của hàm tại mỗi điểm đó, nhưng với một quy tắc đặc biệt: nếu i là số chẵn, giá trị của hàm tại điểm đó được nhân với 2; nếu i là số lẻ, giá trị của hàm tại điểm đó được nhân với 4.
4. Tính và trả về kết quả: Cuối cùng, hàm nhân tổng s với $h/3$ để tính diện tích tổng cộng dưới đồ thị của hàm số, theo quy tắc Simpson, và trả về giá trị này

4.3. Tính tích phân theo phương pháp mong muốn

Function integrate

Đầu vào: ba số thực a , b , ϵ , con trỏ trỏ tới chuỗi hàm $f(x)$, biến cờ flag

Đầu ra: Giá trị tích phân mong muốn(số thực)

1. Khởi tạo số điểm và cấp phát bộ nhớ: Bắt đầu với 3 điểm ($\text{num_points} = 3$) và cấp phát bộ nhớ cho mảng x_values và y_values để lưu trữ các giá trị x và y tương ứng.
2. Tính toán kích thước bước (step): Tính kích thước bước dựa trên khoảng cách giữa a và b chia cho số điểm trừ 1.
3. Khởi tạo biến và tính giá trị x , y ban đầu:
 - 3.1. Duyệt qua mỗi điểm từ 0 đến $\text{num_points} - 1$.
 - 3.2. Tính giá trị x cho mỗi điểm dựa trên a , i , và step .
 - 3.3. Sử dụng giá trị x này để tính giá trị y tương ứng thông qua hàm parse_expression , đòi hỏi một biểu thức hàm và một mảng các biến.
4. Tính toán kết quả tích phân ban đầu sử dụng quy tắc hình thang hoặc Simpson dựa trên giá trị của flag .
5. Lặp lại việc tinh chỉnh kết quả:
 - 5.1. Lưu kết quả tích phân hiện tại vào old_result .
 - 5.2. Tăng số điểm (num_points) lên gấp đôi.
 - 5.3. Cấp phát lại bộ nhớ cho x_values và y_values để phản ánh số điểm mới.
 - 5.4. Tính lại step dựa trên số điểm mới.
 - 5.5. Tính lại các giá trị x và y cho mỗi điểm mới.
 - 5.6. Tính lại kết quả tích phân sử dụng quy tắc hình thang hoặc Simpson với số điểm mới.
 - 5.7. Lặp lại quá trình này cho đến khi sự khác biệt giữa kết quả mới và old_result nhỏ hơn ϵ .
6. Giải phóng bộ nhớ: Sau khi hoàn thành vòng lặp, giải phóng bộ nhớ đã cấp phát cho x_values và y_values .

7. Trả về kết quả: Hàm trả về giá trị tích phân cuối cùng sau khi đạt được độ chính xác mong muốn.

V. Chương trình chính

1. Khởi tạo các biến row và col để theo dõi vị trí hiện tại của mục menu được nổi bật.
2. Khởi tạo môi trường ncurses để thiết lập môi trường cho việc hiển thị và điều khiển giao diện người dùng trong terminal.
3. Hiển thị menu ban đầu bằng cách gọi hàm `print_menu(row, col)` và sau đó làm mới màn hình với `refresh()`.
4. Vào một vòng lặp vô tận để xử lý nhập liệu từ người dùng:
 - 4.1. Sử dụng hàm nào đó (trong C là `getch()`) để lấy ký tự nhập từ bàn phím.
 - 4.2. Dùng câu lệnh `switch` để xác định phím được nhấn và thực hiện các hành động tương ứng:
 - 4.2.1. `KEY_UP`: Di chuyển lựa chọn lên trên. Nếu đang ở mục đầu tiên, chuyển đến mục cuối cùng.
 - 4.2.2. `KEY_DOWN`: Di chuyển lựa chọn xuống dưới. Nếu đang ở mục cuối cùng, chuyển đến mục đầu tiên. Có điều kiện đặc biệt khi row là 0 và col là 1.
 - 4.2.3. `KEY_LEFT`: Di chuyển lựa chọn sang trái nếu col là 1.
 - 4.2.4. `KEY_RIGHT`: Di chuyển lựa chọn sang phải nếu col là 0 và row không phải là 1, 2, hoặc 4.
 - 4.2.5. `'\n'`: Kiểm tra và gọi hàm xử lý (handler) cho lựa chọn hiện tại nếu tồn tại.
 - 4.3. Gọi lại `print_menu(row, col)` để cập nhật và hiển thị menu với lựa chọn mới được nổi bật.
 - 4.4. Kiểm tra và hiển thị thông tin thêm nếu có, bao gồm:
 - 4.4.1. In ra hàm số nếu biến `function` không rỗng.
 - 4.4.2. Nếu `flag2` là 1, tính toán và hiển thị giá trị của hàm số tại điểm `x` sử dụng biểu thức trong `function`.
 - 4.4.3. Nếu `flag3` không bằng 0, hiển thị kết quả tích phân của hàm số trên một khoảng nhất định, sử dụng phương pháp hình thang hoặc Simpson tùy thuộc vào giá trị của `flag3`.
5. Lặp lại vòng lặp vô tận cho đến khi người dùng chọn thoát khỏi chương trình

VI. Phát triển thêm

Cho tới thời điểm hiện tại, chỉ có $f(x)$ là có thể nhập từ file có sẵn và khi tính tích phân ta vẫn chưa biết được thật sự các bước của chúng ta như thế nào. Theo như yêu cầu của bài tập lớn 2 điều này là phải có. Nhưng được cho vào phần phát triển thêm để coi như nói sơ qua việc chương trình muốn mở rộng thì cần làm như thế nào.

Nhập a , b , e , precision và $f(x)$ từ file

Nếu muốn thực hiện nhập a , b , e , precision từ file (ở trường hợp của chúng ta là file txt), để tránh cho việc đặt câu hỏi đâu là a , b , e , precision và $f(x)$ trong file đầu vào kia. Ta sẽ định nghĩa rõ cấu trúc file đầu vào như sau:

- Các dòng sẽ khai báo rõ những gì muốn nhập

VD: muốn nhập $f(x)$ là hàm $3x^2+1$, a là 2

$$a = 2$$

$$f(x) = 3x^2+1$$

Lưu ý:

- Việc định rõ cấu trúc như vậy nhưng không nói phải yêu cầu hết các tham số a , b , e , precision, $f(x)$.
- Để nói lỏng các cách nhập mà vẫn khiến các tham số nhận vào một cách đúng đắn, ta sẽ thêm một hàm nữa để xóa các dấu cách trong một dòng. Lý giải cho việc này là nếu người có thể gõ " $f(x) = 3x^2$ " hoặc " $f(x)=3x^2$ " hoặc " $f(x)=3x^2$ " thì hàm tính toán $f(x)$ tại 1 điểm của chúng ta có thể sai (nếu chúng ta đọc chuỗi $f(x)$ là " $3x^2$ " thay vì " $3x^2$ ") do hàm tính không có khả năng nhận diện dấu cách.

Sau đây là hàm xóa dấu cách trong 1 dòng:

Function `remove_space`

Đầu vào: con trỏ chuỗi s

1. Khởi tạo con trỏ d trùng với con trỏ s .
2. Lặp qua chuỗi s :
 - 2.1. Bỏ qua tất cả các khoảng trắng bằng cách tăng con trỏ s .
 - 2.2. Sao chép ký tự hiện tại từ s sang d .
 - 2.3. Tăng cả hai con trỏ s và d .
 - 2.4. Lặp cho đến khi gặp ký tự kết thúc chuỗi (`'\0'`).
3. Chuỗi s giờ đây không còn khoảng trắng.

Để tiện cho việc không phải tạo thêm một lựa chọn cho menu nữa và tính năng mà chúng ta sắp làm sau đây cũng có nhập hàm $f(x)$ nên ta sẽ chỉ cần sửa hàm

handle_option_6 là được. Sau đây là hàm handle_option_6 sau khi chỉnh sửa(đầu vào và đầu ra vẫn vậy)

1. Khởi tạo một mảng filename với kích thước 100 ký tự để lưu tên file và một mảng line với kích thước 256 ký tự để lưu từng dòng đọc được từ file.
2. Yêu cầu người dùng nhập tên file và lưu vào biến filename.
3. Mở file với tên được chỉ định trong biến filename ở chế độ đọc. Nếu không thể mở file, hiển thị thông báo lỗi, làm mới màn hình, đợi 1 giây rồi thoát khỏi hàm.
4. Đọc từng dòng trong file cho đến khi kết thúc file. Với mỗi dòng đọc được:
 - 4.1. Loại bỏ ký tự xuống dòng cuối cùng nếu có.
 - 4.2. Gọi hàm remove_spaces để loại bỏ tất cả khoảng trắng trong dòng.
 - 4.3. Kiểm tra nội dung của dòng đó:
 - 4.3.1. Nếu dòng bắt đầu bằng "f(x)=", sao chép phần còn lại của dòng sau "f(x)=" vào biến function.
 - 4.3.2. Nếu dòng bắt đầu bằng "a=", đọc giá trị số thực sau "a=" và lưu vào biến a, đánh dấu a đã được nhập(gán a_entered = true).
 - 4.3.3. Nếu dòng bắt đầu bằng "b=", đọc giá trị số thực sau "b=" và lưu vào biến b, đánh dấu b đã được nhập(gán b_entered = true).
 - 4.3.4. Nếu dòng bắt đầu bằng "e=", đọc giá trị số thực sau "e=" và lưu vào biến e, đánh dấu e đã được nhập(gán e_entered = true).
 - 4.3.5. Nếu dòng bắt đầu bằng "precision=", đọc giá trị số nguyên sau "precision=" và lưu vào biến precision, đánh dấu precision đã được nhập(gán precision_entered = true).
5. Sau khi đã đọc và xử lý hết tất cả các dòng trong file, đóng file.

Ở đây, chúng ta có dùng thêm tới 3 biến toàn cục nữa để xác định xem a, b, e, precision đã được nhập hay chưa, điều này là hữu ích khi chúng ta có thể linh động cho thao tác nhập liệu của chúng ta. Tuy nhiên, chính vì vậy nên ở chương trình chính, ngoài phải kiểm tra f(x) có rỗng hay ta cần kiểm tra xem a, b, e, precision đã nhập hay chưa. Và để tiện lợi cho việc người dùng chương trình kiểm tra thực tế a, b, e, precision, chúng ta sẽ in ra nếu chúng đã được nhập. Ngoài ra, để tránh nhầm lẫn nếu chúng ta đã thực hiện hàm reset mà vẫn còn in dùng a, b, e precision cũ thì ở hàm reset chúng ta sẽ gán các biến a_entered, b_entered, e_entered, precision_entered là false

Ghi lại các bước trung gian

Để thực hiện điều này ta cần xác định rõ mình cần xem những gì, ở trong báo cáo này, có lẽ chúng ta cần xem hết mọi thứ từ việc f(x) là hàm gì, tính toán thử các điểm nghi ngờ ta nhận được giá trị như thế nào, khi tính tích phân, các bảng giá

trị được chia như thế nào. Tóm lại, chúng ta sẽ làm một file để ghi lại nhiều thứ. Vì thế việc tạo ra hàm ghi lại những điều mình muốn, khi muốn ghi thì chỉ cần gọi hàm là xong. Hàm `write_log` sẽ làm nhiệm vụ này

Function `write_log`

Đầu vào: con trỏ hằng kiểu ký tự trỏ tới message

1. Mở file ở chế độ append
2. Nếu mở file thành công:
 - 2.1. Sử dụng hàm để ghi lại thông điệp(message) vào file
 - 2.2. Thêm ký tự xuống dòng ở cuối
 - 2.3. Đóng file

Việc có hàm `write_log` sẽ giúp chúng ta làm được những công việc sau:

- + Ghi lại hàm $f(x)$
- + Ghi lại giá trị của f tại điểm x nào đó
- + Làm “nguyên liệu” cho việc ghi lại các bảng giá trị khi tính tích phân

Cách để ghi lại $f(x)$ vào file ghi(có thể coi là `output.txt`):

- Ở hàm `handle_option_1`(hàm yêu cầu nhập $f(x)$ từ bàn phím và giúp người dùng thực hiện yêu cầu đó), sau khi thực hiện hàm `get_input_string` cho biến `function` Ta cần thêm các câu lệnh theo đúng thứ tự sau:
 1. Khai báo một mảng kiểu ký tự(hoặc một chuỗi ký tự) với số lượng tối đa đủ dùng
 2. Ghi vào mảng(hoặc chuỗi) nội dung của `function`
 3. Gọi tới hàm `write_log` với tham số là mảng đã khai báo để ghi `function`

Sau khi làm những điều này, ta tiếp tục làm các câu lệnh phía sau như bình thường. Lưu ý nhỏ ở bước 2, vì các ngôn ngữ lập trình để trang bị các hàm ghi sẵn chỉ cần truyền tham số nên chúng ta không bàn luận sâu và chỉ ghi đơn giản vậy.

- Ở hàm `handle_option_6`, sau khi thực hiện thao tác sao chép với biến `function`, ta làm tương tự như `handle_option_1`

Cách để ghi lại giá trị của f tại điểm x nào đó vào file

Vì việc chỉ tính một giá trị để kiểm tra chỉ xảy ra ở `handle_option_2` nên ta chỉ cần sửa nó đôi chút. Sau khi tính toán xong giá trị của f tại điểm x đã nhập, giống như sửa đổi ở hàm `handle_option_1`:

1. Khai báo một mảng kiểu ký tự(hoặc một chuỗi ký tự) với số lượng tối đa đủ dùng
2. Ghi vào mảng(hoặc chuỗi) nội dung $f(x) = \text{result}$, giả sử $x = 2$, $\text{result} = 3$ thì câu lệnh sẽ ghi lại $f(2) = 3$.
3. Gọi tới hàm `write_log` với tham số là mảng đã khai báo để ghi `function`

Như vậy, chúng ta đã hoàn thành được 2/3 tính năng ghi lại các bước trung gian, ta chỉ thiết việc ghi lại các bảng giá trị x, y dùng để tính tích phân là xong. Và để dễ dàng cho việc quan sát và việc quan sát bảng giá trị cũng không cần khổ khế quá nên ở báo cáo này chúng ta sẽ trình bày cách ghi lại tối đa 6 mốc x, y(bao gồm 3 mốc đầu và 3 mốc cuối), nếu ít hơn thì ghi lại hết. Muốn làm được vậy, ta chỉ cần một hàm sau.

Function print_table:

Đầu vào: mảng x, mảng y, số mốc n

Ý tưởng: Ở phần tử thứ 4 sẽ không ghi mà thay thế bằng việc ghi "...\\t", còn ở phần tử có thứ tự nhỏ hơn 3 hoặc thứ tự lớn hơn n-3 thì sẽ ghi như bình thường

1. Khởi tạo chuỗi log_message: Một chuỗi ký tự log_message được khởi tạo với kích thước 1000 ký tự và được điền bằng chuỗi rỗng. Chuỗi này sẽ được sử dụng để lưu trữ thông điệp log cuối cùng.
2. Thêm tiêu đề cho giá trị x vào log_message: Chuỗi "x: " được nối vào đầu log_message để chỉ ra rằng các giá trị tiếp theo là giá trị của x.
3. Duyệt và thêm giá trị x vào log_message:
 - 3.1. Duyệt qua mảng x_values sử dụng một vòng lặp for theo chỉ số i.
 - 3.2. Đối với mỗi phần tử, kiểm tra nếu nó nằm trong 3 phần tử đầu tiên hoặc 3 phần tử cuối cùng của mảng.
 - 3.3. Đối với các phần tử này, định dạng giá trị của chúng thành chuỗi với độ chính xác 2 chữ số thập phân và nối chuỗi đó vào log_message.
 - 3.4. Nếu đang xử lý phần tử thứ 4 ($i == 3$), thêm chuỗi "...\\t" vào log_message để chỉ ra rằng có một số giá trị không được hiển thị.
4. Thêm một dòng mới vào log_message để tách biệt giữa các giá trị x và y.
5. Thêm tiêu đề cho giá trị y vào log_message tương tự như đã làm với giá trị x.
6. Duyệt và thêm giá trị y vào log_message theo cùng một cách đã làm với giá trị x.
7. Thêm một dòng mới vào cuối log_message.
8. Ghi log_message vào file log sử dụng hàm write_log.

Khi đã làm được hàm print_table này chúng ta sẽ sửa đôi chút ở hàm tích tích phân. Ngay sau bước 3 của hàm integrate, ta thực hiện các thao tác sau:

1. Khai báo chuỗi log_message
2. Ghi lại với nội dung thể hiện lần đầu tiên thực hiện vào chuỗi log_message
3. Dùng hàm write_log với tham số log_message
4. Gọi tới hàm print_table với tham số x_values, y_values, num_points

Tiếp theo, ở trong bước 5, sau khi tính tích phân, ta thực hiện ghi đè lên `log_message` nội dung lần thực hiện là bao nhiêu rồi tiếp tục như việc ghi lần đầu tiên. Lưu ý, chúng ta có công thức của số lần đã tính tích phân và có thể dùng nó để dễ dàng dùng nó cho việc ghi số lần thực hiện. Công thức đó là:

$$\text{Số lần} = \left\lceil \log_2 \left(2 \frac{\text{num point}}{3} \right) \right\rceil$$

VII. Tổng kết

Như vậy, chúng ta đã thiết kế xong chương trình đơn giản phục vụ cho việc tính tích phân. Sau đây là các ưu khuyết điểm chương trình của chúng ta:

Ưu điểm	<ul style="list-style-type: none"> - Có thể coi là một chương trình tốt cho việc tính tích phân - Khả năng mở rộng, bảo trì cao - Khả năng động của chương trình tốt (các thao tác với menu thiết kế tốt) - Có khả năng hoạt động tốt trên họ Unix
Khuyết điểm	<ul style="list-style-type: none"> - Giả thiết $f(x)$ luôn xác định trên $[a, b]$ mặc dù giúp chương trình đơn giản tuy nhiên việc kiểm tra tính xác định của $f(x)$ luôn cần thiết cho việc tính tích phân nhưng ở chương trình này không làm được. - Việc ghi vào file như hàm <code>write_log</code> có thể không an toàn và sẽ khiến chương trình chạy chậm - Hàm $f(x)$ nhập vào yêu cầu khắt khe về dấu nhân, ta không bỏ được theo các mà chúng ta bỏ qua trong thực tế. - Sai số của phương pháp chưa thực sự chính xác khi chỉ so sánh các giá trị tính trước sau - Muốn hoạt động trên Windows, ta phải thực hiện biên dịch chéo

Sau đây, chúng ta sẽ bàn luận về khuyết điểm và hướng khắc phục:

1. Thao tác ghi vào file:

- Xử lý lỗi: Hàm hiện tại không thông báo cho người gọi biết nếu việc mở file thất bại. Trong một ứng dụng thực tế, có thể cần phải xử lý lỗi một cách cụ thể hơn, ví dụ, thông qua việc trả về một giá trị lỗi hoặc ghi lỗi vào một log hệ thống.
- Hiệu suất: Mở và đóng file mỗi lần ghi log có thể không hiệu quả nếu hàm này được gọi nhiều lần trong một khoảng thời gian ngắn. Một cách tiếp cận khác là giữ file mở trong suốt thời gian chạy của ứng dụng hoặc sử dụng một buffer để ghi nhiều message cùng một lúc.

- Đồng bộ hóa: Nếu thiết kế thành ứng dụng đa luồng và nhiều luồng cùng ghi log vào cùng một file, cần phải đảm bảo rằng việc ghi file được đồng bộ hóa để tránh tình trạng xung đột và dữ liệu bị hỏng.

2. Nhập $f(x)$, tính $f(x)$ tại một điểm:

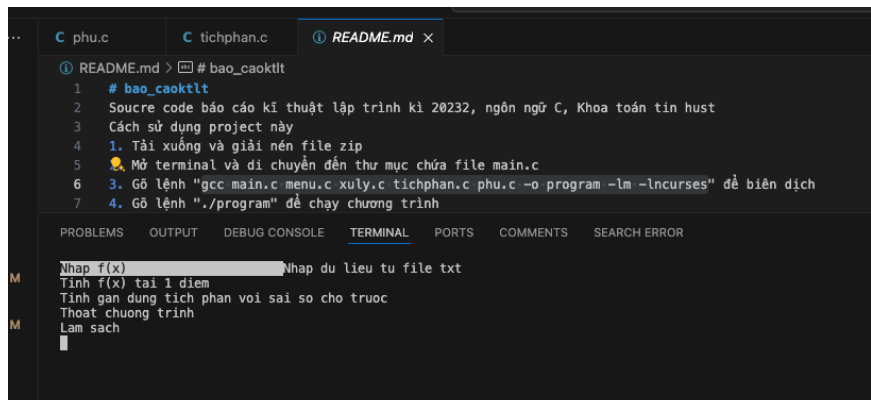
Như đã trình bày trước, việc nhập $f(x)$ và tính $f(x)$ được thiết kế đơn giản và không sử dụng thư viện có sẵn. Để khắc phục nhược điểm đã nêu, ta có thể cân nhắc dùng các thư viện có sẵn cho việc này. Như C thì chúng ta có `tinyexpr.h`, Python thì chúng ta có `Sympy`...

3. Sai số

Việc tính sai số như vậy có thể không tốt nếu gặp các hàm không ổn định. Để giải quyết vấn đề này ta có thể cân nhắc cách tính sai số chính xác dựa trên giá trị lớn nhất của đạo hàm cấp 2 trên đoạn đang xét

VIII. Kiểm thử

8.1. In menu



```

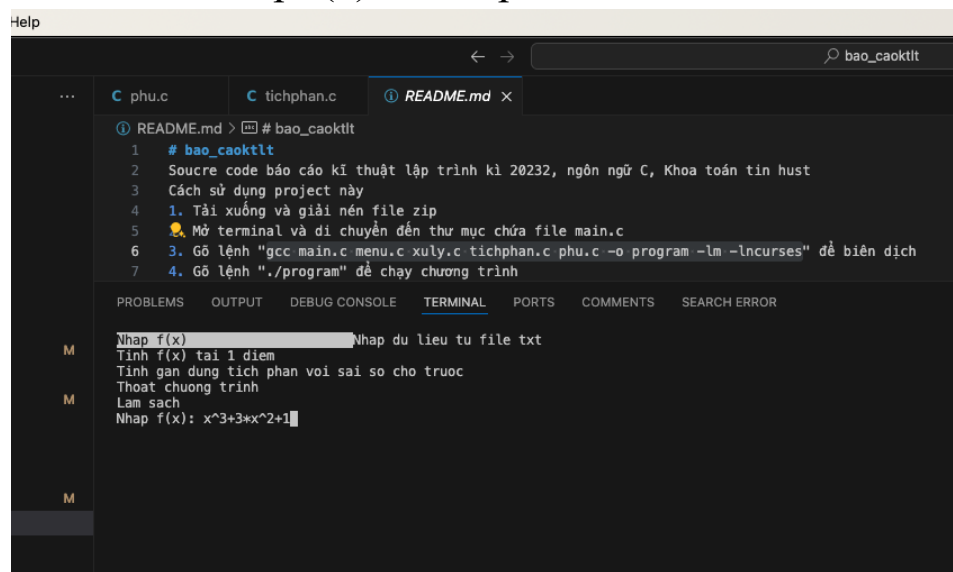
1 # bao_caoktlt
2 Soucre code báo cáo kĩ thuật lập trình kì 20232, ngôn ngữ C, Khoa toán tin hust
3 Cách sử dụng project này
4 1. Tải xuống và giải nén file zip
5 2. Mở terminal và di chuyển đến thư mục chứa file main.c
6 3. Gõ lệnh "gcc main.c menu.c xuly.c tíchphan.c phu.c -o program -lm -lcurses" để biên dịch
7 4. Gõ lệnh "./program" để chạy chương trình

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SEARCH ERROR

Nhập f(x)
Tính f(x) tại 1 điểm
Tính gần đúng tích phân với sai số cho trước
Thoát chương trình
Làm sạch

```

8.2. Nhập $f(x)$ từ bàn phím



```

1 # bao_caoktlt
2 Soucre code báo cáo kĩ thuật lập trình kì 20232, ngôn ngữ C, Khoa toán tin hust
3 Cách sử dụng project này
4 1. Tải xuống và giải nén file zip
5 2. Mở terminal và di chuyển đến thư mục chứa file main.c
6 3. Gõ lệnh "gcc main.c menu.c xuly.c tíchphan.c phu.c -o program -lm -lcurses" để biên dịch
7 4. Gõ lệnh "./program" để chạy chương trình

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SEARCH ERROR

Nhập f(x)
Tính f(x) tại 1 điểm
Tính gần đúng tích phân với sai số cho trước
Thoát chương trình
Làm sạch
Nhập f(x): x^3+3*x^2+1

```

```
# bao_caoktlt
Soucre code báo cáo kĩ thuật lập trình kì 20232, ngôn ngữ C, Khoa toán tin hust
Cách sử dụng project này
1. Tải xuống và giải nén file zip
2. Mở terminal và di chuyển đến thư mục chứa file main.c
3. Gõ lệnh "gcc main.c menu.c xuly.c tichphan.c phu.c -o program -lm -lncurses" để biên dịch
4. Gõ lệnh "./program" để chạy chương trình
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS COMMENTS SEARCH ERROR

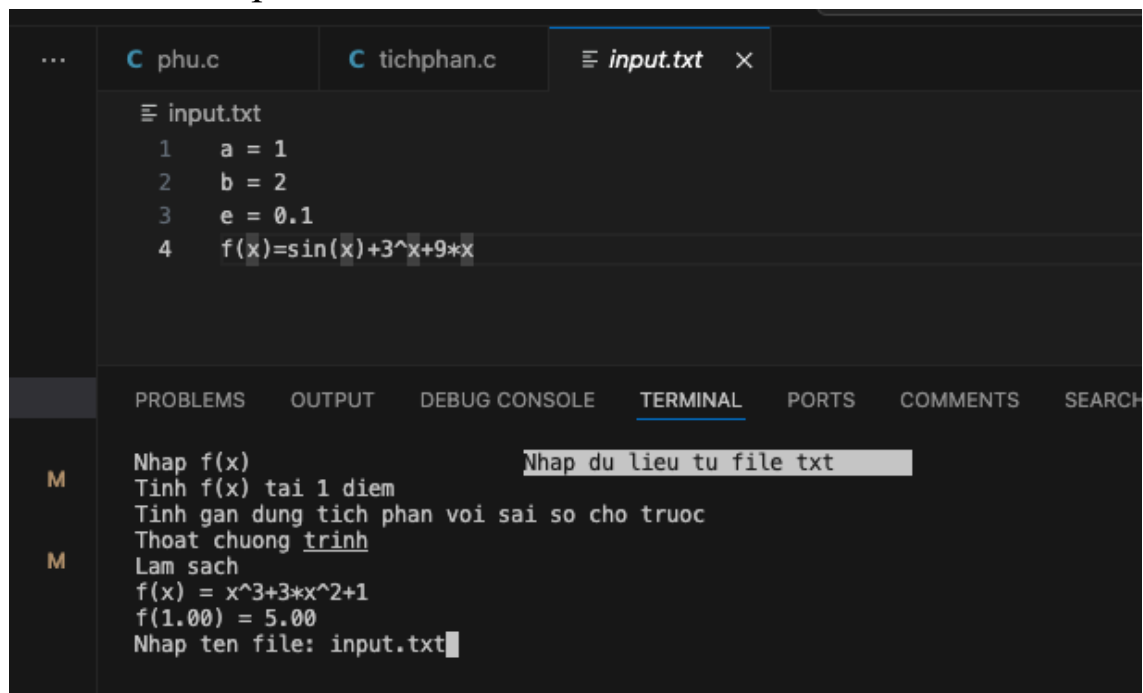
Nhap f(x) Nhap du lieu tu file txt
Tinh f(x) tai 1 diem
Tinh gan dung tich phan voi sai so cho truoac
Thoat chuong trinh
Lam sach
f(x) = x^3+3*x^2+1

8.3. Tính $f(x)$ tại 1 điểm, điểm đó nhập từ bàn phím

```
Nhap f(x) Nhap du lieu tu file txt
Tinh f(x) tai 1 diem
Tinh gan dung tich phan voi sai so cho truoac
Thoat chuong trinh
Lam sach
f(x) = x^3+3*x^2+1
Nhap diem can tinh: 1
```

```
Nhap f(x) Nhap du lieu tu file txt
Tinh f(x) tai 1 diem
Tinh gan dung tich phan voi sai so cho truoac
Thoat chuong trinh
Lam sach
f(x) = x^3+3*x^2+1
f(1.00) = 5.00
```


8.4. Nhập dữ liệu từ file txt

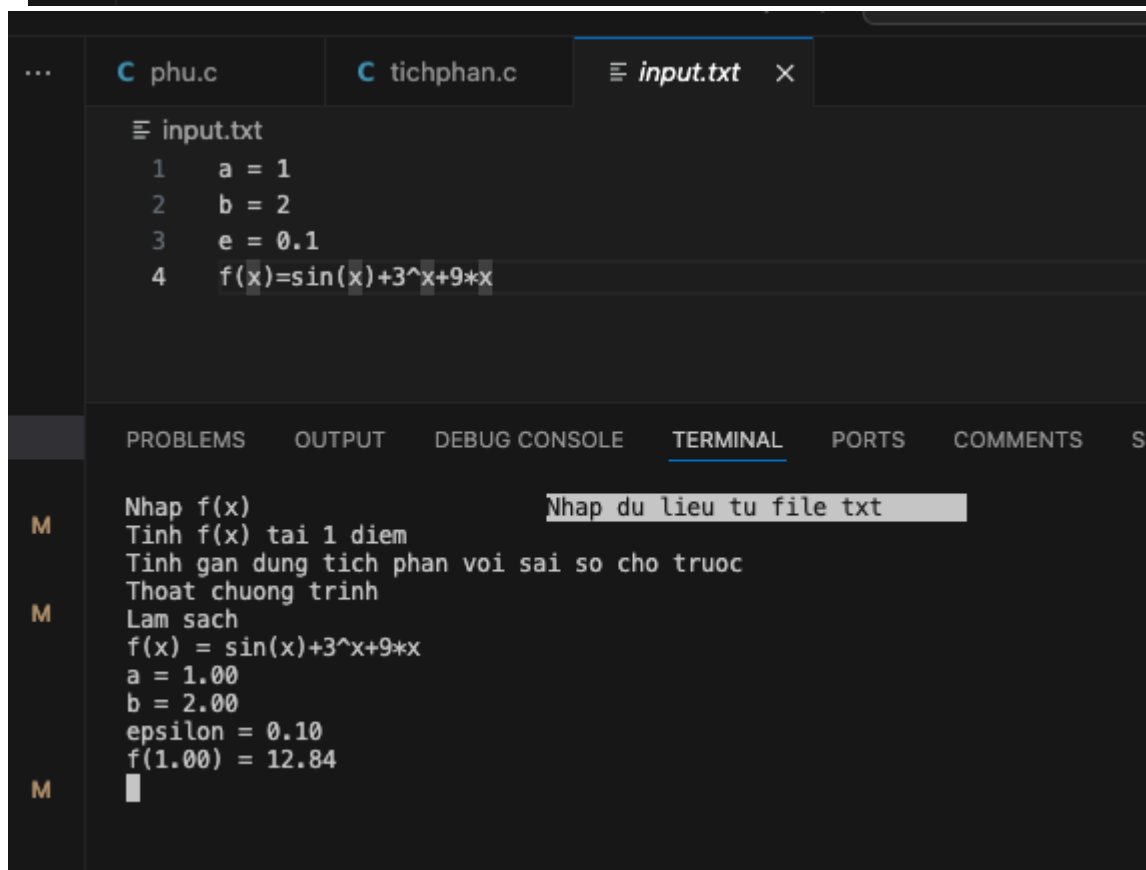


The screenshot shows a code editor with three tabs: `phu.c`, `tichphan.c`, and `input.txt`. The `input.txt` file contains the following code:

```
1 a = 1
2 b = 2
3 e = 0.1
4 f(x)=sin(x)+3*x+9*x
```

Below the editor, the `TERMINAL` tab is active, displaying the program's output:

```
Nhap f(x)
Tinh f(x) tai 1 diem
Tinh gan dung tich phan voi sai so cho truoac
Thoat chuong trinh
Lam sach
f(x) = x^3+3*x^2+1
f(1.00) = 5.00
Nhap ten file: input.txt
```



The screenshot shows the same code editor with the `input.txt` file. The `TERMINAL` tab displays the program's output for a second run:

```
Nhap f(x)
Tinh f(x) tai 1 diem
Tinh gan dung tich phan voi sai so cho truoac
Thoat chuong trinh
Lam sach
f(x) = sin(x)+3*x+9*x
a = 1.00
b = 2.00
epsilon = 0.10
f(1.00) = 12.84
```

8.5. Tính tích phân theo Simpson

Như chúng ta thấy ở file input chưa cho biết precision nên khi nhấn chọn tính tích phân, chương trình sẽ yêu cầu nhập precision

...

phu.c

tichphan.c

input.txt

input.txt

1 a = 1

2 b = 2

3 e = 0.1

4 f(x)=sin(x)+3^x+9*x

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

COMMENTS

M

Nhap f(x)

Tinh f(x) tai 1 diem

Tinh gan dung tich phan voi sai so cho truoc

Thoat chuong trinh

M

Lam sach

f(x) = sin(x)+3^x+9*x

a = 1.00

b = 2.00

epsilon = 0.10

f(1.00) = 12.84

M

Nhap so chu so thap phan: 4

← →

...

phu.c

tichphan.c

input.txt

input.txt

1 a = 1

2 b = 2

3 e = 0.1

4 f(x)=sin(x)+3^x+9*x

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

M

Phuong phap hinh thang

Phuong phap Simpson

M

```
input.txt
1  a = ~/Documents/bao_caoktlt/input.txt
2  b = 2
3  e = 0.1
4  f(x)=sin(x)+3^x+9*x

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SEARCH ERROR

Nhap f(x)                                Nhap du lieu tu file txt
Tinh f(x) tai 1 diem
Tinh gan dung tích phân voi sai so cho truoc
Thoat chương trình
Lam sach
f(x) = sin(x)+3^x+9*x
a = 1.00
b = 2.00
epsilon = 0.10
precision = 4
f(1.00) = 12.84
Tích phân của f(x) tren [1.00, 2.00] voi sai so khong qua 0.10 theo pp simpson la: 19.8319
```

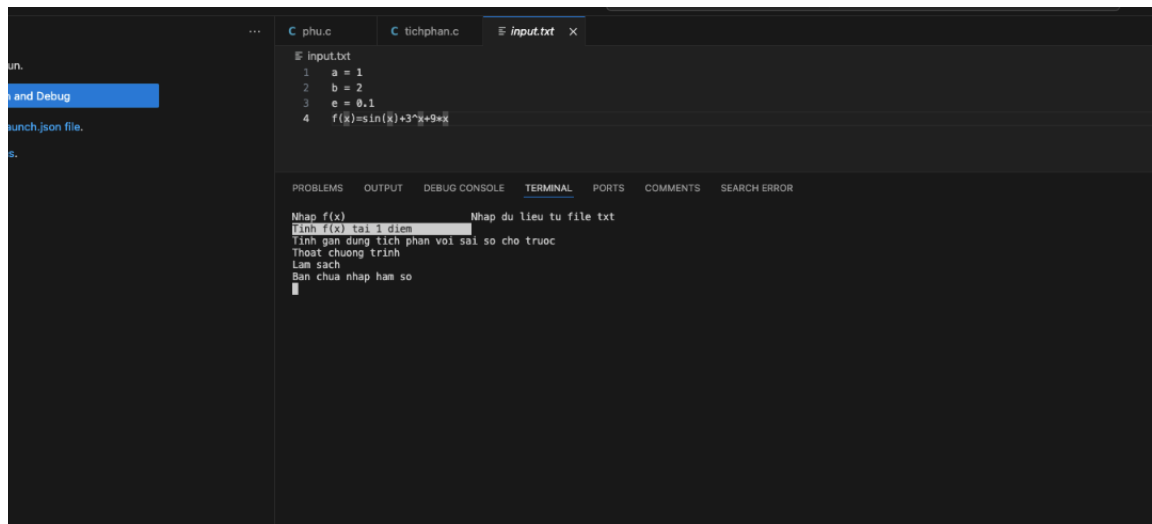
8.6. Làm sạch

```
3  e = 0.1
4  f(x)=sin(x)+3^x+9*x

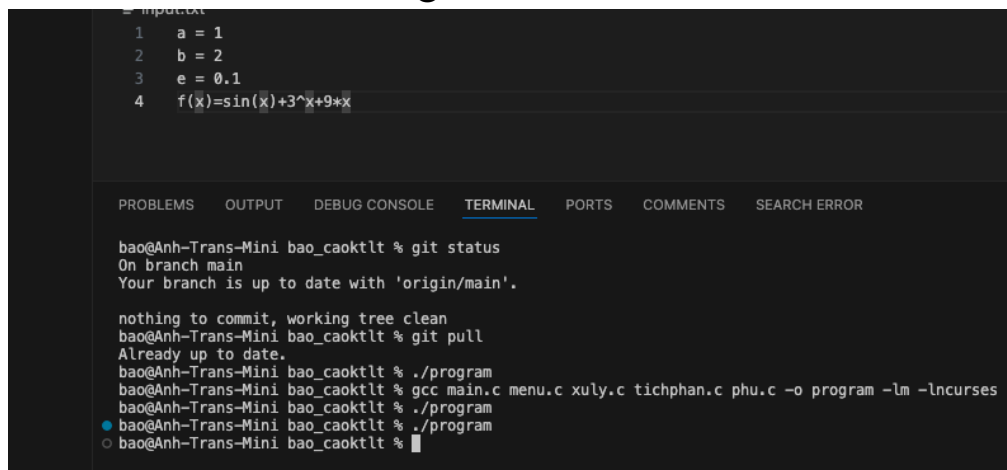
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Nhap f(x)                                Nhap du lieu tu file txt
Tinh f(x) tai 1 diem
Tinh gan dung tích phân voi sai so cho truoc
Thoat chương trình
Lam sach
```

Khi đã làm sạch mà chúng ta nếu cố gắng chọn tính $f(x)$ hay tính gần đúng tích phân sẽ hiện như sau



8.7. Thoát chương trình



Mục lục

Lời mở đầu.....	2
I. Xây dựng module chương trình	3
1.1. Menu.....	3
1.2. Tính giá trị $f(x)$ tại một điểm cho trước	3
1.3. Tính tích phân	4
1.4. Lưu ý khi đọc báo cáo	6
II. Thiết kế menu.....	6
2.1. In ra menu	7
2.2. Nhập một chuỗi từ bàn phím.....	8
2.3. Nhập một số thực từ bàn phím	8
2.4. Thao tác với các lựa chọn	9
2.5. Các hàm thực hiện ứng với các lựa chọn.....	10
2.6. In ra menu con cho việc lựa chọn cách tính tích phân.....	12
III. Tính giá trị $f(x)$ tại một điểm	13
3.1. Phân tích biểu thức và tính cộng trừ	14
3.2. Phân tích và tính toán các phép nhân chia	14
3.3. Phân tích và tính toán các phép lấy lũy thừa	15
3.4. Phân tích biểu thức và tính toán các đơn vị cơ bản trong biểu thức	16
3.5. Tính toán giá trị của các hàm cơ bản như hàm lượng giác, logarit,..	16
3.6. Tính toán giá trị của biến ẩn.....	17
IV. Tính tích phân	18
4.1. Tính tích phân dùng công thức hình thang với bộ điểm cho trước.....	18
4.2. Tính tích phân dùng công thức Simpson với bộ điểm cho trước.....	19
4.3. Tính tích phân theo phương pháp mong muốn	20
V. Chương trình chính.....	21
VI. Phát triển thêm.....	22
Nhập a, b, e, precision và $f(x)$ từ file.....	22
Ghi lại các bước trung gian	23
VII. Tổng kết	26
VIII. Kiểm thử.....	27
8.1. In menu.....	27
8.2. Nhập $f(x)$ từ bàn phím	27

8.3.	Tính $f(x)$ tại 1 điểm, điểm đó nhập từ bàn phím.....	28
8.4.	Nhập dữ liệu từ file txt	30
8.5.	Tính tích phân theo Simpson.....	30
8.6.	Làm sạch.....	32
8.7.	Thoát chương trình	33