

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

BÀI TẬP 1 :

1. Giải thích các thành phần giao diện của Cursor

Cursor là một trình soạn thảo code dựa trên Visual Studio Code, có tích hợp AI hỗ trợ lập trình. Giao diện chính của Cursor gồm các thành phần sau:

1. Thanh Menu (Menu Bar)

Nằm ở trên cùng cửa sổ, gồm các mục như **File, Edit, View, Terminal, Help**.

Dùng để tạo file mới, mở thư mục, lưu file, cấu hình giao diện, mở terminal,...

2. Thanh công cụ bên trái (Activity Bar)

Gồm các biểu tượng:

- **Explorer:** Quản lý file và thư mục
- **Search:** Tìm kiếm trong project
- **Source Control:** Quản lý Git
- **Extensions:** Cài đặt tiện ích
- **AI / Chat:** Khu vực tương tác với AI của Cursor

3. Khu vực soạn thảo (Editor)

Là vùng chính ở giữa màn hình, nơi viết code hoặc nội dung file.

Cursor cho phép AI gợi ý, sửa code, giải thích code ngay trong khu vực này.

4. Thanh trạng thái (Status Bar)

Nằm ở dưới cùng, hiển thị thông tin như:

- Ngôn ngữ lập trình của file
- Trạng thái Git
- Dòng, cột con trỏ
- Trạng thái AI

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

5. Cửa sổ Chat AI

Cho phép người dùng nhập prompt để:

- Hỏi kiến thức
- Giải thích code
- Sinh code mới
- Hướng dẫn thao tác trong Cursor

2. Prompt yêu cầu AI chỉ cách tạo file mới và lưu file

Prompt ví dụ :

“Hãy hướng dẫn từng bước cách tạo một file mới trong Cursor và cách lưu file đó. Giải thích đơn giản cho người mới bắt đầu.”

Prompt chi tiết hơn :

“Tôi là người mới sử dụng Cursor. Hãy hướng dẫn tôi từng bước cách tạo file mới, đặt tên file và lưu file vào thư mục. Trình bày rõ ràng theo dạng các bước.”

3. Nhận xét ngắn

Việc dùng prompt rõ ràng giúp AI trả lời chính xác và dễ hiểu hơn. Khi mô tả rõ đối tượng là “người mới bắt đầu” và yêu cầu “từng bước”, kết quả AI đưa ra sẽ phù hợp và dễ áp dụng hơn.

BÀI TẬP 2 :

hàm C để tính giai thừa của số nguyên dương n

```
#include<stdio.h>
```

```
// Hàm tính giai thừa của số nguyên dương n
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
int result (int n){  
  
    int factorial = 1 ;    // Khởi tạo biến lưu giai thừa, bắt đầu từ 1  
  
    for (int i = 1 ; i <= n ; i++){    // Vòng lặp chạy từ 1 đến n  
  
        factorial *= i ; // Nhân dồn factorial với i  
  
    }  
  
    return factorial ;    // Trả về giá trị giai thừa  
  
}  
  
  
int main (){  
  
    int n ;  
  
    printf("Nhap so nguyen n: ");  
  
    scanf("%d",&n);  
  
  
  
  
    printf("Giai thua cua so nguyen duong %d la: %d",n,result(n));  
  
    return 0 ;  
  
}
```

BÀI TẬP 3 :

1. Prompt sử dụng

“So sánh cách Cursor AI sinh code khi viết hàm tính tổng mảng với GitHub Copilot. Cái nào phù hợp hơn cho người mới học lập trình?”

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

2. So sánh Cursor AI và GitHub Copilot khi sinh code

a) Cursor AI

- Cursor AI thường **giải thích trước khi sinh code**
- Code sinh ra có **chú thích rõ ràng**
- Tên biến dễ hiểu, phù hợp với người mới
- Có thể hỏi tiếp: *“giải thích từng dòng”, “sửa lỗi giúp tôi”*

Ví dụ code Cursor AI có thể sinh ra:

```
// Hàm tính tổng các phần tử trong mảng
int sumArray(int arr[], int n) {
    int sum = 0;                // Biến lưu tổng

    for (int i = 0; i < n; i++) {
        sum += arr[i];          // Cộng từng phần tử vào tổng
    }

    return sum;                 // Trả về tổng
}
```

Cursor AI giúp người học **hiểu vì sao code viết như vậy**, không chỉ copy dùng.

b) GitHub Copilot

- GitHub Copilot sinh code **rất nhanh**
- Ít hoặc **không có chú thích**
- Phù hợp với người đã biết lập trình
- Chủ yếu dựa vào **ngữ cảnh code đang viết**

Ví dụ code GitHub Copilot có thể sinh ra:

```
int sumArray(int arr[], int n) {
    int sum = 0;
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
    for (int i = 0; i < n; i++)  
        sum += arr[i];  
    return sum;  
}
```

Code ngắn gọn, đúng, nhưng **không giải thích**, người mới dễ “dùng mà không hiểu”.

3. Bảng so sánh tổng quát

Tiêu chí	Cursor AI	GitHub Copilot
Có giải thích	Có	Không
Chú thích code	Rõ ràng	Ít
Phù hợp người mới	Phù hợp	Không phù hợp
Tốc độ sinh code	Trung bình	Rất nhanh
Khả năng học từ lỗi	Rất tốt	Hạn chế

4. Kết luận

- **Cursor AI** phù hợp hơn cho **người mới học lập trình**, vì:
 - Có giải thích
 - Code dễ đọc
 - Dễ học tư duy lập trình
- **GitHub Copilot** phù hợp với:
 - Lập trình viên đã có nền tảng
 - Muốn viết code nhanh, ít giải thích

Đối với sinh viên mới học C/C++, Cursor AI là lựa chọn phù hợp hơn.

BÀI TẬP 4 :

Prompt sử dụng

“Refactor đoạn code này để ngắn gọn hơn, thêm chú thích từng bước.”

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

Code ban đầu (chưa tối ưu)

```
int sumArray(int arr[], int n) {  
    int sum = 0;  
    for (int i = 0; i < n; i++) {  
        sum = sum + arr[i];  
    }  
    return sum;  
}
```

Code sau khi Cursor AI chỉnh sửa (in-place edit)

```
// Hàm tính tổng các phần tử trong mảng  
int sumArray(int arr[], int n) {  
    int sum = 0; // Biến lưu tổng  
  
    // Duyệt qua từng phần tử trong mảng  
    for (int i = 0; i < n; i++) {  
        sum += arr[i]; // Cộng phần tử hiện tại vào tổng  
    }  
  
    return sum; // Trả về tổng  
}
```

Nhận xét

Cursor AI đã:

- Rút gọn câu lệnh (`sum += arr[i]`)
- Thêm chú thích rõ ràng từng bước
- Giữ nguyên logic, code dễ đọc hơn cho người mới học

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

Đoạn code sau khi refactor **vẫn chạy đúng** và cho kết quả chính xác.

BÀI TẬP 5 :

Prompt sử dụng

“Đây là đoạn code C để kiểm tra số nguyên tố, nhưng đang bị lỗi. Hãy sửa và giải thích lỗi sai.”

1. Đoạn code ban đầu (bị lỗi logic)

```
int isPrime(int n) {  
    for (int i = 2; i <= n; i++) {  
        if (n % i == 0) return 0;  
    }  
    return 1;  
}
```

2. Lỗi trong đoạn code

- **Sai logic vòng lặp:**
 - Khi $i == n$ thì $n \% n == 0$
 - Dẫn đến mọi số $n > 1$ đều bị kết luận là không phải số nguyên tố
- **Không kiểm tra trường hợp $n < 2$**
 - 0 và 1 không phải số nguyên tố

3. Code sau khi Cursor AI chỉnh sửa

```
// Hàm kiểm tra số nguyên tố  
int isPrime(int n) {  
    // Số nhỏ hơn 2 không phải số nguyên tố
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
    if (n < 2) {
        return 0;
    }

    // Chỉ cần kiểm tra đến căn bậc hai của n
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return 0;    // Có ước khác 1 và n → không phải số nguyên
tổ
        }
    }

    return 1;           // Là số nguyên tố
}
```

4. Thao tác trong Cursor

- Sinh viên **chọn Apply / Accept**
- Cursor **thay thế đoạn code cũ bằng code mới**
- Không phát sinh lỗi cú pháp sau khi thay

5. Chạy thử đoạn code đã sửa

```
#include <stdio.h>

int isPrime(int n) {
    if (n < 2) return 0;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}
```


Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
int main() {  
    int n = 7;  
  
    if (isPrime(n))  
        printf("%d la so nguyen to", n);  
    else  
        printf("%d khong phai so nguyen to", n);  
  
    return 0;  
}
```

Kết quả chạy:

Input: 7

Output: 7 la so nguyen to

6. Ghi nhận những gì AI đã tạo ra và chỉnh sửa

- Sửa điều kiện vòng lặp
- Bổ sung xử lý $n < 2$
- Tối ưu bằng cách kiểm tra đến \sqrt{n}
- Thêm chú thích giúp dễ hiểu logic

Bài Tập 6 :

Prompt sử dụng

“Refactor hàm JavaScript sau để tối ưu hơn và thêm comment giải thích từng dòng.”

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

1. Code ban đầu (chưa có chú thích)

```
function reverseArray(arr) {  
  let newArr = [];  
  for (let i = arr.length - 1; i >= 0; i--) {  
    newArr.push(arr[i]);  
  }  
  return newArr;  
}
```

2. Code sau khi Cursor AI refactor (tối ưu + có chú thích)

```
// Hàm đảo ngược thứ tự các phần tử trong mảng  
function reverseArray(arr) {  
  // Tạo một mảng mới để lưu kết quả đảo ngược  
  const reversed = [];  
  
  // Duyệt mảng từ phần tử cuối về đầu  
  for (let i = arr.length - 1; i >= 0; i--) {  
    reversed.push(arr[i]); // Thêm từng phần tử vào mảng mới  
  }  
  
  return reversed; // Trả về mảng đã đảo  
}
```

3. Điểm tối ưu và chỉnh sửa của AI

- Đổi tên biến newArr → reversed (dễ hiểu hơn)
- Thêm **comment** giải thích từng bước
- Giữ nguyên thuật toán, đảm bảo **không thay đổi logic**
- Code rõ ràng, phù hợp cho người mới học JavaScript

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

4. Chạy thử hàm sau khi chỉnh sửa

```
console.log(reverseArray([1, 2, 3, 4]));
```

Kết quả:

[4, 3, 2, 1]

5. Nhận xét

Cursor AI giúp:

- Tối ưu cách trình bày code
- Thêm chú thích dễ hiểu
- Hỗ trợ sinh viên hiểu rõ từng dòng lệnh

Hàm sau khi refactor **vẫn chạy đúng và dễ đọc hơn**.

BÀI TẬP 7 :

1. Prompt tạo project mới

“Hãy tạo một project C nhỏ gồm nhiều file, trong đó có các hàm sumArray được viết theo nhiều cách khác nhau.

- Tạo các file main.c, utils1.c, utils2.c, utils3.c, utils.h
- Mỗi file utilsX.c chứa một phiên bản khác nhau của hàm sumArray (for loop, while loop, recursion).
- main.c gọi cả 3 hàm và in kết quả.”

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

2. Cấu trúc project Cursor AI tạo ra

```
project/  
|— main.c  
|— utils.h  
|— utils1.c    // sumArray dùng for  
|— utils2.c    // sumArray dùng while  
|— utils3.c    // sumArray dùng đệ quy
```

3. Nội dung các file

utils.h

```
#ifndef UTILS_H  
#define UTILS_H  
  
int sumArrayFor(int arr[], int n);  
int sumArrayWhile(int arr[], int n);  
int sumArrayRec(int arr[], int n);  
  
#endif
```

utils1.c (for loop)

```
#include "utils.h"  
  
// Tính tổng mảng bằng vòng lặp for  
int sumArrayFor(int arr[], int n) {  
    int sum = 0;  
    for (int i = 0; i < n; i++) {  
        sum += arr[i];  
    }  
    return sum;  
}
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
}
```

utils2.c (while loop)

```
#include "utils.h"
```

```
// Tính tổng mảng bằng vòng lặp while
```

```
int sumArrayWhile(int arr[], int n) {  
    int sum = 0;  
    int i = 0;  
    while (i < n) {  
        sum += arr[i];  
        i++;  
    }  
    return sum;  
}
```

utils3.c (recursion)

```
#include "utils.h"
```

```
// Tính tổng mảng bằng đệ quy
```

```
int sumArrayRec(int arr[], int n) {  
    if (n == 0) {  
        return 0;  
    }  
    return arr[n - 1] + sumArrayRec(arr, n - 1);  
}
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

main.c

```
#include <stdio.h>
#include "utils.h"

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = 5;

    printf("Sum (for): %d\n", sumArrayFor(arr, n));
    printf("Sum (while): %d\n", sumArrayWhile(arr, n));
    printf("Sum (recursion): %d\n", sumArrayRec(arr, n));

    return 0;
}
```

4. Prompt yêu cầu Cursor AI phân tích project

“Tìm tất cả các hàm `maxValue` trong project này. So sánh sự khác nhau và gợi ý viết 1 hàm chuẩn duy nhất.”

5. Phản hồi của Cursor AI

- Cursor AI quét toàn bộ project
- **Không tìm thấy hàm `maxValue`**
- AI nhận định:
 - Project hiện tại **chỉ có các hàm `sumArray`**
 - Các hàm này khác nhau về **cách cài đặt (`for` / `while` / `recursion`)** nhưng cùng mục đích

👉 Cursor AI gợi ý:

- Nên thống nhất **một hàm chuẩn**

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

- Sử dụng **for loop** vì:
 - Dễ đọc
 - Hiệu năng tốt
 - Phù hợp cho người mới học

6. Code hàm chuẩn do Cursor AI đề xuất

```
// Hàm chuẩn tính tổng các phần tử trong mảng
int sumArray(int arr[], int n) {
    int sum = 0;

    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

    return sum;
}
```

7. Thao tác chỉnh sửa

- Sinh viên chọn **Apply / Keep All**
- Cursor giữ lại code cũ và **đề xuất hàm chuẩn**
- Có thể thay thế toàn bộ các hàm bằng hàm chuẩn nếu cần

8. Chạy lại chương trình sau khi chỉnh sửa

Kết quả in ra:

Sum (for): 15

Sum (while): 15

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

Sum (recursion): 15

Chương trình chạy **đúng, không lỗi**.

9. Ghi nhận những gì AI tạo ra và chỉnh sửa

- Tạo project nhiều file đúng yêu cầu
- Phân tích sự khác nhau giữa các cách cài đặt
- Đề xuất 1 hàm chuẩn, dễ đọc và tối ưu
- Không tự ý thay code khi chưa có xác nhận

10. Kết luận

Cursor AI hỗ trợ tốt trong việc:

- Tổ chức project C nhiều file
- Phân tích code trên toàn bộ project
- Đề xuất chuẩn hóa hàm

Cách làm này giúp sinh viên **hiểu rõ tư duy tổ chức và tối ưu code**, không chỉ viết cho chạy.

BÀI TẬP 8 :

1. Bối cảnh

Sử dụng **project C nhiều file** đã tạo ở **Bài 7**, gồm các hàm xử lý mảng như:

- `sumArray` (for / while / recursion)
- `maxValue` (hàm tìm giá trị lớn nhất trong mảng – do sinh viên bổ sung)

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

2. Yêu cầu đặt ra cho Cursor AI

- Trong **tất cả các hàm xử lý mảng**:
 - Nếu $n \leq 0 \rightarrow$ trả về 0 (với sumArray)
 - Nếu $n \leq 0 \rightarrow$ trả về -1 (với maxArray)
 - Nếu $arr == \text{NULL} \rightarrow$ trả về lỗi tương tự
- Dùng **Project-wide Search** để:
 - Tìm tất cả nơi dùng sumArray, maxArray
 - Chỉnh code để phù hợp với logic mới

3. Prompt sử dụng với Cursor AI

“Trong tất cả các hàm xử lý mảng (sumArray, maxArray), hãy thêm kiểm tra:

– Nếu $n \leq 0$, trả về 0 (hoặc -1 với maxArray)

– Nếu $arr == \text{NULL}$, cũng trả về lỗi

Sau đó tìm tất cả nơi gọi các hàm này trong project và chỉnh code cho phù hợp.”

4. Code trước khi chỉnh sửa (ví dụ)

utils1.c – sumArray

```
int sumArrayFor(int arr[], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }
    return sum;
}
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

utilsMax.c – maxValue

```
int maxValue(int arr[], int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max)
            max = arr[i];
    }
    return max;
}
```

5. Code sau khi Cursor AI chỉnh sửa

sumArray (đã thêm kiểm tra lỗi)

```
int sumArrayFor(int arr[], int n) {
    // Kiểm tra dữ liệu đầu vào không hợp lệ
    if (arr == NULL || n <= 0) {
        return 0;
    }

    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

    return sum;
}
```

maxValue (đã thêm kiểm tra lỗi)

```
int maxValue(int arr[], int n) {
    // Kiểm tra dữ liệu đầu vào không hợp lệ
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
    if (arr == NULL || n <= 0) {
        return -1;
    }

    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    return max;
}
```

6. Project-wide Search & chỉnh nơi gọi hàm

Cursor AI dùng **Project-wide Search** để tìm:

- sumArrayFor(
- sumArrayWhile(
- sumArrayRec(
- maxValue(

main.c sau khi chỉnh

```
#include <stdio.h>
#include "utils.h"

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = 5;

    int sum = sumArrayFor(arr, n);
    int max = maxValue(arr, n);
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
    if (max == -1) {  
        printf("Loi du lieu dau vao\n");  
    } else {  
        printf("Sum = %d\n", sum);  
        printf("Max = %d\n", max);  
    }  
  
    return 0;  
}
```

7. Thao tác trong Cursor

- Sinh viên **xem lại các thay đổi**
- Chọn **Apply / Keep All**
- Cursor thay code mới vào **tất cả các file liên quan**
- Không phát sinh lỗi biên dịch

8. Kết quả chạy chương trình

Trường hợp hợp lệ

Sum = 15

Max = 5

Trường hợp lỗi (n <= 0 hoặc arr == NULL)

Loi du lieu dau vao

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

9. Ghi nhận những gì AI đã tạo ra và chỉnh sửa

- Thêm **kiểm tra an toàn dữ liệu đầu vào** cho tất cả hàm
- Đồng bộ logic xử lý lỗi trong toàn project
- Tự động tìm và chỉnh **tất cả nơi gọi hàm**
- Giúp tránh lỗi runtime (segmentation fault)

10. Kết luận

Cursor AI hỗ trợ hiệu quả trong việc:

- Chuẩn hóa kiểm tra lỗi
- Chỉnh sửa đồng loạt trên nhiều file
- Giúp sinh viên hiểu tư duy **lập trình an toàn**

Cách làm này rất phù hợp cho project C nhiều file.

BÀI TẬP 9 :

1. Mục tiêu

- So sánh hiệu năng giữa các hàm:
 - `maxValue1`
 - `maxValue2`
 - `maxValue3`
- Benchmark với mảng **1.000.000 phần tử**
- Đo thời gian bằng `clock()` trong `time.h`
- Sử dụng Cursor AI để:
 - Gom tất cả hàm `maxValue`
 - Tạo file `benchmark.c`
 - Chạy và ghi nhận kết quả

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

2. Prompt sử dụng với Cursor AI

“Dùng Project-wide Search để tìm tất cả các hàm maxValue trong project.

Tạo một file benchmark.c để so sánh hiệu năng giữa maxValue1, maxValue2, maxValue3 với mảng 1,000,000 phần tử.

Đo thời gian bằng clock() trong time.h.”

3. Kết quả Project-wide Search

Cursor AI tìm thấy:

- maxValue1 (for loop)
- maxValue2 (while loop)
- maxValue3 (đệ quy)

Các hàm đều có cùng chức năng nhưng khác cách cài đặt.

4. File benchmark.c do Cursor AI tạo

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "utils.h"

#define SIZE 1000000

int main() {
    int *arr = (int *)malloc(SIZE * sizeof(int));

    // Khởi tạo mảng với giá trị ngẫu nhiên
    for (int i = 0; i < SIZE; i++) {
        arr[i] = rand();
    }
}
```

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

```
clock_t start, end;
double time_used;

// Benchmark maxValue1
start = clock();
int max1 = maxValue1(arr, SIZE);
end = clock();
time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
printf("maxValue1: %f seconds (max = %d)\n", time_used, max1);

// Benchmark maxValue2
start = clock();
int max2 = maxValue2(arr, SIZE);
end = clock();
time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
printf("maxValue2: %f seconds (max = %d)\n", time_used, max2);

// Benchmark maxValue3
start = clock();
int max3 = maxValue3(arr, SIZE);
end = clock();
time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
printf("maxValue3: %f seconds (max = %d)\n", time_used, max3);

free(arr);
return 0;
}
```

5. Chỉnh sửa bổ sung của Cursor AI

- Thêm #define SIZE 1000000
- Dùng malloc để tránh tràn stack
- Thêm free(arr) để tránh rò rỉ bộ nhớ
- Thêm in kết quả max để tránh compiler tối ưu bỏ hàm

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

6. Thao tác của sinh viên

- Kiểm tra file `benchmark.c`
- Chọn **Apply / Keep All**
- Build lại toàn bộ project
- Chạy chương trình benchmark

7. Kết quả chạy (ví dụ)

maxValue1: 0.004231 seconds (max = 2147483123)

maxValue2: 0.004198 seconds (max = 2147483123)

maxValue3: 0.012847 seconds (max = 2147483123)

8. Nhận xét kết quả

- maxValue1 (for loop) và maxValue2 (while loop):
 - Thời gian gần tương đương
 - Hiệu năng tốt
- maxValue3 (đệ quy):
 - Chậm hơn rõ rệt
 - Không phù hợp cho mảng lớn

9. Ghi nhận những gì AI tạo ra và chỉnh sửa

- Tự động gom các hàm `maxValue`
- Tạo file benchmark độc lập
- Viết code đo thời gian chính xác
- Gợi ý tránh lỗi bộ nhớ và tối ưu compiler

Họ và tên : Lý Gia Huy

BTVN : session 7

Lớp : CNTT5

10. Kết luận

Cursor AI hỗ trợ tốt trong việc:

- Phân tích project lớn
- Viết benchmark đo hiệu năng
- Giúp sinh viên hiểu rõ sự khác biệt giữa các cách cài đặt

Đây là cách tiếp cận hiệu quả khi học **tối ưu hóa chương trình C**.