**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**BÀI TẬP LỚN**

**MÔN: KỸ THUẬT LẬP TRÌNH**

Giảng viên hướng dẫn: ThS. TRẦN PHONG NHÃ

Sinh viên thực hiện: HUỲNH NGUYÊN BỔN   
Lớp : CQ.65.CNTT

Khoá : 65

MSSV : 6551071009

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**BÀI TẬP LỚN**

**MÔN: KỸ THUẬT LẬP TRÌNH**

Giảng viên hướng dẫn: ThS. TRẦN PHONG NHÃ

Sinh viên thực hiện: HUỲNH NGUYÊN BỔN Lớp : CQ.65.CNTT

Khoá : 65

MSSV : 6551071009

# LỜI CẢM ƠN

Lời nói đầu tiên,em xin gửi tới Quý Thầy Cô Bộ Môn Công Nghệ Thông Tin Trường Đại Học Phân Hiệu Giao Thông Vận Tải tại Thành Phố Hồ Chí Minh lời chúc sức khỏe và lời cảm ơn sâu sắc nhất.

Trước tiên, em xin cảm ơn nhà trường đã tạo điều kiện cơ sở vật chất tốt nhất để em có thể hoàn thành tốt bài báo cáo này,.

Đặc biệt, em xin gửi lời cảm ơn chân thành Thầy Trần Phong Nhã đã nhiệt tình hướng dẫn ,hỗ trợ và truyền đạt những kiến thức quý báo nhất trong quá trình thực hiện bài báo cáo này.Điều đó đã giúp cho em có thêm nhiều kiến thức bổ ích,trau dồi và rèn luyện thêm nhiều kĩ năng quan trọng.

Do những giới hạn về kiến thức,đồng thời bản thân em cũng thiếu kinh nghiệm trong việc thực hiện bao cáo vì vậy khó có thể tránh khỏi những thiếu sót .Em kính mong sự chỉ dẫn ,nhận xét và đóng góp đến từ Thầy để bài báo cáo này có thể hoàn thiện hơn nữa.

Lời cuối cùng,em xin kính chúc Thầy Cô Bộ Môn Công Nghệ Thông Tin Trường Đại Học Phân Hiệu Giao Thông Vận Tải tại Thành Phố Hồ Chí Minh và đặc biệt là thầy Trần Phong Nhã sẽ luôn có thật nhiều sức khỏe,hạnh phúc và thành công hơn nữa.

Em xin chân thành cảm ơn!.

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

**Giáo viên hướng dẫn**

**Trần Phong Nh**

Mục lục

[LỜI CẢM ƠN 1](#_Toc198587336)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc198587337)

[A. Lý thuyết 4](#_Toc198587338)

[1. Hàm: 4](#_Toc198587339)

[2. Con trỏ 5](#_Toc198587340)

[a. Con trỏ và địa chỉ: 5](#_Toc198587341)

[b. Tham chiếu và giải tham chiếu: 5](#_Toc198587342)

[3. Con trỏ mảng 7](#_Toc198587343)

[4. Con trỏ hàm: 7](#_Toc198587344)

[5. Cấp phát động 9](#_Toc198587345)

[a. Hàm malloc(): 9](#_Toc198587346)

[b. Hàm calloc(): 10](#_Toc198587347)

[c. Hàm free(): 11](#_Toc198587348)

[6. Xử lý tệp 12](#_Toc198587349)

[a. Tạo file 12](#_Toc198587350)

[b. Ghi nội dung vào file 12](#_Toc198587351)

[c. Đọc nội dung từ file 13](#_Toc198587352)

[7. Kiểu cấu trúc 14](#_Toc198587353)

[8. Danh sách liên kết (Danh sách liên kết đơn) 15](#_Toc198587354)

[B. Ứng dụng 16](#_Toc198587355)

# A. Lý thuyết

## 1. Hàm:

Bạn đã từng sử dụng các hàm như sqrt, pow, abs... của các thư viện có sẵn trong C, tuy nhiên để giải quyết bài toán của bạn thì bạn cần phải tự xây dựng các hàm để giải quyết các chức năng nhỏ cho bài toán của mình.

Hàm (function) là một các khối lệnh có nhiệm vụ thực hiện một chức năng nào đó.

Ở các bài học trước bạn đều viết mã nguồn C bên trong hàm main, chương trình sẽ trở nên khó quản lý khi số lượng dòng code trở nên lớn, cùng điểm qua các lợi ích của việc sử dụng hàm

Ví dụ 1 : Hàm có kiểu trả về là int, có 3 tham số là a, b, c

int tong(int a, int b, int c){

int sum = a + b + c;

return sum;

}

Ví dụ 2 : Hàm có kiểu trả về là void, có 3 tham số là a kiểu int, b kiểu long long, c kiểu double

void display(int a, long long b, double c){

printf("%d %lld %.2lf\n", a, b, c);

}

## 2. Con trỏ

### a. Con trỏ và địa chỉ:

Con trỏ hay biến con trỏ cũng là một biến thông thường nhưng giá trị mà nó lưu lại là địa chỉ của 1 biến khác.

Ví dụ biến kiểu int N trong chương trình sẽ có địa chỉ nhất định trong bộ nhớ, để lưu trữ giá trị địa chỉ này ta cần biến con trỏ kiểu int

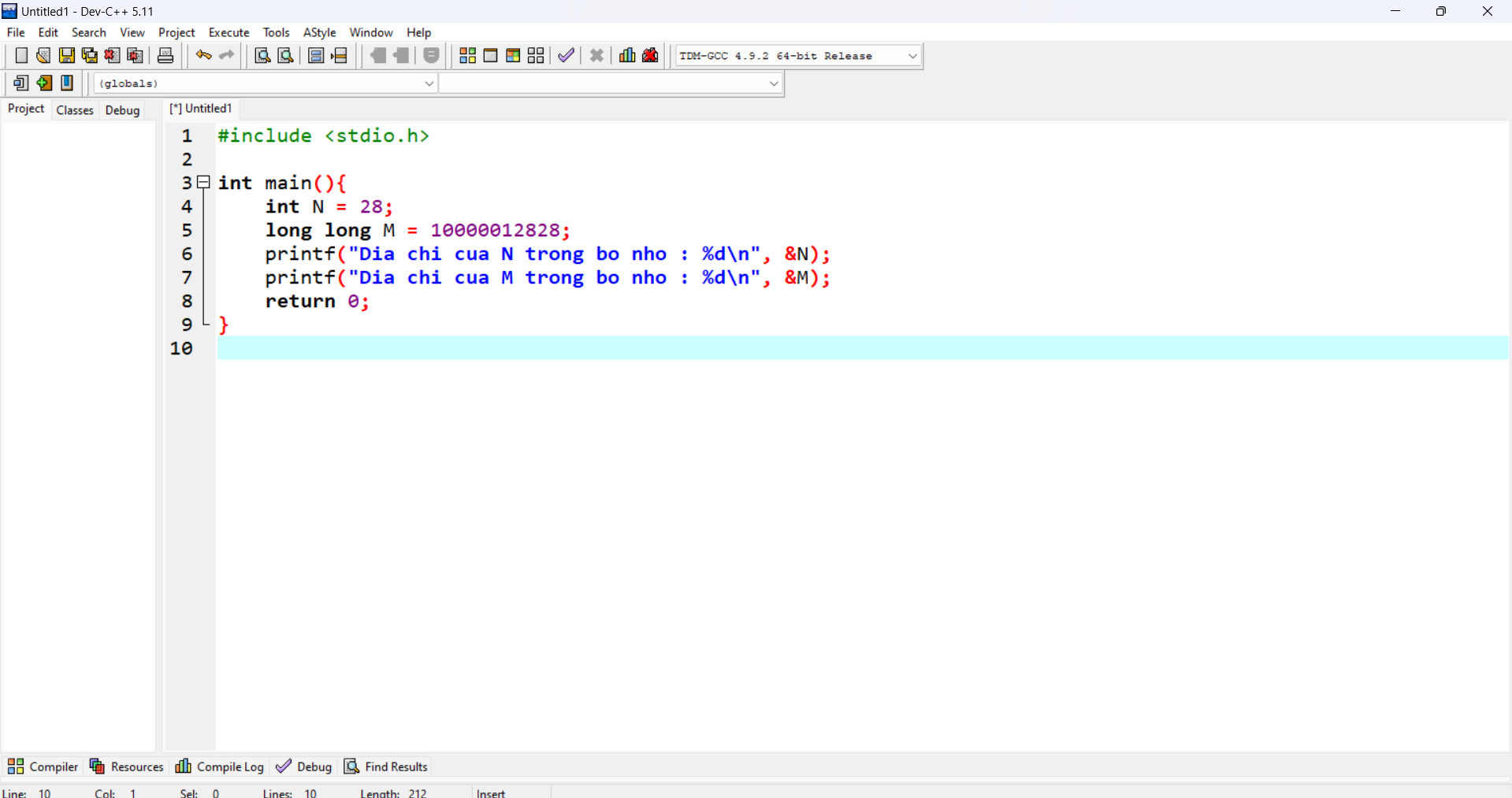
Khi khai báo biến con trỏ ta thêm dấu \* vào trước tên biến.

Cú pháp khai báo : Kiểu\_Dữ\_Liệu \*Tên\_Biến\_Con\_Trỏ;

Mỗi biến trong chương trình đều được cấp phát vùng nhớ để lưu trữ giá trị của nó, ví dụ biến int sẽ được cấp phát 4 byte liên tiếp để lưu trữ và lấy địa chỉ của byte đầu tiên làm địa chỉ cho biến.

Để in ra địa chỉ của biến bạn dùng toán tử &

Ví dụ: &N sẽ cho bạn địa chỉ của biến N



Chú ý : Khi bạn chạy code trên thì địa chỉ của N và M sẽ khác, và mỗi lần bạn chạy có thể sẽ có một địa chỉ khác nhau.

Con trỏ được sinh ra để lưu địa chỉ của biến ở kiểu dữ liệu tương ứng với nó, ví dụ biến con trỏ kiểu int sẽ lưu được địa chỉ của biến int.

Chương trình sau sẽ gán địa chỉ của N cho biến con trỏ ptr, khi đó ta nói con trỏ ptr trỏ tới biến N

### b. Tham chiếu và giải tham chiếu:

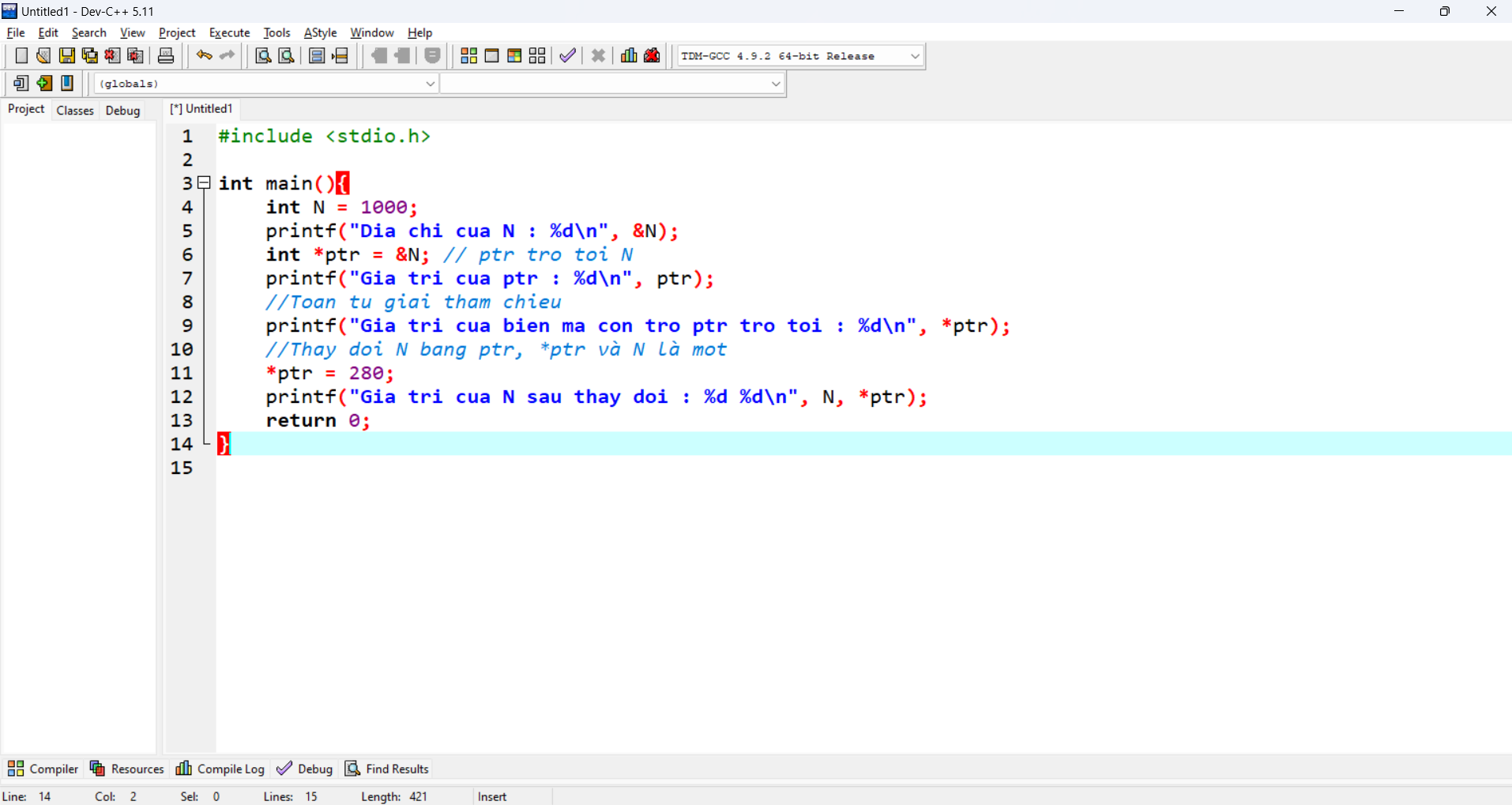
Khi con trỏ ptr trỏ tới hay tham chiếu (reference) tới biến N thì thông qua con trỏ ptr ta có thể truy xuất, thay đổi giá trị của biến N mà không cần dùng N.

Để truy xuất tới giá trị của biến mà con trỏ đang trỏ tới ta dùng toán tử giải tham chiếu \* (dereference)

Sau khi con trỏ ptr trỏ tới biến N thì N và \*ptr là một, đều truy xuất đến ô nhớ mà N đang chiếm để lấy giá trị tại ô nhớ đó.

Lưu ý là bạn cần phân biệt dấu \* khi khai báo con trỏ ptr và dấu \* khi giải tham chiếu con trỏ ptr. Dấu \* khi khai báo thể hiện ptr là một con trỏ còn dấu \* trước ptr ở những câu lệnh sau là toán tử giải tham chiếu.

Ví dụ:

Một biến có thể được trỏ tới bởi nhiều con trỏ, khi đó bạn có thể thông qua bất cứ 1 con trỏ nào để thay đổi giá trị của biến mà nó đang trỏ tới.

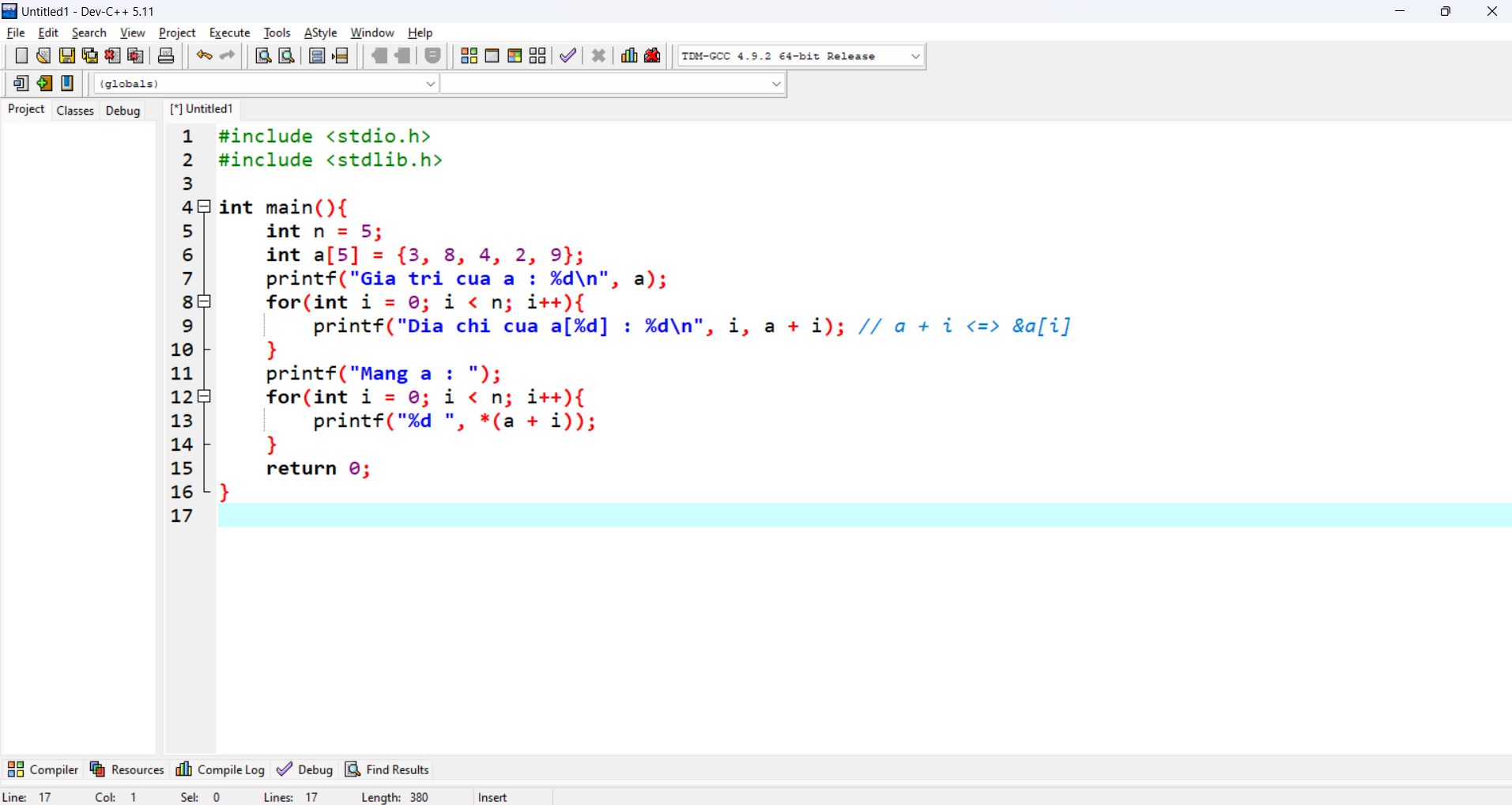
## 3. Con trỏ mảng

Giá trị của mảng chính là địa chỉ phần tử đầu tiên trong mảng, ví dụ mảng A[] = {3, 8, 4, 2, 9} thì A[] có giá trị là địa chỉ của A[0]

Tên mảng chính là một hằng con trỏ và bạn không thể thay đổi được

Mảng 1 chiều A[] thì con trỏ trỏ tới phần tử A[i] là A + i, vậy (A + i) tương đương với &A[i] hay địa chỉ của A[i] và \*(A + i) tương đương với A[i].

Ví dụ 1: Sử dụng mảng thông qua con trỏ



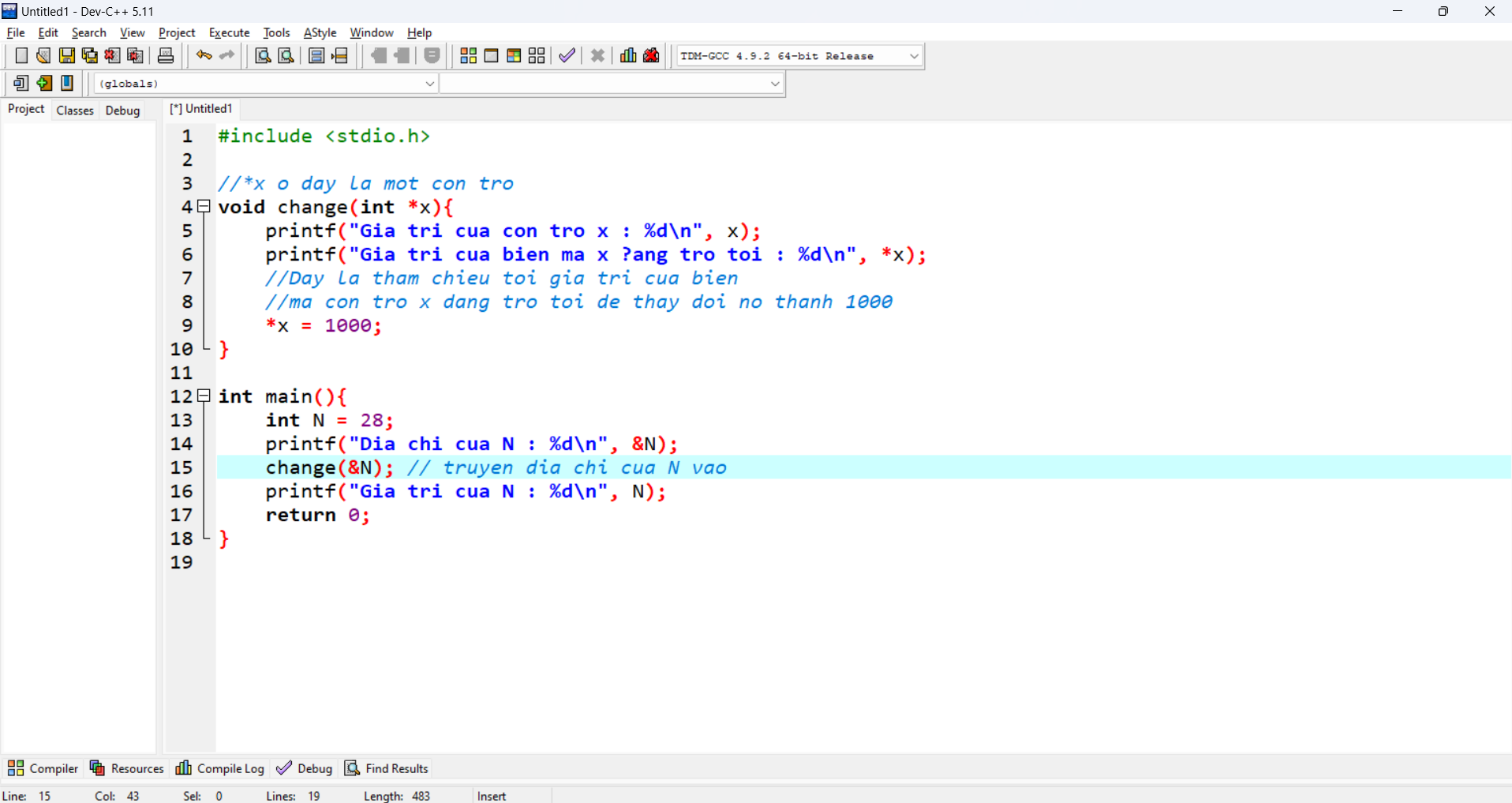
## 4. Con trỏ hàm:

Con trỏ làm tham số cho hàm:

Để thay đổi giá trị của 1 biến sau khi hàm kết thúc thì việc truyền giá trị là không hợp lý, thay vì đó bạn hãy sử dụng con trỏ với mục đích là thay đổi giá trị của biến thông qua con trỏ.

Khi hàm có tham số là một con trỏ thì khi gọi hàm bạn cần truyền vào một giá trị phù hợp, có thể là địa chỉ của 1 biến hoặc một con trỏ khác.

Ví dụ 1: Thay đổi giá trị của biến sau khi hàm kết thúc

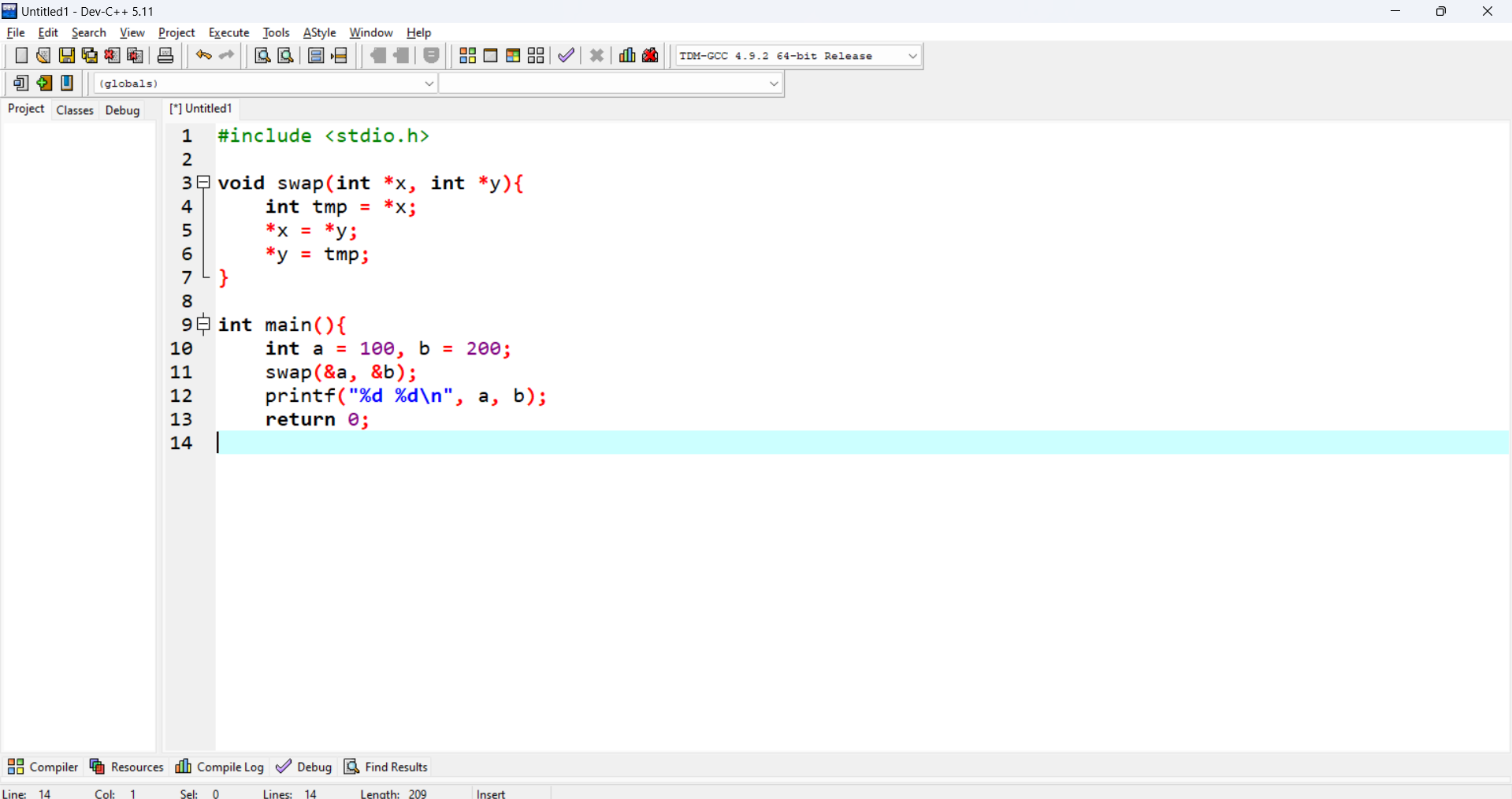
Giải thích:

Hàm change có tham số là một con trỏ kiểu int có tên là x, trong main bạn gọi change và truyền địa chỉ của N vào.

Khi đó x sẽ được gán giá trị là địa chỉ của N, trong hàm change thì câu lệnh \*x = 1000 sẽ truy xuất tới ô nhớ mà x đang trỏ tới và gán giá trị 1000. Mà x lại đang tham chiếu tới N nên giá trị của N đã bị thay đổi.

Sau khi hàm change kết thúc thì giá trị của N đã bị thay đổi thực sự.

Ví dụ 2 : Hoán đổi giá trị của 2 biến



Ví dụ 3 : Hàm trả về con trỏ

## 5. Cấp phát động

Cấp phát động là một kỹ thuật giúp bạn có thể xin cấp phát một vùng nhớ phù hợp với nhu cầu của bài toán trong lúc thực thi thay vì phải khai báo cố định.

Cấp phát động thường được sử dụng để cấp phát mảng động hoặc sử dụng trong các cấu trúc dữ liệu.

Có nhiều cấu trúc dữ liệu quan trọng dựa trên kỹ thuật này như : Danh sách liên kết, cây nhị phân hay các cấu trúc dữ liệu dạng mảng động.

Trong ngôn ngữ lập trình C cung cấp cho bạn 4 hàm trong thư viện để bạn có thể thao tác với việc cấp phát động vùng nhớ và giải phóng vùng nhớ sau khi sử dụng, bao gồm:

* **malloc()**
* **calloc()**
* **free()**
* **realloc()**

### a. Hàm malloc():

Hàm malloc() viết tắt của từ memory allocation tức là cấp phát động vùng nhớ, hàm này được sử dụng để xin cấp phát khối bộ nhớ theo kích thước byte mong muốn.

Giá trị trả về của hàm là một con trỏ kiểu void, nên ép kiểu sang kiểu dữ liệu mà bạn cần dùng.

Các giá trị trong các ô nhớ được cấp phát là giá trị rác

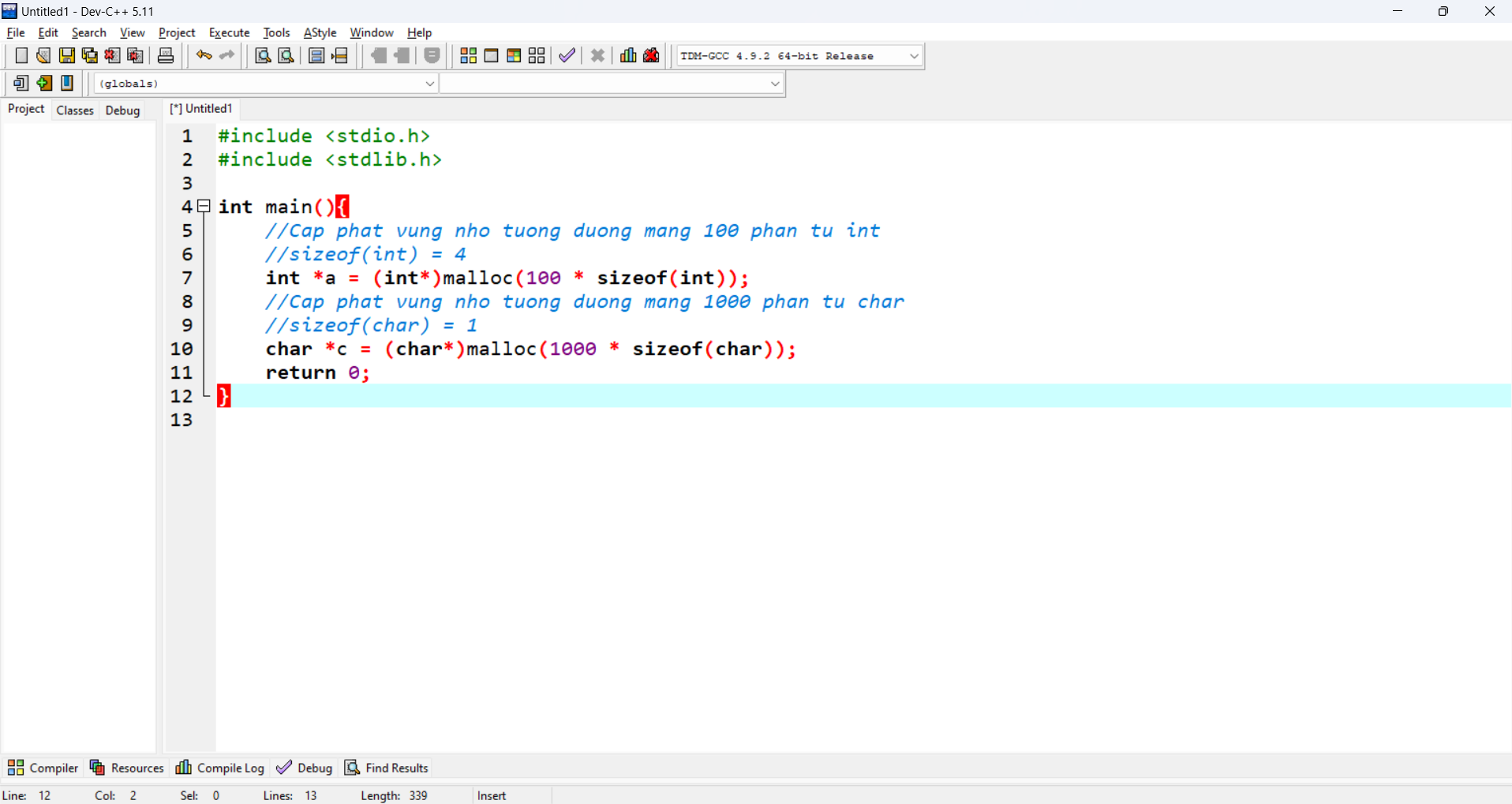
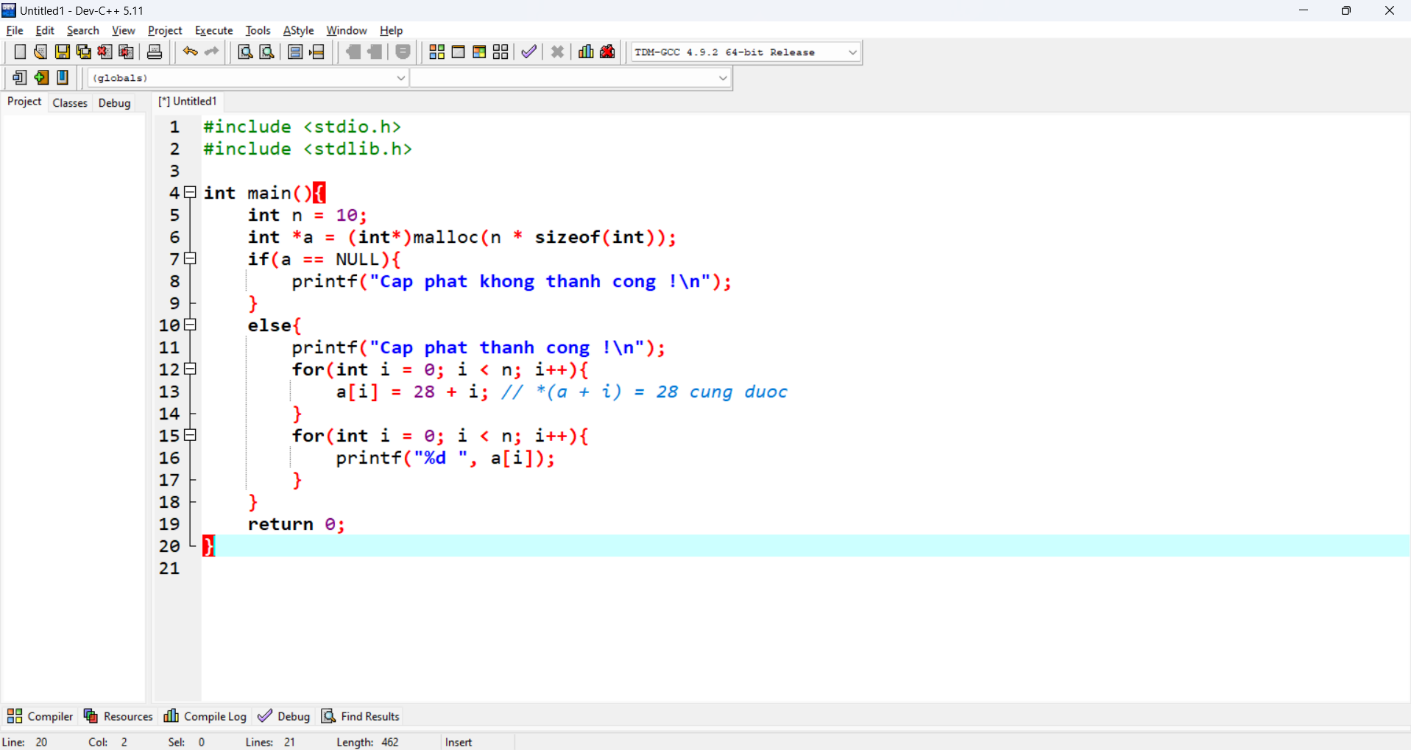
Cú pháp: ptr = (cast\_type\*)malloc(byte\_size)

Trong đó :

* ptr là con trỏ lưu trữ ô nhớ đầu tiên của vùng nhớ được cấp phát
* cast\_type\* là kiểu con trỏ mà bạn muốn ép kiểu sang
* byte\_size là kích thước theo byte bạn muốn cấp phát

Trong trường hợp không cấp phát đủ vùng nhớ thì hàm malloc sẽ trả về con trỏ NULL.

Ví dụ 1:

Ví dụ 2:

### b. Hàm calloc():

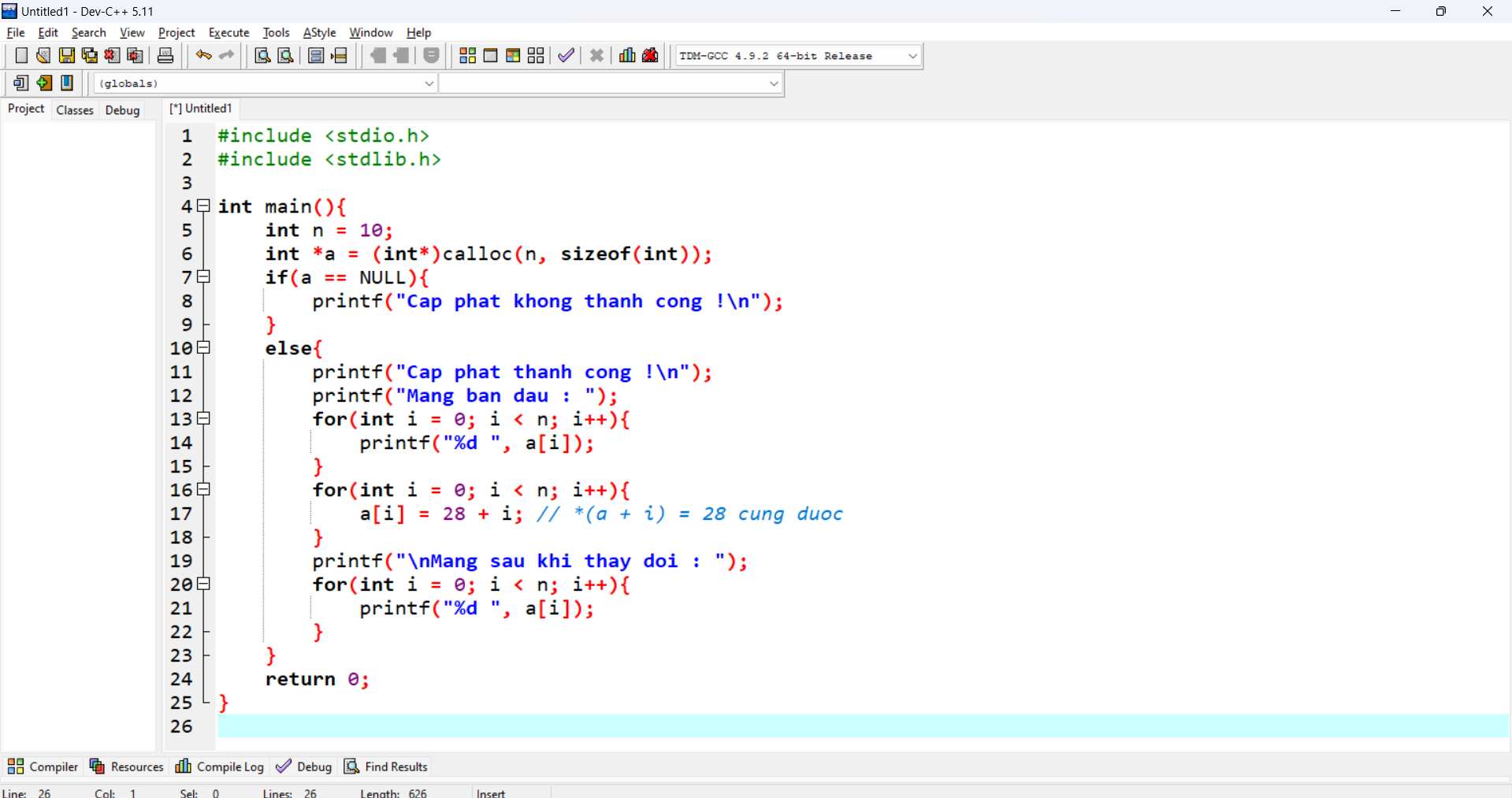
Hàm calloc() viết tắt của contiguous allocation tương tự như malloc() sử dụng để cấp phát vùng nhớ động nhưng các giá trị của các vùng nhớ được cấp phát sẽ có giá trị mặc định là 0 thay vì giá trị rác như hàm malloc().

Cú pháp: ptr = (cast\_type\*) calloc(n, element\_size)

Trong đó :

* ptr là con trỏ lưu trữ ô nhớ đầu tiên của vùng nhớ được cấp phát
* cast\_type\* là kiểu con trỏ mà bạn muốn ép kiểu sang
* n là số lượng phần tử bạn muốn cấp phát
* element\_size là kích thước theo byte của 1 phần tử

Ví dụ:



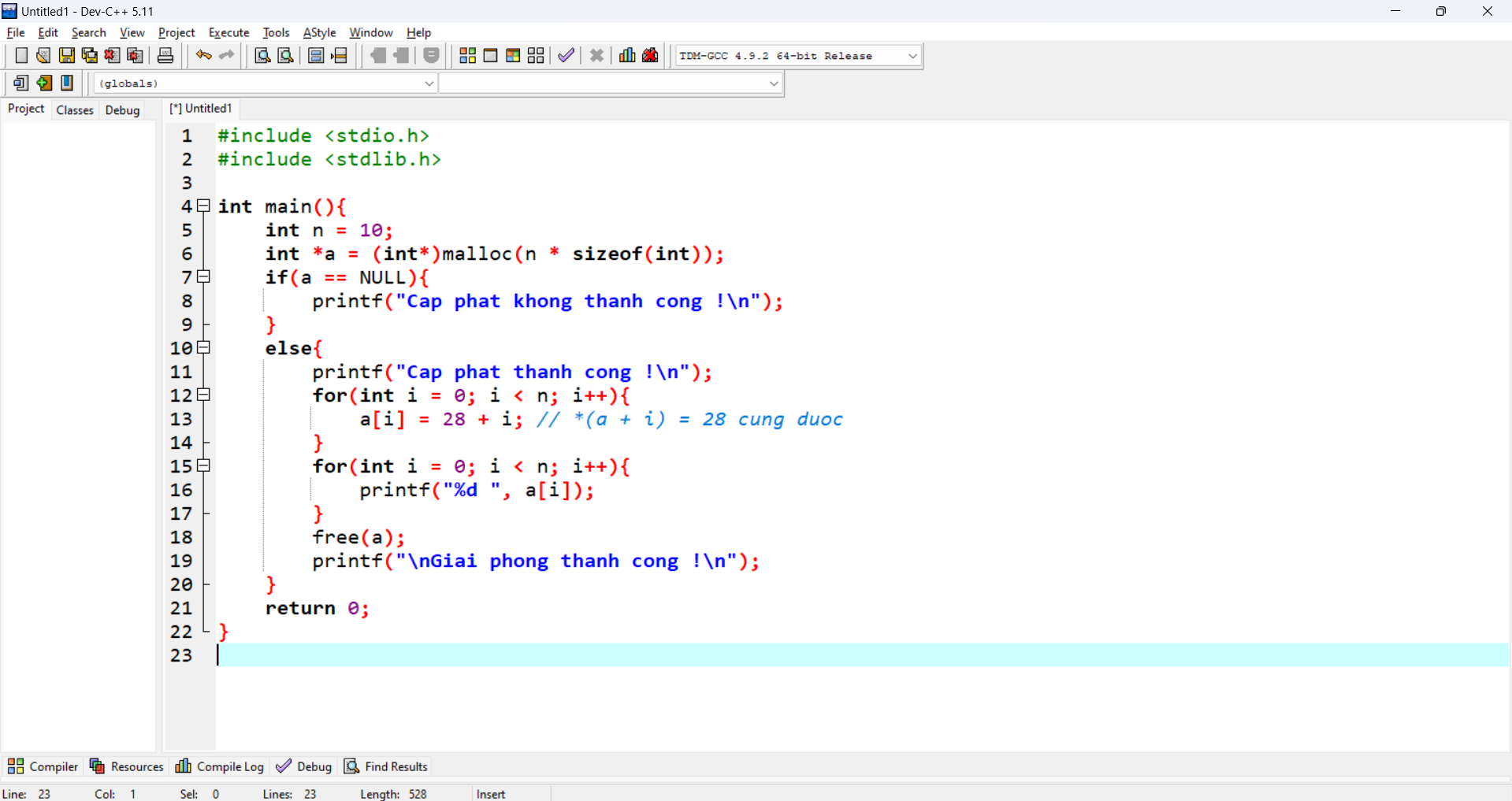
### c. Hàm free():

Hàm malloc() và calloc() xin cấp phát vùng nhớ nhưng lại không tự giải phóng vùng nhớ mà nó xin cấp phát, hàm free() có chắc năng giải phóng vùng nhớ mà malloc() hoặc calloc() đã xin cấp phát.

Việc sử dụng free() sau khi sử dụng malloc() và calloc() là cần thiết để tránh lãng phí bộ nhớ

Cú pháp: free(ptr)

Ví dụ:

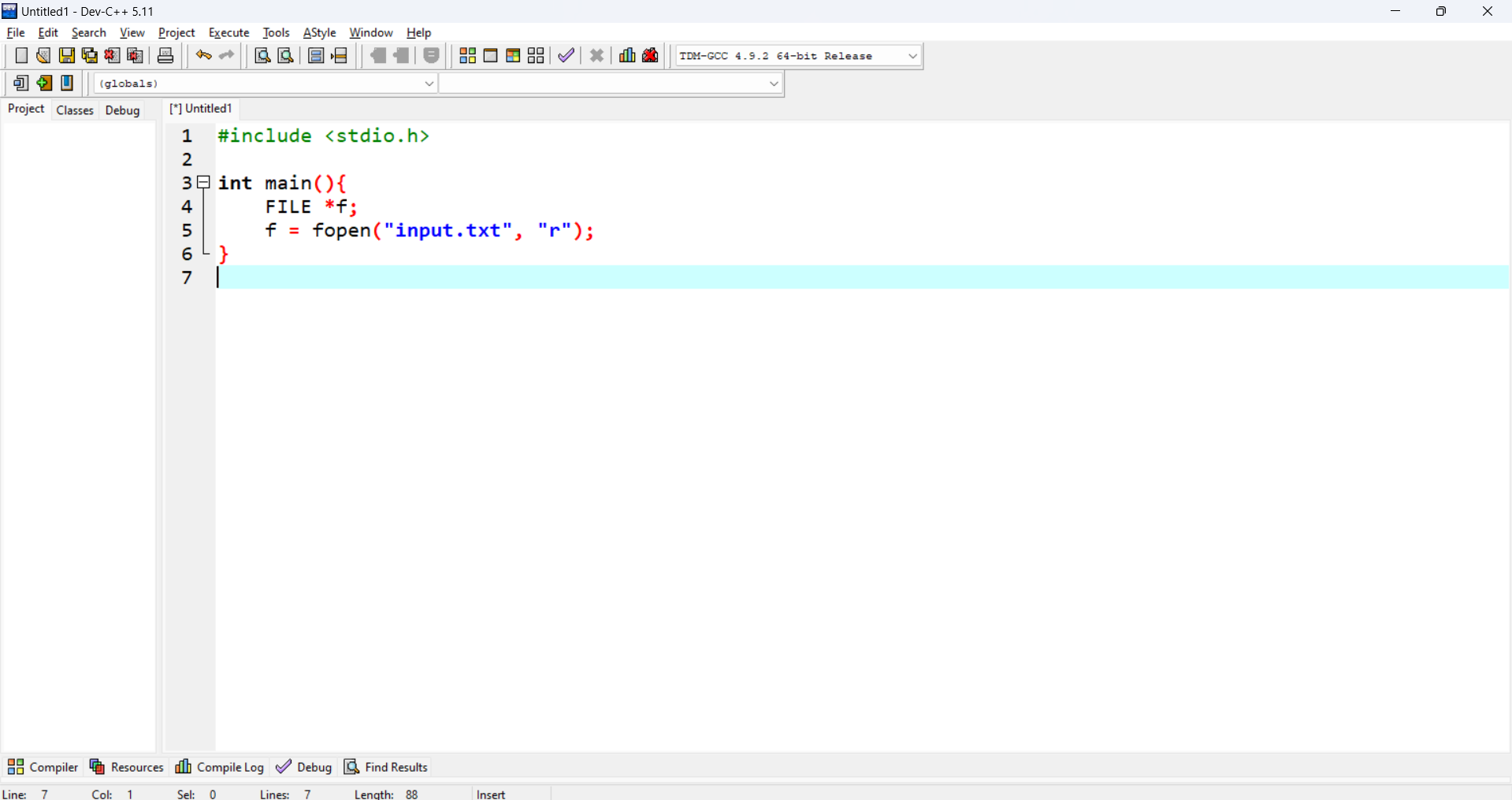


## 6. Xử lý tệp

### a. Tạo file

Để làm việc với file thì đầu tiên ta cần mở file đó lên, trong C để chương trình có thể làm việc với file ta cần thông qua 1 con trỏ kiểu FILE

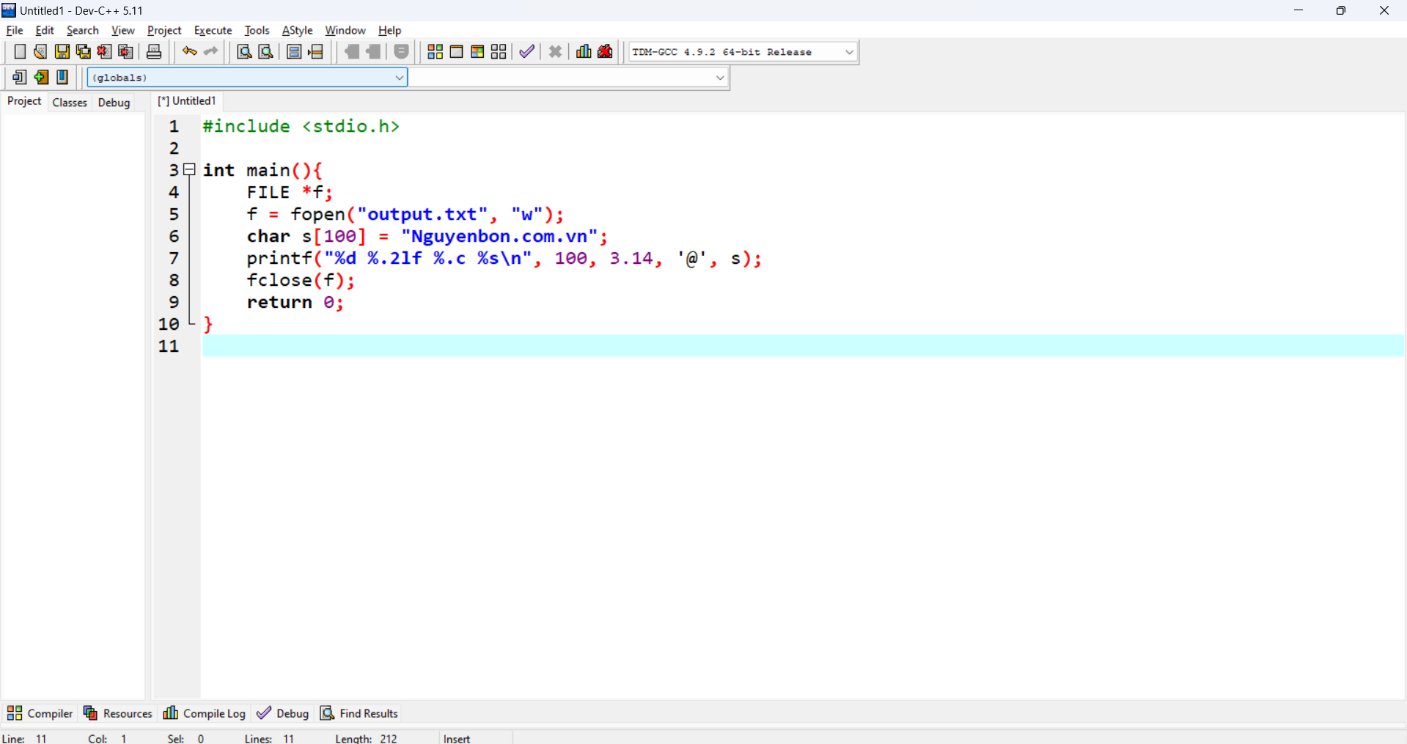
Ví dụ:



### b. Ghi nội dung vào file

Để ghi nội dung vào file ta làm tương tự như in nội dung ra màn hình, các hàm ghi nội dung vào file thường có thêm chữ f ở trước.

Để ghi nội dung vào file thì ta chỉ cần dùng hàm fprintf().

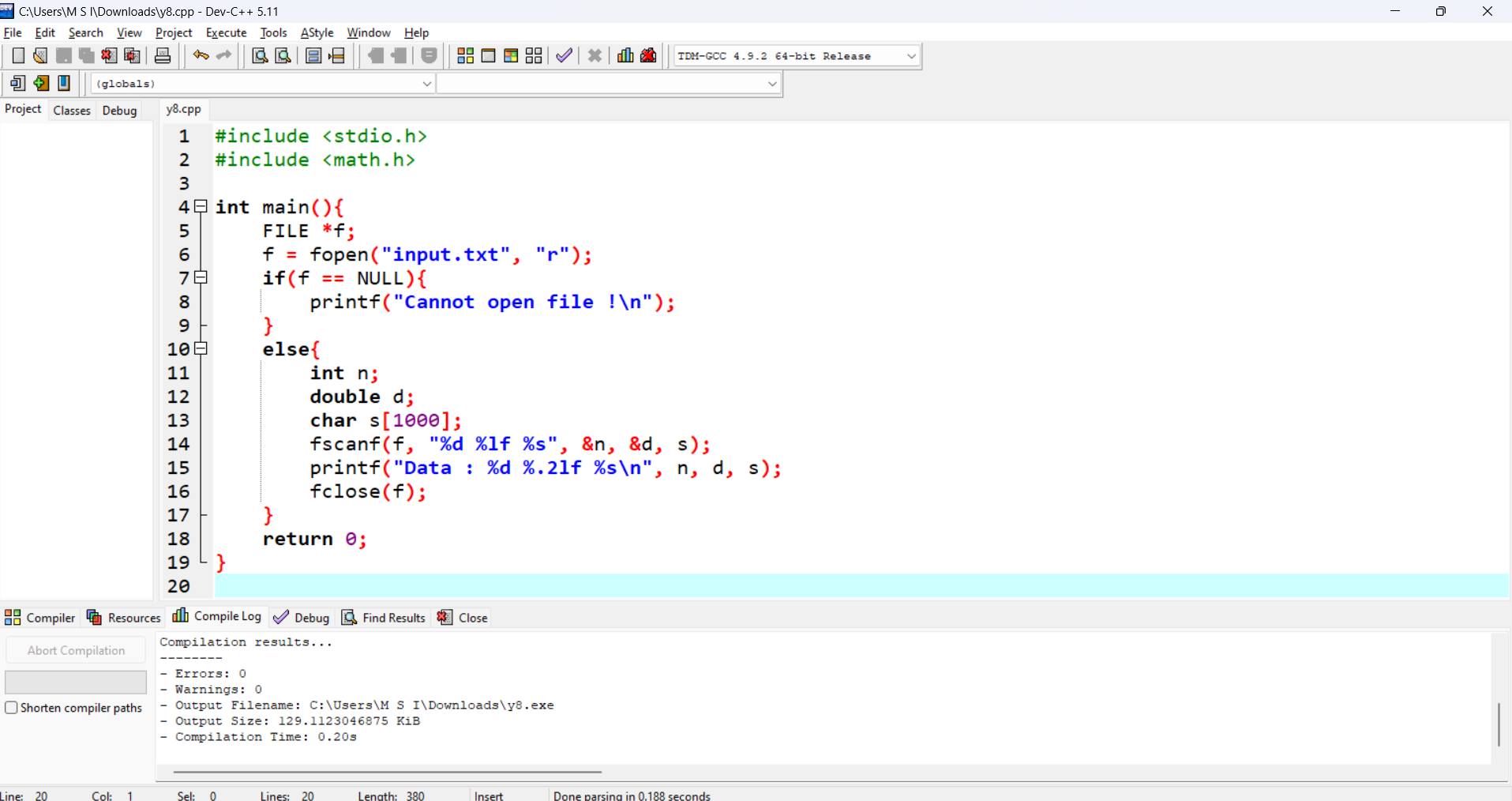
 Ví dụ: Ghi 1 số nguyên, 1 số thực, 1 kí tự và 1 chuỗi ký tự vào trong file output.txt

### c. Đọc nội dung từ file

Để đọc nội dung từ file ta sử dụng 2 hàm chính là fscanf() và fgets(), ngoài ra còn có fgetchar()

* **Hàm fscanf()**

Ví dụ : Đọc số nguyên int, số thực double, chuỗi ký tự không có dấu cách trong file input.txt, sau đó in ra màn hình

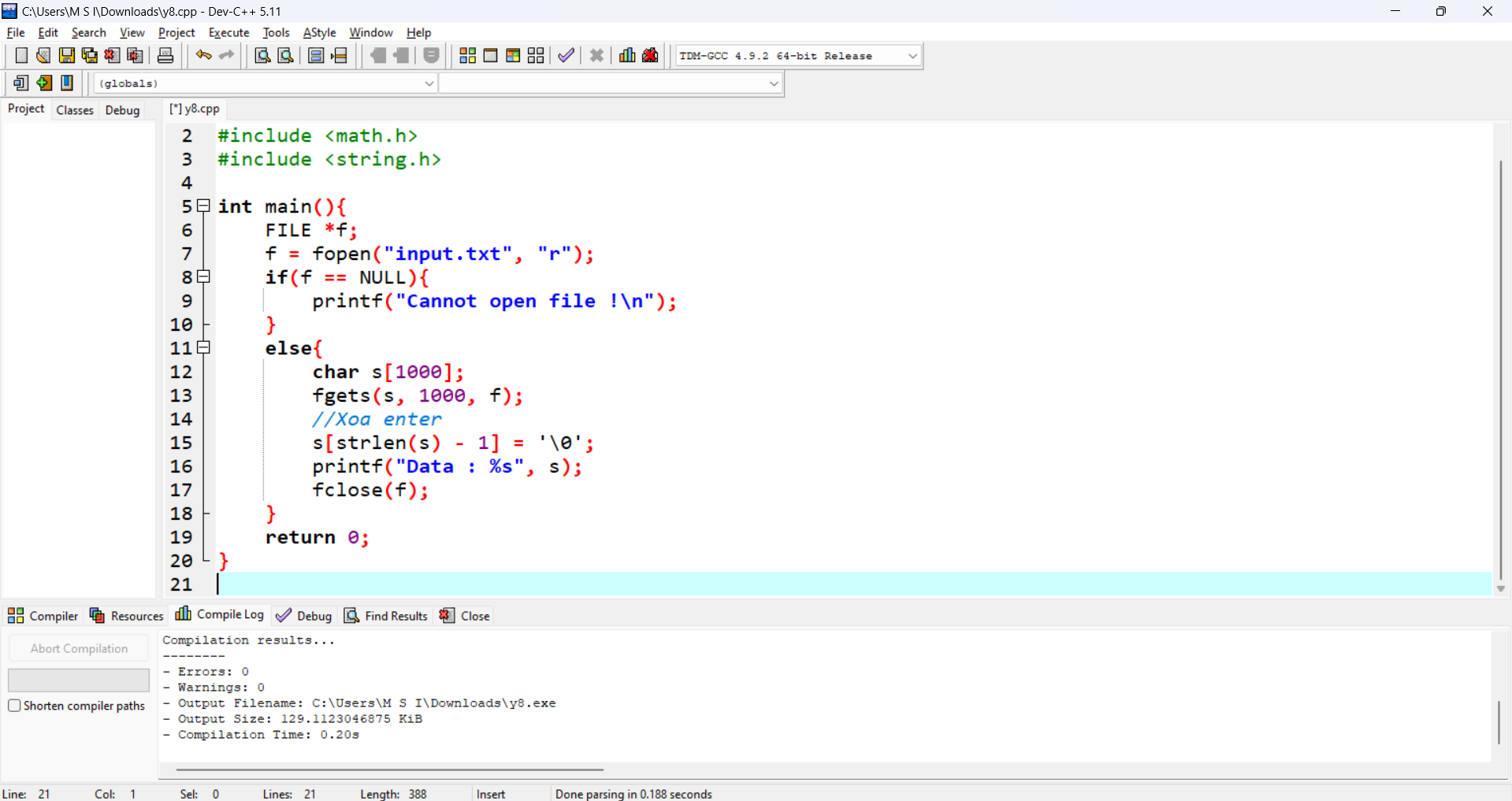


* **Hàm fgets()**

Hàm fgets() được sử dụng tương tự hàm gets(), dùng để đọc chuỗi ký tự có dấu cách.

Hàm này cũng bị trôi lệnh như hàm gets nên ta cần lưu ý xử lý, ngoài ra nó còn đọc cả ký tự enter ở cuối dòng nên ta cần loại bỏ ký tự enter này khỏi chuỗi

Ví dụ:



## 7. Kiểu cấu trúc

Struct là giải pháp khi ta cần giải quyết các bài toán thực tế khi mà đối tượng mình cần lưu lại trong chương trình cần rất nhiều thông tin.

Ví dụ khi ta muốn lưu thông tin của một sinh viên thì không có kiểu dữ liệu nào trong C phù hợp cả, mà ta cần tập hợp nhiều kiểu dữ liệu vào làm một.

Struct hay cấu trúc là một kiểu dữ liệu mà tự ta định nghĩa ra bằng cách gộp nhiều kiểu dữ liệu có sẵn lại nhằm mục đích có thể mô tả nhiều trường thông tin của đối tượng ta cần lưu.

Cú pháp khai báo :

struct ten\_struct{

data\_type1 data\_field1;

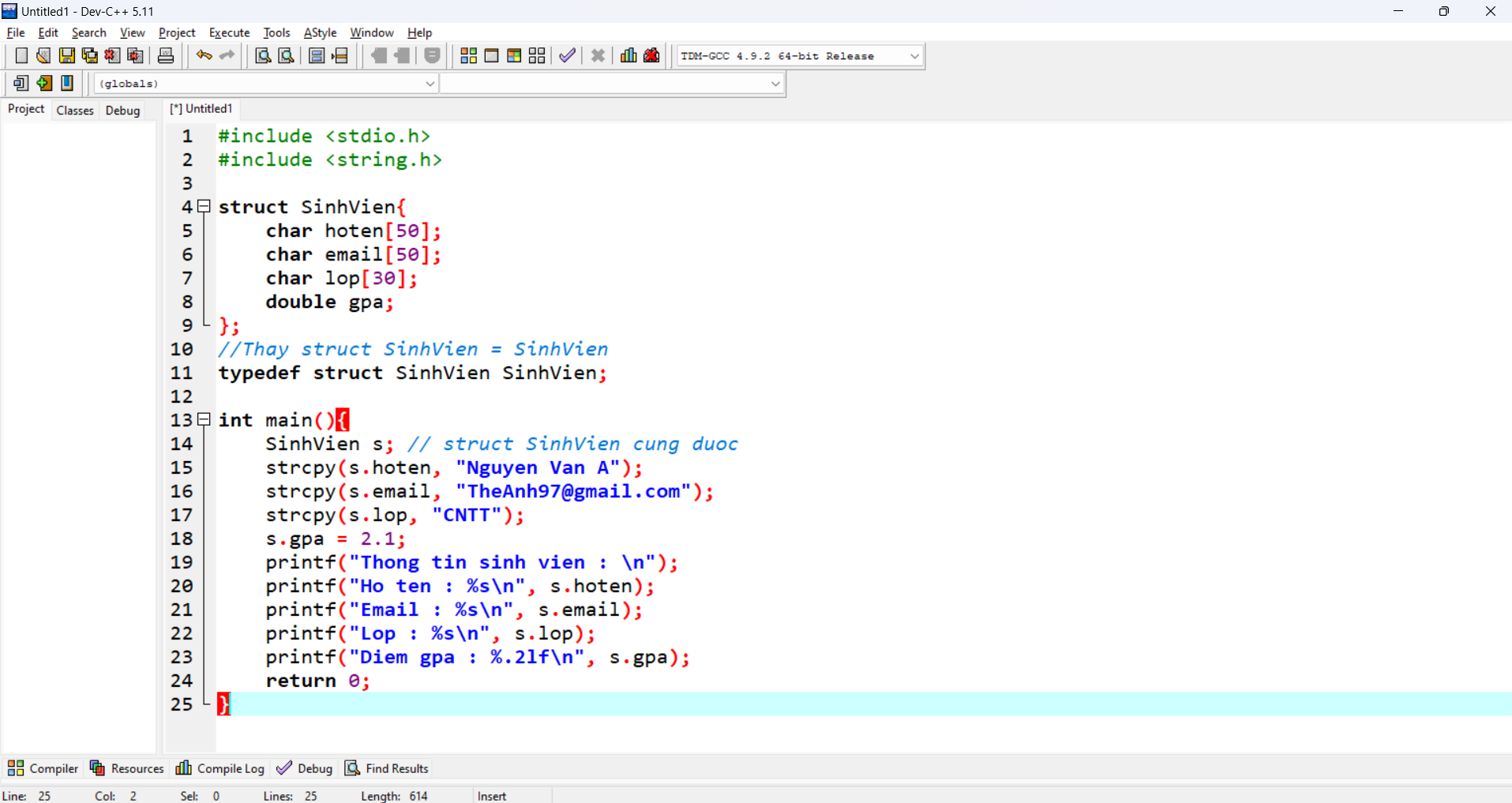
data\_type2 data\_field2;

....

data\_typen data\_fieldn;

};

Trong struct bạn sẽ liệt kê các trường thông tin mà bạn cần lưu cho cấu trúc bạn đang xây dựng kèm theo kiểu dữ liệu của trường thông tin đó.

Ví dụ: 

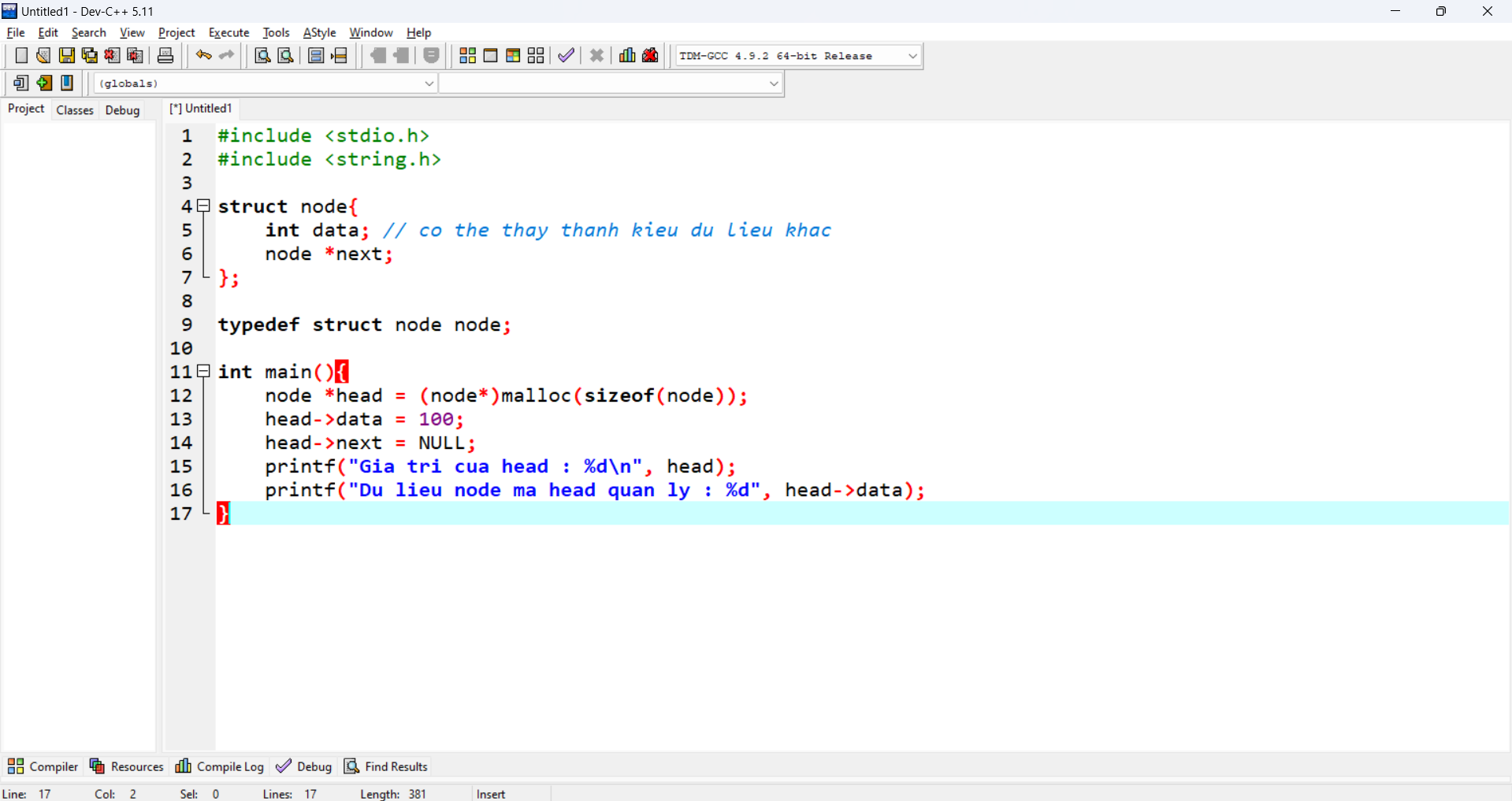
## 8. Danh sách liên kết (Danh sách liên kết đơn)

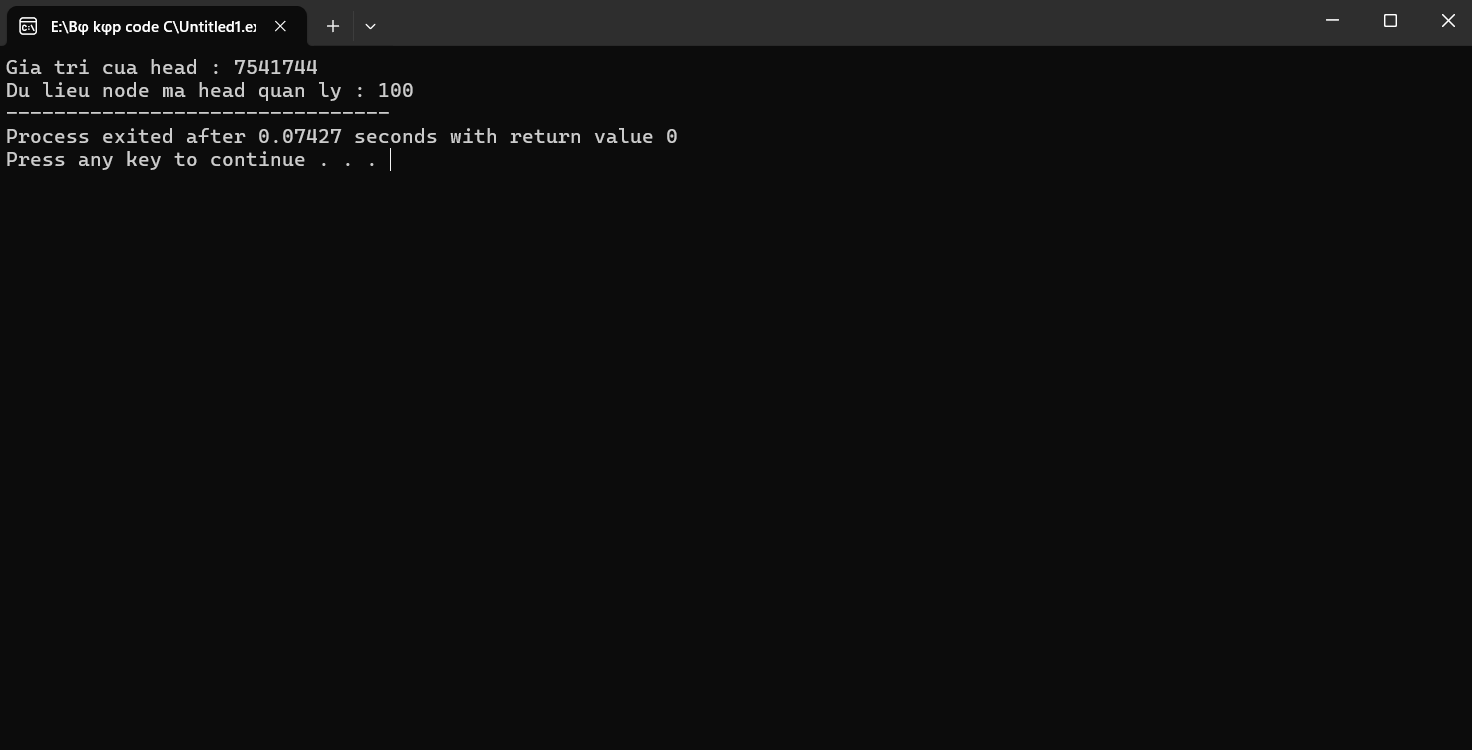
**Danh sách liên kết đơn**- Singly linked list là một cấu trúc dữ liệu, nó tương tự như một mảng động với những tính chất quan trọng như sau :

* Mở rộng và thu hẹp một cách linh hoạt
* Các phần tử trong DSLK gọi là node và được cấp phát động khi cần
* Số lượng phần tử trong DSLK phụ thuộc vào bộ nhớ heap
* Dễ dàng chèn và xóa phần tử
* Các phần tử trong DSLK không có thứ tự
* Truy cập phần tử trong DSLK cần truy cập tuần tự không thể truy cập qua chỉ số
* Mỗi node trong DSLK cần có thêm 1 con trỏ để lưu liên kết

Mỗi phần tử trong DSLK được gọi là một node hay nút, node sẽ lưu thông tin dữ liệu (ví dụ như một số nguyên, 1 chuỗi ký tự, 1 sinh viên...) và ngoài ra cần có phần liên kết, phần liên kết này giúp các node có thể liên lạc với nhau. Mỗi node sẽ lưu thêm địa chỉ của node phía sau nó trong DSLK thông qua 1 thuộc tính con trỏ.

Node cuối cùng trong danh sách liên kết thì phần liên kết của nó sẽ lưu con trỏ NULL

Ví dụ: 



# B. Ứng dụng

**QUẢN LÝ QUÁN CAFÉ**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Cau truc mon uong

typedef struct {

int id;

char ten[50];

float gia;

} Mon;

// Cau truc don hang

typedef struct Don {

Mon mon;

int sl;

struct Don\* next;

} Don;

// Hàm nhap thuc don

Mon\* nhapMenu(int\* n) {

printf("Nhap so mon: ");

scanf("%d", n);

getchar(); // xóa \n

Mon\* ds = (Mon\*)malloc((\*n) \* sizeof(Mon));

for (int i = 0; i < \*n; i++) {

ds[i].id = i + 1;

printf("Mon %d:\n", i + 1);

printf("Ten: "); gets(ds[i].ten);

printf("Gia: "); scanf("%f", &ds[i].gia); getchar();

}

return ds;

}

// Ham in thuc don

void xemMenu(Mon\* ds, int n) {

printf("\n--- THUC DON ---\n");

for (int i = 0; i < n; i++) {

printf("%d. %s - %.0f VND\n", ds[i].id, ds[i].ten, ds[i].gia);

}

}

// Tao don hang

Don\* taoDon(Mon\* ds, int n) {

Don\* head = NULL, \*tail = NULL;

int id, sl;

while (1) {

printf("Nhap ID mon (0 de ket thuc): ");

scanf("%d", &id);

if (id == 0) break;

if (id < 1 || id > n) {

printf("Sai ID!\n");

continue;

}

printf("So luong: ");

scanf("%d", &sl);

Don\* node = (Don\*)malloc(sizeof(Don));

node->mon = ds[id - 1];

node->sl = sl;

node->next = NULL;

if (head == NULL)

head = tail = node;

else {

tail->next = node;

tail = node;

}

}

return head;

}

// In don hang va luu vao file

void xuatDon(Don\* head) {

FILE\* f = fopen("donhang.txt", "a");

float tong = 0;

printf("\n--- DON HANG ---\n");

fprintf(f, "=== DON HANG MOI ===\n");

while (head != NULL) {

float tien = head->sl \* head->mon.gia;

printf("%s x %d = %.0f\n", head->mon.ten, head->sl, tien);

fprintf(f, "%s x %d = %.0f\n", head->mon.ten, head->sl, tien);

tong += tien;

head = head->next;

}

printf("Tong tien: %.0f VND\n", tong);

fprintf(f, "Tong tien: %.0f\n\n", tong);

fclose(f);

}

// Tinh doanh thu tu file

void tinhDoanhThu() {

FILE\* f = fopen("donhang.txt", "r");

if (!f) {

printf("Chua co file don hang!\n");

return;

}

char line[100];

float tong = 0;

while (fgets(line, sizeof(line), f)) {

if (strncmp(line, "Tong tien:", 10) == 0) {

float tien;

sscanf(line, "Tong tien: %f", &tien);

tong += tien;

}

}

fclose(f);

printf(">> Doanh thu hien tai: %.0f VND\n", tong);

}

// Menu

void menu() {

printf("\n=== QUAN LY QUAN CAFE ===\n");

printf("1. Nhap thuc don\n");

printf("2. Xem thuc don\n");

printf("3. Tao don hang\n");

printf("4. Xem don hang hien tai\n");

printf("5. Tinh tong doanh thu\n");

printf("0. Thoat\n");

}

int main() {

Mon\* menuCafe = NULL;

int soMon = 0;

Don\* don = NULL;

int chon;

do {

menu();

printf("Chon: ");

scanf("%d", &chon);

getchar();

switch (chon) {

case 1:

if (menuCafe) free(menuCafe);

menuCafe = nhapMenu(&soMon);

break;

case 2:

if (menuCafe)

xemMenu(menuCafe, soMon);

else

printf("Chua nhap menu!\n");

break;

case 3:

if (!menuCafe) {

printf("Nhap menu truoc!\n");

} else {

if (don) {

// xóa don cu

Don\* p = don;

while (p) {

Don\* tmp = p;

p = p->next;

free(tmp);

}

}

don = taoDon(menuCafe, soMon);

}

break;

case 4:

if (don)

xuatDon(don);

else

printf("Chua co don hang!\n");

break;

case 5:

tinhDoanhThu();

break;

case 0:

printf("Tam biet!\n");

break;

default:

printf("Sai lua chon!\n");

}

} while (chon != 0);

// Giai phong bo nho

if (menuCafe) free(menuCafe);

while (don) {

Don\* tmp = don;

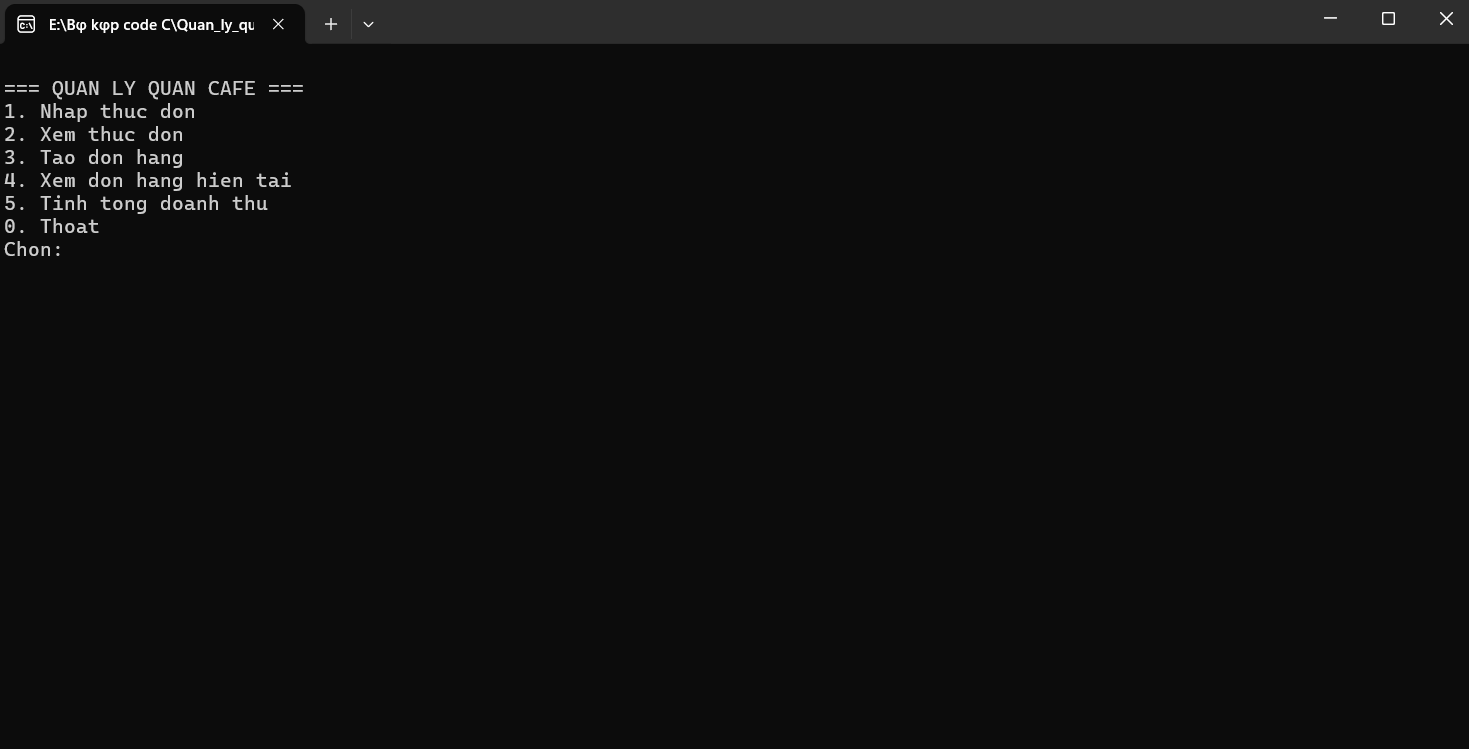
don = don->next;

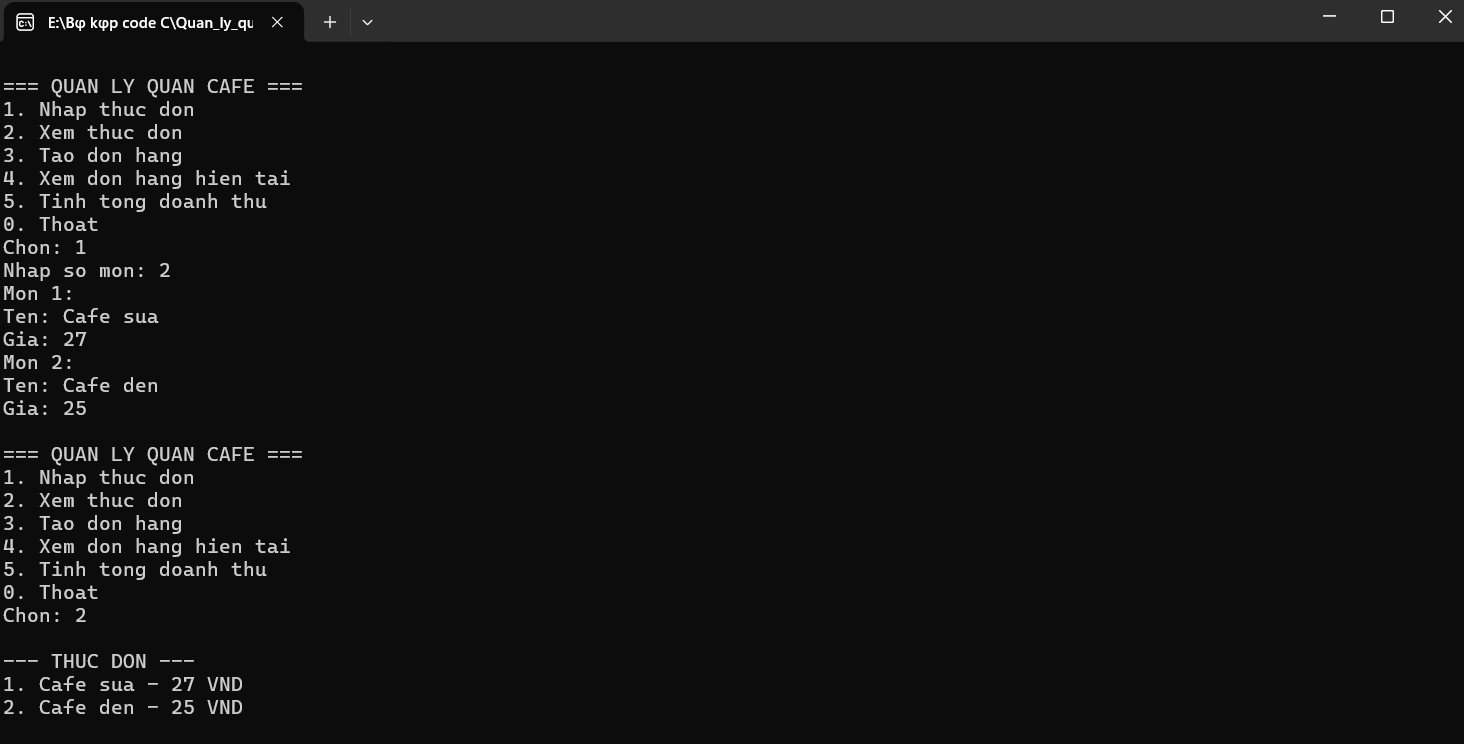
free(tmp);

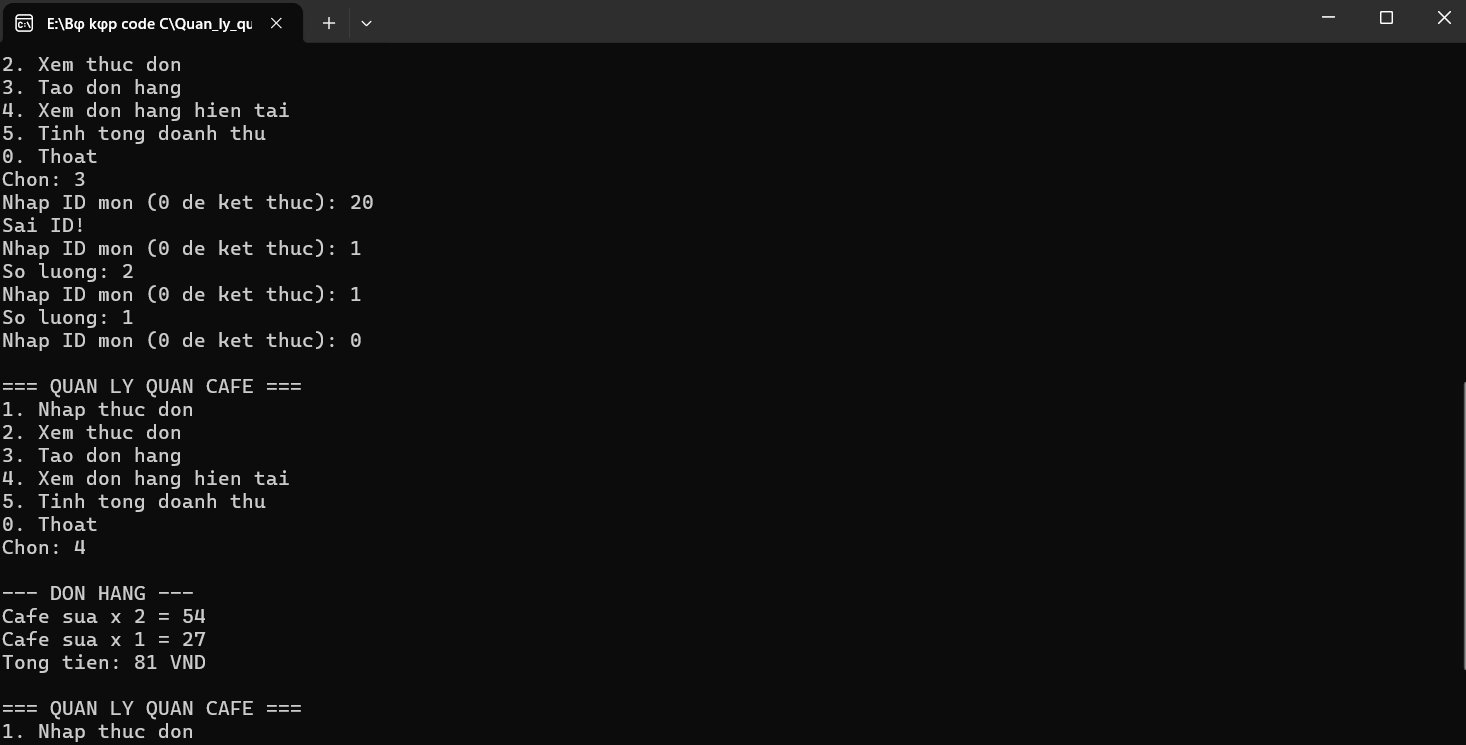
}

return 0;

}

****

****

****

