

---

## MỤC LỤC

---

1.	LAB 1 - Biến, Toán tử và Kiểu dữ liệu	Trang 2
2.	LAB 2 - Toán tử và Biểu thức	Trang 5
3.	LAB 3 - Điều kiện	Trang 13
4.	LAB 4 - Vòng lặp	Trang 20
5.	LAB 5 - Mảng	Trang 26
6.	LAB 6 - Con trỏ	Trang 34
7.	LAB 7 - Hàm	Trang 40
8.	LAB 8 - Chuỗi	Trang 43
9.	LAB 9 - Các Kiểu dữ liệu Nâng cao và Sắp xếp	Trang 48
10.	LAB 10 - Quản Lý Tập Tin	Trang 54

## Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Sử dụng biến, kiểu dữ liệu và biểu thức số học.

## Phần I – Trong thời gian 1 giờ 30 phút đầu:

### 3.1 Biến

Như chúng ta đã biết, Biến là tên đặt cho vị trí bộ nhớ máy tính, có thể dùng để lưu trữ các giá trị khác nhau tại những thời điểm khác nhau. Trong chương này, chủ yếu chúng ta sẽ học cách tạo và sử dụng biến.

#### 3.1.1 Tạo biến

Tạo biến bao gồm việc tạo kiểu dữ liệu và tên hợp lý cho biến, ví dụ:

```
int currentVal;
```

Trong ví dụ trên, tên biến là “currentVal” có kiểu dữ liệu là số nguyên (integer).

### 3.2 Kiểu dữ liệu

Kiểu dữ liệu định nghĩa loại giá trị mà sẽ được lưu trong một biến nào đó, ví dụ:

```
int currentVal;
```

Trong ví dụ trên “int” chỉ rằng biến currentVal sẽ lưu giá trị kiểu số nguyên (integer).

### 3.3 Biểu thức số học

Một biểu thức số học trong C bao gồm một tên biến nằm phía bên trái của dấu “=”, tên biến hoặc hằng nằm bên phải dấu “=”. Biến và hằng nằm bên phải của dấu “=” được nối với nhau bởi những toán tử số học như +, -, \*, và /. Thí dụ,

```
delta = alpha * beta / gamma + 3.2 * 2 / 5;
```

Bây giờ chúng ta xét một chương trình tính tiền lãi đơn giản như sau

#### Ví dụ 1:

1. Gọi trình soạn thảo để nhập những câu lệnh cho chương trình C.

2. Tạo ra một tập tin mới.

3. Nhập vào đoạn mã sau:

```
#include <stdio.h>

void main()
{
    int principal, period;
```

```
float rate, si;

principal = 1000;
period = 3;
rate = 8.5;

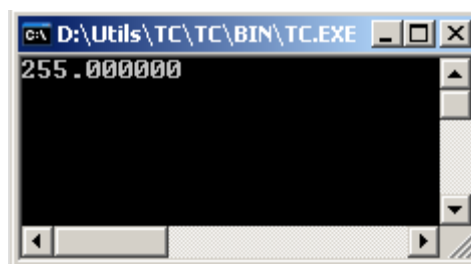
si = principal * period * rate / 100;

printf("%f", si);
}
```

Để thấy kết quả ở đầu ra, thực hiện tiếp các bước sau:

1. Lưu tập tin với tên myprogramI.C.
2. Biên dịch tập tin myprogramI.C.
3. Thực thi chương trình myprogramI.C.
4. Trở về trình soạn thảo.

Mẫu kết xuất cho chương trình trên như hình sau:



Hình 3.1: Kết quả của myprogramI.C

**Ví dụ 2:**

1. Tạo một tập tin mới.
2. Gõ vào mã sau:
 

```
#include <stdio.h>

void main()
{
    int a, b, c, sum;

    printf("\nEnter any three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

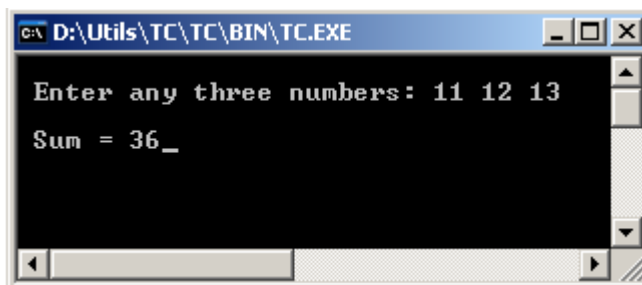
    sum = a + b + c;

    printf("\n Sum = %d", sum);
}
```
3. Lưu tập tin với tên myprogramII.C.
4. Biên dịch tập tin myprogramII.C.

5. Thực thi chương trình myprogramII.C.

6. Trở về trình soạn thảo.

Mẫu kết quả ở đầu ra của chương trình trên như hình sau:



Hình 3.2: Kết quả của myprogramII.C

## Phần II – Trong thời gian 30 phút kế tiếp:

1. Viết một chương trình nhập vào một số và tính bình phương của số đó.

**Hướng dẫn:** Thực hiện theo các bước sau:

- a. Nhập vào một số.
- b. Nhân số đó với chính nó và hiển thị kết quả đó.

## Bài tập tự làm

1. Viết chương trình tính diện tích và chu vi của một vòng tròn.

### Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Sử dụng được các toán tử số học, so sánh và luận lý
- Chuyển đổi các kiểu dữ liệu
- Hiểu được thứ tự ưu tiên giữa các toán tử.

Các bước trong chương này đã được nghiên cứu kỹ và giải thích chi tiết để chúng ta có thể hiểu rõ và áp dụng chúng một cách hoàn chỉnh. Ta hãy theo các bước cẩn thận.

### Phần I - Trong thời gian 1 giờ 30 phút đầu:

#### Ví dụ 1:

Trong chương này, ta sẽ viết một chương trình tính toán tiền lãi đơn giản (lãi thuần chưa tính tiền vốn vào) khi ta vay tiền.

Công thức để tính toán là  $p * n * r / 100$ . Ở đây ‘**p**’ có nghĩa là tiền vốn, ‘**n**’ là số năm và ‘**r**’ có nghĩa là tỉ lệ lãi suất.

Chương trình khai báo ba biến số thực ‘float’ có tên là p, n và r. Chú ý rằng, các biến được khai báo trên cùng một dòng mã thì ta dùng dấu phẩy (,) để phân chia chúng với nhau. Mỗi biến trên được gán một giá trị.

Xét dòng mã sau:

```
printf("\nAmount is: %f", p*n*r/100);
```

Trong *printf()* ở trên, chúng ta đã dùng ‘%f’ để hiển thị giá trị của biến kiểu float (số thực), giá trị biến này là  $p*n*r/100$ , công thức dùng tính lãi đơn giản được đưa vào trong *printf()*. Đó là p, n và r được nhân với nhau và kết quả được chia cho 100. Như vậy *printf()* sẽ hiển thị lãi đơn.

### Gọi Borland C.

#### 5.1 Tính lãi đơn

##### 1. Tạo ra một tập tin mới.

##### 2. Gõ đoạn mã sau trong ‘Edit window’:

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

void main()
{
    float p, n, r;
    clrscr();
    p = 1000;
    n = 2.5;
    r = 10.5;

    printf("\n Amount is: %f", p*n*r/100);
}
```

3. Lưu tập tin với tên *simple.c*.

4. Biên dịch tập tin *simple.c*.

5. Thực thi chương trình *simple.c*.

6. Trở về trình soạn thảo.

**Kết quả:**

```
The Amount is: 262.500000
```

## 5.2 Dùng toán tử số học

Trong phần này ta sẽ viết một chương trình có sử dụng toán tử số học.

Chương trình này khai báo bốn biến số nguyên tên là a, b, c và d. Giá trị sẽ gán cho các biến a, b và c là: a = 50, b = 24, c = 68

Xét dòng mã sau:

```
d = a*b+c/2;
```

a nhân với b. c chia cho 2. Kết quả của a\*b được cộng với thương số của c/2. Giá trị này sẽ gán cho d qua toán tử (=). Biểu thức được tính như sau:

1.  $50 * 24 = 1200$
2.  $68 / 2 = 34$
3.  $1200 + 34 = 1234$
4.  $d = 1234$

'printf()' : hiển thị giá trị của biến d.

Xét biểu thức:

```
d = a * (b+c+ (a-c) *b) ;
```

Ở đây dấu ngoặc đơn trong cùng có độ ưu tiên cao nhất. Do vậy, (a-c) được tính trước. Sau đó, tính tới các dấu ngoặc đơn ngoài. Kết quả của (a-c) được nhân cho b bởi vì '\*' có độ ưu tiên cao hơn '-' và '+'. Biểu thức được tính như dưới đây:

1.  $d = 50 * (24 + 68 + (50 - 68) * 24)$
2.  $d = 50 * (24 + 68 + (-18) * 24)$
3.  $d = 50 * (24 + 68 + (-432))$
4.  $d = 50 * (92 - 432)$
5.  $d = 50 * (-340)$
6.  $d = -17000$

Các biểu thức khác được tính tùy vào các toán tử đã được dùng. Kết quả được hiển thị bởi lệnh 'printf()'.

**1. Tạo mới một tập tin.**

**2. Gõ đoạn mã sau trong 'Edit window':**

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

```
void main()
{
    int a,b,c,d;
    clrscr();
    a = 50;
    b = 24;
    c = 68;
    d = a*b+c/2;
    printf("\n The value after a*b+c/2 is: %d", d);
    d = a%b;
    printf("\n The value after a mod b is: %d", d);
    d = a*b-c;
    printf("\n The value after a*b-c is: %d", d);
    d = a/b+c;
    printf("\n The value after a/b+c is: %d", d);

    d = a+b*c;
    printf("\n The value after a+b*c is: %d", d);

    d = (a+b)*c;
    printf("\n The value after (a+b)*c is: %d", d);

    d = a*(b+c+(a-c)*b);
    printf("\n The value after a*(b+c+(a-c)*b) is: %d", d);
}
```

**3. Lưu tập tin với tên *arith.c*.**

**4. Biên dịch tập tin *arith.c*.**

**5. Thực thi chương trình *arith.c*.**

**6. Trở về trình soạn thảo.**

**Kết quả xuất:**

```
The value after a*b+c/2 is: 1234
The value after a mod b is: 2
The value after a*b-c is: 1132
The value after a/b+c is: 70
The value after a+b*c is: 1682
The value after (a+b)*c is: 5032
The value after a*(b+c+(a-c)+b) is: -17000
```

### **5.3 Dùng toán tử so sánh và luận lý**

Trong phần này chúng ta sẽ viết một chương trình sử dụng toán tử so sánh và toán tử luận lý.

Ba biến số nguyên tên là a, b và c được khai báo trong chương trình này. Các giá trị gán cho biến như sau: a = 5, b = 6 & c = 7.

Xét những dòng mã sau:

1. a + b >= c;

Đầu tiên,  $a+b$  sẽ được tính (toán tử số học có độ ưu tiên cao hơn toán tử so sánh), kết quả là 11. Kế đến giá trị 11 được so sánh với  $c$ . Kết quả là 1(true) bởi vì  $11 > 7$ .

## 2. Xét biểu thức khác:

$a > 10 \ \&\& \ b < 5;$

Tính toán đầu tiên sẽ là  $a > 10$  và  $b < 5$ , bởi vì toán tử so sánh ( $> <$ ) có quyền ưu tiên cao hơn toán tử luận lý AND ( $\&\&$ ). Tính toán theo sau:

1.  $5 > 10 \ \&\& \ 6 < 5$
2. FALSE  $\&\&$  FALSE
3. FALSE tức là, 0

## 1. Tạo một tập tin mới.

## 2. Gõ đoạn mã sau vào ‘Edit window’:

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

void main()
{
    int a = 5, b = 6, c = 7;
    printf ("int a = 5, b = 6, c = 7;\n");
    printf("The value of a > b is \t%i\n\n", a > b);
    printf("The value of b < c is \t%i\n\n", b < c);
    printf("The value of a + b >= c is \t%i\n\n", a + b >= c);
    printf("The value of a - b <= b - c is \t%i\n\n", a - b <= b - c);
    printf("The value of b - a == b - c is \t%i\n\n", b - a == b - c);
    printf("The value of a*b != c * c is \t%i\n\n", a * b < c * c);
    printf("Result of a>10 && b<5 = %d\n\n", a>10 && b<5);
    printf("Result of a > 100 || b < 50 = %d\n\n", a>100 || b<50);
}
```

## 3. Lưu tập tin với tên *compare.c*.

## 4. Biên dịch tập tin *compare.c*.

## 5. Thực thi chương trình *compare.c*.

## 6. Trở về trình soạn thảo.

## Kết quả xuất:

```
int a = 5, b = 6, c = 7;

The value of a > b is          0

The value of  b < c  is          1

The value of  a + b >= c  is      1

The value of  a - b <= b - c  is    1

The value of  b - a == b - c  is    0

The value of  a * b != c * c  is    1
```



```
Result of a > 10 && b < 5 = 0
```

```
Result of a > 100 || b < 50 = 1
```

## 5.4 Chuyển đổi kiểu dữ liệu

Trong phần này, ta sẽ viết một chương trình để hiểu rõ việc chuyển đổi kiểu dữ liệu.

Trong biểu thức đầu tiên, tất cả đều là số nguyên 'int'  $40 / 17 * 13 / 3$  sẽ có kết quả là 8 ( $40 / 17$  làm tròn ra 2,  $2 * 13 = 26$ ,  $26 / 3$  làm tròn ra 8)

Biểu thức thứ hai như sau:

1. Định giá trị:  $40 / 17 * 13 / 3.0$
2.  $40 / 17$  làm tròn kết quả là 2
3.  $2 * 13 = 26$
4. Nhưng vì có số 3.0 đã ép kiểu phép chia cuối cùng thành số kiểu double, vì vậy  $26.0 / 3.0 = 8.666667$

Trong biểu thức thứ ba:

Nếu chúng ta di chuyển dấu chấm thập phân sang số 13 ( $40 / 17 * 13.0 / 3$ ), kết quả vẫn sẽ là 8.666667 bởi vì:

1.  $40 / 17$  làm tròn là 2
2. Số 13.0 ép kiểu dữ liệu phép nhân thành double nhưng kết quả vẫn là 26 vì  $2.0 * 13.0 = 26.0$
3. Và 26.0 ép phép chia cuối cùng thành kiểu double, vì vậy  $26.0 / 3.0 = 8.666667$

Trong biểu thức cuối:

nếu chúng ta di chuyển dấu chấm thập phân sang số 17 ( $40 / 17.0 * 13 / 3$ ), kết quả bây giờ sẽ là 10.196078 bởi vì:

1. 17.0 ép kiểu của phép chia đầu thành kiểu double và  $40.0 / 17.0 = 2.352941$
2.  $2.352941 * 13.0 = 30.588233$
3. và  $30.588233 / 3.0 = 10.196078$

### 1. Tạo một tập tin mới.

### 2. Gõ đoạn mã sau vào 'Edit window':

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

void main()
{
    clrscr();
    printf("40/17*13/3 = %d", 40/17*13/3);
    printf("\n\n40/17*13/3.0 = %lf", 40/17*13/3.0);
    printf("\n\n40/17*13.0/3 = %lf", 40/17*13.0/3);
    printf("\n\n40/17.0*13/3 = %lf", 40/17.0*13/3);
}
```

3. Lưu tập tin với tên *type.c*.

4. Biên dịch tập tin *type.c*.

5. Thực thi chương trình *type.c*.

6. Trở về trình soạn thảo.

**Kết quả xuất:**

```
40/17*13/3 = 8
40/17*13/3.0 = 8.666667
40/17*13.0/3 = 8.666667
40/17.0*13/3 = 10.196078
```

## 5.5 Thứ tự ưu tiên của các toán tử

Trong phần này chúng ta sẽ viết một chương trình để tìm hiểu thứ tự ưu tiên giữa các toán tử.

Biểu thức sau đây sẽ được tính như sau:

$(4-2*9/6 \leq 3 \ \&\& \ (10*2/4-3 > 3 \ || \ (1 < 5 \ \&\& \ 8 > 10)))$

Hãy theo các quy tắc chúng ta đã học trong chương “Toán tử và Biểu thức” (Chú ý rằng biểu thức được in đậm dưới đây sẽ được tính trước)

1.  $(4-2*9/6 \leq 3 \ \&\& \ (10*2/4-3 > 3 \ || \ (1 < 5 \ \&\& \ 8 > 10)))$
2.  $(4-2*9/6 \leq 3 \ \&\& \ (10*2/4-3 > 3 \ || \ (1 \ \&\& \ 0)))$
3.  $(4-2*9/6 \leq 3 \ \&\& \ (10*2/4-3 > 3 \ || \ 0))$
4.  $(4-2*9/6 \leq 3 \ \&\& \ (20/4-3 > 3 \ || \ 0))$
5.  $(4-2*9/6 \leq 3 \ \&\& \ (5-3 > 3 \ || \ 0))$
6.  $(4-2*9/6 \leq 3 \ \&\& \ (2 > 3 \ || \ 0))$
7.  $(4-2*9/6 \leq 3 \ \&\& \ (0 \ || \ 0))$
8.  $(4-2*9/6 \leq 3 \ \&\& \ 0)$
9.  $(4-18/6 \leq 3 \ \&\& \ 0)$
10.  $(4-3 \leq 3 \ \&\& \ 0)$
11.  $(1 \leq 3 \ \&\& \ 0)$
12.  $(1 \ \&\& \ 0)$
13. **0 (False)**

1. Tạo một tập tin mới.

2. Gõ đoạn mã sau vào cửa sổ soạn thảo:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    clrscr();
    printf("Result = %d", (4-2*9/6<=3 && (10*2/4-3 > 3 ||
                                         (1 < 5 && 8>10))));
}
```

3. Lưu tập tin với tên *precede.c*.

4. Biên dịch tập tin *precede.c*.

5. Thực thi chương trình *precede.c*.

6. Trở về trình soạn thảo.

**Kết quả xuất:**

Result = 0

## Phần II - Trong thời gian 30 phút kế tiếp:

1. Tính giá trị biểu thức sau:

$$10 * 3^6 * 6 + 5 - 2 \text{ AND } (2 * 2 + 6 / 3 > 1 \text{ OR } 2 > 8)$$

Thực hiện như sau:

Gõ vào biểu thức trên sử dụng câu lệnh printf(). AND được thay thế bởi && và OR được thay thế bởi ||.

2. Giả sử tất cả biến có kiểu là *int*. Tìm giá trị cho mỗi biến sau:

- $x = (2 + 3) * 6;$
- $x = (12 + 6) / 2 * 3;$
- $y = x = (2 + 3) / 4;$
- $y = 3 + 2 * (x = 7 / 2);$
- $x = (\text{int}) 3.8 + 3.3;$
- $x = (2 + 3) * 10.5;$
- $x = 3 / 5 * 22.0;$
- $x = 22.0 * 3 / 5;$

## Bài tập tự làm

1. Tính giá trị được gán (nằm phía bên trái) cho mỗi trường hợp sau:

```
int s, m = 3, n = 5, r, t;
```

```
float x = 3.0, y;
```

```
t = n/m;
```

```
r = n%m;
```

```
y = n/m;
```

```
t = x*y-m/2;
```

```
x = x*2.0;
```

```
s = (m+n) / r;
```

```
y = --n;
```

2. Viết một chương trình nhập vào một số thực. Đơn vị tính cho số này là centimet (cm). Hãy in ra số tương đương tính bằng foot (số thực, có 1 số lẻ thập phân) và inch (số thực, có 1 số lẻ thập phân). Độ chính xác của foot và inch là một số lẻ thập phân.

Hướng dẫn: 2.54 centimeters = 1 inch, và 12 inches = 1 foot.

Nếu giá trị nhập vào là 333.3, kết quả là:

333.3 centimeters tương đương 10.9 feet.

333.3 centimeters tương đương 131.2 inches.

3. Tìm giá trị của iResult cho những câu lệnh sau:

```
int iResult, a = 10, b = 8, c = 6, d = 5, e = 2;
```

```
iResult = a - b - c - d;  
iResult = a - b + c - d;  
iResult = a + b / c / d;  
iResult = a + b / c * d;  
iResult = a / b * c * d;  
iResult = a % b / c * d;  
iResult = a % b % c % d;  
iResult = a - (b - c) - d;  
iResult = (a - (b - c)) - d;  
iResult = a - ((b - c) - d);  
iResult = a % (b % c) * d * e;  
iResult = a + (b - c) * d - e;  
iResult = (a + b) * c + d * e;  
iResult = (a + b) * (c / d) % e;
```

### Mục tiêu:

Kết thúc bài học này, bạn có thể:

#### ➤ Sử dụng:

- Câu lệnh if
- Câu lệnh if – else
- Câu lệnh với nhiều if
- Câu lệnh if lồng nhau
- Câu lệnh switch.

Các bước trong bài học này được trình bày chi tiết, rõ ràng và cẩn thận. Điều này giúp ta hiểu rõ về công cụ lập trình. Thực hiện theo các bước sau thật cẩn thận.

### Phần I - Trong thời gian 1 giờ 30 phút đầu:

#### 8.1 Lệnh if:

##### Ví dụ 1:

Trong phần này chúng ta sẽ viết một chương trình để tính tiền hoa hồng phải trả cho người bán hàng dựa vào số lượng hàng họ bán được.

##### Bài toán:

Công ty SARA sẽ trả 10% tiền hoa hồng cho nhân viên bán hàng của công ty nếu doanh số bán hàng của nhân viên đạt \$10,000 hoặc hơn. Tính tiền hoa hồng phải trả cuối mỗi tháng.

Bài toán khai báo hai biến kiểu 'float' là `sales_amt` và `com`. Chú ý, các biến được khai báo trong cùng một dòng trong chương trình thì sử dụng dấu phẩy (,) để phân cách giữa các biến.

Theo dõi đoạn mã lệnh dưới đây:

```
printf("Enter the Sales Amount: ");
scanf("%f", &sales_amt);
```

Trong hàm `printf()`, chúng ta hiển thị thông điệp yêu cầu nhập doanh số bán hàng, và trong hàm `scanf()` sử dụng `%f` để nhận một giá trị từ người dùng. Giá trị nhập vào sẽ được gán cho biến `sales_amt`.

```
if (sales_amt >= 10000)
    com = sales_amt * 0.1;
```

Câu lệnh trên được dùng để kiểm tra giá trị của biến `sales_amt` có lớn hơn hoặc bằng 10000 không. `>=` là toán tử so sánh, sẽ trả về giá trị **đúng** hoặc **sai**. Trong trường hợp, nếu bạn nhập vào giá trị 15000, điều kiện (`sales_amt >= 10000`) có kết quả là **đúng**. Nếu **đúng**, nó sẽ thực thi câu lệnh `com = sales_amt * 0.1`. Bây giờ giá trị của biến `com` sẽ là **1500**. Nếu điều kiện là **sai**, nó sẽ in ra giá trị tiền hoa hồng là 0. Ở đây chúng ta thấy, điều kiện **if** chỉ có một lệnh duy nhất. Nếu có nhiều hơn một lệnh cho điều kiện **if**, các lệnh phải được đặt trong cặp dấu ngoặc {}.

```
printf("\n Commission = %f", com);
```

Câu lệnh trên được sử dụng để hiển thị giá trị tiền hoa hồng. `%f` được sử dụng để hiển thị giá trị của một biến 'float' được đưa ra sau dấu phẩy ở cuối của hàm `printf()`. Vì vậy, `printf()` ở đây hiển thị tiền hoa hồng tính được.

#### 8.1.1 Tính tiền hoa hồng:

##### 1. Tạo một tập tin mới.

##### 2. Nhập vào đoạn mã lệnh sau đây trong cửa sổ 'Edit Window':

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

```
void main()
{
    float com = 0, sales_amt;
    clrscr();
    printf("Enter the Sales Amount: ");
    scanf("%f", &sales_amt);
    if (sales_amt >= 10000)
        com = sales_amt * 0.1;
    printf("\n Commission = %f", com);
}
```

3. Lưu tập tin với tên *comm.C*.
4. Biên dịch tập tin *comm.C*.
5. Thực thi chương trình *comm.C*.
6. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

```
Enter the Sales Amount: 15000
Commission = 1500.000
```

## 8.2 Lệnh 'if-else':

Trong phần này chúng ta sẽ viết một chương trình sử dụng lệnh *if-else*. Chương trình hiển thị số lớn hơn trong hai số.

Theo dõi các dòng mã lệnh sau:

```
if (num1 > num2)
    printf("\n The greater number is: %d", num1);
else
    printf("\ The greater number is: %d", num2);
```

Trong đoạn mã lệnh này hàm *printf()* đầu tiên chỉ được thực thi nếu giá trị của biến *num1* lớn hơn giá trị của biến *num2*, khi đó phần **else** được bỏ qua. Nếu giá trị của biến *num1* không lớn hơn giá trị của biến *num2*, hàm *printf()* được bỏ qua. Trong trường hợp này hàm *printf()* thứ hai, lệnh theo sau **else**, được thực thi.

Trong chương trình sau, bởi vì giá trị của biến *num1* lớn hơn *num2*, hàm *printf()* đầu tiên được thực thi.

1. Tạo một tập tin mới.
2. Nhập vào đoạn mã lệnh sau đây trong cửa sổ 'Edit Window':

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int Num1, Num2;
    clrscr();
    Num1 = 540;
    Num2 = 243;
    if (Num1 > Num2)
        printf("\n The Greater Number Is: %d", Num1);
    else
        printf("\n The Greater Number Is: %d", Num2);
}
```

1. Lưu tập tin với tên *ifelse.C*.
2. Biên dịch chương trình *ifelse.C*.
3. Thực thi chương trình *ifelse.C*.
4. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

The greater number is: 540

### 8.3 Lệnh ‘if-else-if’:

Trong phần này chúng ta sẽ viết một chương trình sử dụng lệnh *if – else – if*. Chương trình sẽ hiển thị số lớn hơn trong hai số, hoặc sẽ hiển thị các số là bằng nhau.

Trong chương trình ở phần trước, có hai biến ‘số nguyên’ *num1* và *num2* được khai báo. Các biến được gán giá trị.

Quan sát các dòng mã lệnh sau:

```
if (num1 == num2)
    printf("\nNumbers are Equal");
else if (num1 < num2)
    printf("\nThe Larger Number is: %d", num2);
else
    printf("\nThe Larger Number is: %d", num1);
```

Trong đoạn mã lệnh trên, điều kiện ‘**if**’ đầu tiên (*num1 == num2*) kiểm tra xem giá trị của biến *num1* có bằng biến *num2* không. Trong C, ký hiệu *==* được sử dụng để kiểm tra hai toán hạng có bằng nhau không. Nếu điều kiện đầu tiên (*num1 == num2*) có giá trị **true** thì hàm *printf()* theo ngay sau sẽ được thực thi. Nếu điều kiện đầu tiên không đúng, điều kiện của **else-if** sẽ được kiểm tra. Trong trường hợp điều kiện này (*num1 < num2*) thỏa, hàm *printf()* theo sau nó sẽ được thực thi. Nếu cả hai điều kiện của **if** và **else if** đều không thỏa mãn, thì câu lệnh sau **else** cuối cùng sẽ được thực thi.

Trong chương trình dưới đây, vì giá trị của *num1* nhỏ hơn *num2*, lệnh *printf()* thứ hai sẽ được thực thi.

#### 1. Tạo một tập tin mới.

#### 2. Nhập đoạn mã lệnh sau trong cửa sổ ‘Edit Window’.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num1, num2;
    num1 = 77;
    num2 = 90;
    if (num1 == num2)
        printf("\nThe Numbers are equal");
    else if (num1 < num2)
        printf("\n The Larger Number is: %d", num2);
    else
        printf("\n The Larger Number is: %d", num1);
}
```

#### 1. Lưu tập tin với tên *ifelseif.C*.

#### 2. Biên dịch tập tin *ifelseif.C*.

#### 3. Thực thi chương trình *ifelseif.C*.

#### 4. Trở về cửa sổ ‘Edit Window’.

## KẾT QUẢ:

The Larger Number is: 90

### 8.4 Lệnh ‘if lồng nhau’:

Trong phần này chúng ta sẽ viết một chương trình để hiểu rõ về lệnh ‘**if lồng nhau**’.

## Bài toán:

Công ty MONTEK đã ra quyết định chi tiền hoa hồng cho bộ phận bán hàng tùy thuộc vào doanh thu bán sản phẩm. Tỷ lệ hoa hồng được tính như sau:

Doanh thu bán	Loại sản phẩm	Hoa hồng
> 10,000\$	A	10%
	--	8%
<= 10,000	--	5%

Tính tiền hoa hồng cuối mỗi tháng.

Trong chương trình này chúng ta tính tiền hoa hồng dựa vào loại sản phẩm và lượng sản phẩm bán được.

Quan sát các dòng mã lệnh sau:

```
printf("\nEnter the Sales Amount: ");
scanf("%f", &sales_amt);
printf("\n Enter the Grade: ");
scanf("%c", &grade);
```

Hàm *scanf()* đầu tiên được dùng để nhập doanh thu bán hàng, và hàm *scanf()* thứ hai được sử dụng để nhập loại sản phẩm. Định dạng *%c* được sử dụng để nhận một ký tự từ người dùng.

```
if (sales_amt > 10000)
    if (grade == 'A')
        com = sales_amt * 0.1;
    else
        come = sales_amt * 0.08;
else
    com = sales_amt * 0.05;
```

Giả sử chúng ta nhập doanh thu bán hàng là 15000 và loại sản phẩm là 'A'. Chương trình sẽ kiểm tra điều kiện **if** (*sales\_amt* > 10000); vì điều kiện này có kết quả đúng, chương trình sẽ tiếp tục lệnh **if** thứ hai (*grade* == 'A'). Điều kiện này cũng thỏa, vì vậy tiền hoa hồng được tính **com = sales\_amt \* 0.1**.

Chúng ta xem một tình huống khác với doanh thu bán hàng là 15000 và loại sản phẩm là 'B'. Chương trình sẽ kiểm tra điều kiện **if** đầu tiên (*sales\_amt* > 10000), điều kiện này thỏa. Và chương trình thực hiện tiếp lệnh **if** thứ hai, trong trường hợp này điều kiện không thỏa mãn, chương trình sẽ chuyển đến lệnh **else** tương ứng, ...

```
else
    com = sales_amt * 0.08;
```

Nếu chúng ta nhập giá trị 10000 hoặc nhỏ hơn, chương trình sẽ chuyển đến điều kiện **else** sau cùng và tính tiền hoa hồng theo công thức:

```
com = sales_amt * 0.05;
```

1. Tạo một tập tin mới.
2. Nhập đoạn mã lệnh sau trong cửa sổ 'Edit Window'.

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

void main()
{
    float com = 0, sales_amt;
    char grade;
```



```
clrscr();
printf("\nEnter the Sales Amount: ");
scanf("%f", &sales_amt);
printf("\n Enter the Grade: ");
scanf("%c", &grade);
if (sales_amt > 10000)
    if (grade == 'A')
        com = sales_amt * 0.1;
    else
        com = sales_amt * 0.08;
else
    com = sales_amt * 0.05;
printf("\n Commission = %f", com);
}
```

3. Lưu tập tin với tên *nestif.C*.
4. Biên dịch tập tin *nestif.C*.
5. Thực thi chương trình *nestif.C*.
6. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

```
Enter the Sales Amount: 15000
Enter the grade: A
Commission = 1500
```

## 8.5 Sử dụng lệnh 'switch':

Trong phần này chúng ta sẽ sử dụng lệnh 'switch'. Chương trình hiển thị kết quả tùy vào toán tử toán học được sử dụng.

Trong chương trình này có hai biến số nguyên là *num1* và *num2* và một biến ký tự *op* được khai báo. Các biến được gán giá trị. Một phép toán số học được lưu trong biến *op*.

Biến *op* được truyền vào biểu thức sau 'switch'. **case** đầu tiên so sánh giá trị của biến *op* với '+'. Nếu nhân (+) so khớp với giá trị trong *op*, thì các dòng mã lệnh sau được thực thi:

```
res = num1 + num2;
printf("\n The sum is: %d", res);
break;
```

Tổng của *num1* và *num2* được lưu trong biến *res*. Lệnh *printf()* hiển thị giá trị của biến *res*. Lệnh *break* để thoát khỏi lệnh 'switch'.

Trong **case** thứ hai, giá trị của *op* được so sánh với '-', và sau đó biểu thức *num1 - num2* được thực thi. Tương tự, nếu giá trị của *op* là '\*' và '/', *num1 \* num2* và *num1 / num2* được thực thi.

Nếu không có trường hợp nào ở trên thỏa mãn, thì lệnh *printf()* của 'default' sẽ được thực thi.

1. Tạo một tập tin mới.
2. Nhập đoạn mã lệnh sau trong cửa sổ 'Edit Window'.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num1, num2, res;
    char op;
    num1 = 90;
    num2 = 33;
```

```

op = '-';
clrscr();
switch (op)
{
    case '+':
        res = num1 + num2;
        printf("\nThe Sum is: %d", res);
        break;
    case '-':
        res = num1 - num2;
        printf("\nNumber after Subtraction: %d", res);
        break;
    case '/':
        res = num1 / num2;
        printf("\n nNumber after Division: %d", res);
        break;
    case '*':
        res = num1 * num2;
        printf("\n nNumber after multiplication: %d", res);
        break;
    default:
        printf("\nInvalid");
        break;
}
}

```

3. Lưu tập tin với tên *case.C*.
4. Biên dịch tập tin *case.C*.
5. Thực thi chương trình *case.C*.
6. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

Number after Subtraction: 57

## Phần II - Trong thời gian 30 phút kế tiếp:

1. Một học viên được kiểm tra 3 môn học. Mỗi bài kiểm tra tối đa là 100 điểm. Điểm trung bình của học viên được tính, và học viên được xếp loại tùy thuộc vào kết quả của điểm trung bình theo qui luật sau:

Điểm trung bình	Loại
$\geq 90$	E+
$80 < 90$	E
$70 < 80$	A+
$60 < 70$	A
$50 < 60$	B+
$< 50$	RỐT

Để thực hiện:

- a. Nhập vào điểm của 3 môn học và lưu trong 3 biến khác nhau là *M1*, *M2* và *M3*.
- b. Tính điểm trung bình ( $\text{avg} = (M1 + M2 + M3)/3$ ).
- c. Xác định loại của học viên dựa trên điểm trung bình đã tính.
- d. Hiển thị loại.

---

**Bài tập tự làm**

---

1. Khai báo hai biến  $x$  và  $y$ . Gán trị cho các biến này. Số  $x$  được in ra màn hình chỉ khi  $x$  nhỏ hơn 2000 và lớn hơn 3000, và số  $y$  chỉ được in ra màn hình khi  $y$  nằm giữa 100 và 500.

2. Viết chương trình trình bày khả năng của máy tính của bạn. Người dùng nhập và một ký tự trong bảng chữ cái và chương trình hiển thị ngôn ngữ lập trình tương ứng. Một vài ví dụ nhập và xuất như sau:

**Nhập**

A hoặc a  
B hoặc b  
C hoặc c  
D hoặc d  
F hoặc f  
P hoặc p  
V hoặc v

**Xuất**

Ada  
Basic  
COBOL  
dBASE III  
Fortran  
Pascal  
Visual C++

Sử dụng lệnh 'switch' để chọn và hiển thị thông điệp thích hợp. Sử dụng nhãn **default** để hiển thị thông điệp nếu ký tự nhập không nằm trong danh sách liệt kê trên.

3. Nhập giá trị vào ba biến và in ra màn hình giá trị lớn nhất.

## Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Sử dụng cấu trúc vòng lặp
- Viết một vài chương trình:
  - Sử dụng vòng lặp 'for'
  - Sử dụng vòng lặp 'while'
  - Sử dụng vòng lặp 'do...while'.

Các bước trong bài học này được trình bày chi tiết, rõ ràng và cẩn thận. Điều này giúp ta hiểu rõ về công cụ lập trình. Thực hiện theo các bước sau thật cẩn thận.

## Phần I - Trong thời gian 1 giờ 30 phút đầu:

### 10.1 Sử dụng vòng lặp 'for':

Trong phần này chúng ta sẽ viết một chương trình sử dụng vòng lặp 'for'. Chương trình hiển thị các số chẵn từ 1 đến 30.

Trong chương trình, một biến 'số nguyên', *num*, được khai báo. Vòng lặp 'for' được sử dụng để hiển thị các số chẵn đến 30. Đối số đầu tiên của vòng lặp 'for', khởi tạo biến *num* là 2. Đối số thứ hai của vòng lặp 'for', kiểm tra giá trị của biến có nhỏ hơn hoặc bằng 30 không. Nếu điều kiện này thỏa, lệnh trong vòng lặp được thực hiện. Lệnh 'printf()' được sử dụng để hiển thị giá trị của biến *num*.

Trong đối số thứ ba, giá trị của biến *num* được tăng lên 2. Trong C, *num +=2* giống như *num = num + 2*. Lệnh 'printf' được thực thi khi đối số thứ hai vẫn thỏa. Một khi giá trị của biến trở nên lớn hơn 30, điều kiện không thỏa nữa và vì vậy vòng lặp không được thực thi. Dấu ngoặc nhọn {} không cần thiết khi chỉ có một câu lệnh hiện diện trong vòng lặp, nhưng việc sử dụng cặp dấu ngoặc {} là một thói quen lập trình tốt.

#### 1. Tạo một tập tin mới.

#### 2. Nhập vào đoạn mã lệnh sau đây trong cửa sổ 'Edit Window':

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int    num;
    clrscr();
    printf("The even Numbers from 1 to 30 are \n ");
    for (num = 2; num <= 30; num +=2)
        printf("%d\n", num);
}
```

#### 3. Lưu tập tin với tên *for.C*.

#### 4. Biên dịch tập tin *for.C*.

#### 5. Thực thi chương trình *for.C*.

#### 6. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

```
The even Numbers from 1 to 30 are
2
4
```

6  
8  
10  
12  
14  
16  
18  
20  
22  
24  
26  
28  
30

## 10.2 Sử dụng vòng lặp ‘while’:

Trong phần này chúng ta sẽ viết một chương trình sử dụng vòng lặp ‘while’. Chương trình hiển thị các số từ 10 đến 0 theo thứ tự đảo ngược.

Trong chương trình có một biến số nguyên *num*. Biến được khởi tạo.

Xét dòng mã lệnh sau:

```
while (num >= 0)
{
    printf("\n%d", num);
    num--;
}
```

Lệnh ‘while’ kiểm tra, giá trị của biến *num* có lớn hơn 0 hay không. Nếu điều kiện thỏa lệnh ‘printf()’ được thực thi và giá trị của biến *num* giảm 1. Trong C, *num--* làm việc giống như *num = num - 1*. Vòng lặp ‘while’ vẫn tiếp tục khi giá trị của biến lớn hơn 1 hoặc bằng 0.

### 1. Tạo một tập tin mới.

### 2. Nhập vào đoạn mã lệnh sau đây trong cửa sổ ‘Edit Window’:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int num;
    clrscr();
    num = 10;
    printf("\n Countdown");
    while (num >= 0)
    {
        printf("\n%d", num);
        num--;
    }
}
```

### 3. Lưu tập tin với tên *while.C*.

### 4. Biên dịch tập tin *while.C*.

### 5. Thực thi chương trình *while.C*.

### 6. Trở về cửa sổ ‘Edit Window’.

## KẾT QUẢ:

```

Countdown
10
9
8
7
6
5
4
3
2
1
0

```

### 10.3 Sử dụng vòng lặp ‘do...while’:

Trong phần này chúng ta sẽ viết một chương trình sử dụng vòng lặp ‘do...while’. Vòng lặp ‘do... while’ khác với vòng lặp ‘while’ là nó thực thi lệnh trước khi đánh giá biểu thức. Một điều quan trọng cần phải nhớ là, không giống như vòng lặp ‘while’, phần thân của vòng lặp ‘do’ sẽ được thực hiện ít nhất một lần. Bởi vì vòng lặp ‘while’ đánh giá biểu thức trước khi thực thi lệnh, nếu điều kiện là sai (0) ngay lúc bắt đầu, phần lệnh sẽ không bao giờ được thực thi.

Chương trình sẽ nhận vào các số nguyên và hiển thị chúng cho đến khi số 0 được nhập vào. Sau đó nó sẽ thoát khỏi vòng lặp ‘do...while’ và in ra các số nguyên đã được nhập.

Chương trình khai báo hai biến *cnt* và *cnt1*. Bên trong vòng lặp ‘do –while’ chúng ta sẽ nhập số bằng cách sử dụng mã lệnh sau:

```

printf("\nEnter a Number: ");
scanf("%d", &cnt);

```

Lệnh bên dưới sẽ hiển thị số đã nhập.

```

printf("No. is %d", cnt);

```

*cnt1++* sẽ tăng giá trị biến *cnt1* lên 1. Giả sử nếu chúng ta nhập vào số 0, trước hết nó sẽ in giá trị và sau đó kiểm tra điều kiện. Trong trường hợp này điều kiện là sai. Nó sẽ thoát khỏi vòng lặp và in giá trị của biến *cnt1*. Biến *cnt1* được giảm một đơn vị trước khi in ra bởi vì số nguyên cuối cùng (0) không được đếm.

#### 1. Tạo một tập tin mới.

#### 2. Nhập vào đoạn mã lệnh sau đây trong cửa sổ ‘Edit Window’:

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int    cnt = 0, num;
    clrscr();

    do
    {
        printf("\n Enter a Number: ");
        scanf("%d", &num);
        printf("No. is %d", num);
        cnt++;
    } while (num != 0);
    printf("\n The total numbers entered were %d", --cnt);
}

```

3. Lưu tập tin với tên *dowhile.C*.
4. Biên dịch tập tin *dowhile.C*.
5. Thực thi chương trình *dowhile.C*.
6. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

```
Enter a number 11
No is 11
Enter a number 50
No is 50
Enter a number 0
No is 0
The total numbers entered were 2
```

## 10.4 Sử dụng lệnh break:

Lệnh break giúp thoát ra khỏi vòng lặp for, while, do-while hay lệnh switch ngay lập tức.

Chương trình sau minh họa các dùng của lệnh *break*.

Quan sát đoạn mã lệnh sau:

```
for (cnt = 1; cnt <= 10; cnt++)
{
    if (cnt == 5)
        break;
    printf("%d\n", cnt);
}
```

Đoạn mã lệnh trên sử dụng một vòng lặp 'for' để in ra các giá trị từ 1 đến 10. Giá trị của biến *cnt* được khởi tạo là 1, và sau đó nó sẽ kiểm tra điều kiện. Nếu điều kiện là đúng, nó sẽ thực thi các câu lệnh bên trong vòng lặp.

Trong trường hợp này, chương trình chỉ in ra 1, 2, 3, 4. Khi giá trị của biến *cnt* là 5, điều kiện *if* trở nên đúng, và điều khiển sẽ thoát khỏi vòng lặp.

1. Tạo một tập tin mới.
2. Nhập vào đoạn mã lệnh sau đây trong cửa sổ 'Edit Window':

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int cnt;
    clrscr();
    for (cnt = 1; cnt <= 10; cnt++)
    {
        if (cnt == 5)
            break;
        printf("%d\t", cnt);
    }
}
```

3. Lưu tập tin với tên *breakex.C*.
4. Biên dịch tập tin *breakex.C*.
5. Thực thi chương trình *breakex.C*.
6. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

1	2	3	4
---	---	---	---

### 10.4 Sử dụng lệnh *continue*:

Lệnh *continue* khi được dùng trong một vòng lặp *while*, *for*, hoặc *do-while* sẽ bỏ qua tất cả các câu lệnh phía sau và lần lặp kế tiếp được thực thi.

Chương trình sau minh họa cách sử dụng của lệnh *continue*.

Xem đoạn mã lệnh sau:

```
for ( cnt = 1; cnt <=10; cnt++)
{
    if (cnt ==5)
        continue;
    printf("%d\t", cnt);
}
```

Đoạn mã lệnh sử dụng một vòng lặp *for* để in ra các giá trị từ 1 đến 10. Giá trị của biến *cnt* được khởi tạo là 1, sau đó điều kiện sẽ được kiểm tra. Nếu điều kiện là đúng, các lệnh trong vòng lặp sẽ được thực thi.

Trong trường hợp này, chương trình chỉ in ra các số 1, 2, 3, 4, 6, 7, 8, 9 và 10. Khi giá trị của biến *cnt* là 5, điều kiện *if* trở nên đúng và điều khiển trở về đầu vòng lặp *for* mà không in ra giá trị 5.

#### 1. Tạo một tập tin mới.

#### 2. Nhập vào đoạn mã lệnh sau đây trong cửa sổ 'Edit Window':

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int cnt;
    clrscr();
    for (cnt = 1; cnt <= 10; cnt++)
    {
        if (cnt == 5)
            continue;
        printf("%d\t", cnt);
    }
}
```

#### 3. Lưu tập tin với tên *continueex.C*.

#### 4. Biên dịch tập tin *continueex.C*.

#### 5. Thực thi chương trình *continueex.C*.

#### 6. Trở về cửa sổ 'Edit Window'.

## KẾT QUẢ:

1	2	3	4	6	7	8	9	10
---	---	---	---	---	---	---	---	----



---

**Phần II: Trong thời gian 30 phút kế tiếp:**

1. Tìm giai thừa của một số.

Gợi ý: Xem công thức tính giai thừa của một số :

- $n! = n * (n-1) * (n-2) * ... * 1$
- $4! = 4 * 3 * 2 * 1$
- $1! = 1$
- $0! = 1$

Gợi ý:

- Nhập vào một số.
- Khởi đầu, thiết đặt giai thừa của một số là 1.
- Trong khi số còn lớn hơn 1.
- Tính giai thừa của một số bằng giai thừa nhân với số đó.
- Giảm số xuống một đơn vị.
- In ra giai thừa.

---

**Bài tập tự làm**

1. Khai báo một biến lưu tuổi của một người. In ra tên của người đó với số lần in bằng số tuổi.
2. Viết chương trình sinh dãy số theo dạng sau:  
1  
12  
123  
1234  
12345  
123456  
1234567  
12345678  
123456789
3. Viết chương trình in ra bảng cửu chương của một số được nhập vào.

### Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Sử dụng mảng một chiều
- Sử dụng mảng hai chiều.

Các bước trong bài học này được trình bày chi tiết, rõ ràng và cẩn thận. Điều này giúp ta hiểu rõ về công cụ lập trình. Thực hiện theo các bước sau thật cẩn thận.

### Phần I – Trong thời gian 1 giờ 30 phút đầu:

#### 12.1 Mảng

Các mảng có thể được phân làm hai dạng dựa vào chiều của mảng: Mảng một chiều và mảng đa chiều. Trong bài này, chúng ta sẽ tập trung vào cách tạo và sử dụng các mảng.

##### 12.1.1 Sự sắp xếp một mảng một chiều

Mảng một chiều có thể được sử dụng để lưu trữ một tập các giá trị có cùng kiểu dữ liệu. Xét một tập điểm của sinh viên trong một môn học. Chúng ta sẽ sắp xếp các điểm này theo thứ tự giảm dần.

Các bước sắp xếp mảng một chiều theo thứ tự giảm như sau:

#### 1. Nhập vào số lượng các điểm.

Để thực hiện điều này, một biến phải được khai báo và giá trị của biến phải được nhập. Mã lệnh như sau:

```
int n;
printf("\n Enter the total number of marks to be entered : ");
scanf("%d", &n);
```

#### 2. Nhập vào tập các điểm.

Để nhập vào tập các giá trị cho một mảng, mảng phải được khai báo. Mã lệnh như sau,

```
int num[100];
```

Số phần tử của mảng được xác định bằng giá trị đã nhập vào biến n. n phần tử của mảng phải được khởi tạo giá trị. Để nhập n giá trị, sử dụng vòng lặp **for**. Một biến nguyên cần được khai báo để sử dụng như là chỉ số của mảng. Biến này giúp truy xuất từng phần tử của mảng. Sau đó giá trị của các phần tử mảng được khởi tạo bằng cách nhận các giá trị nhập vào từ người dùng. Mã lệnh như sau:

```
int l;
for(l = 0; l < n; l++)
{
    printf("\n Enter the marks of student %d : ", l + 1);
    scanf("%d", &num[l]);
}
```

Vì các chỉ số của mảng luôn bắt đầu từ 0 nên chúng ta cần khởi tạo biến l là 0. Mỗi khi vòng lặp được thực thi, một giá trị nguyên được gán đến một phần tử của mảng.

#### 3. Tạo một bản sao của mảng.

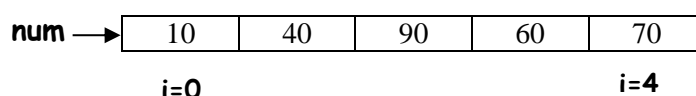
Trước khi sắp xếp mảng, tốt hơn là nên giữ lại mảng gốc. Vì vậy một mảng khác được khai báo và các phần tử của mảng thứ nhất có thể được sao chép vào mảng mới này. Các dòng mã lệnh sau được sử dụng để thực hiện điều này:

```
int desnum[100], k;
for(k = 0; k < n; k++)
    desnum[k] = num[k];
```

#### 4. Sắp xếp mảng theo thứ tự giảm dần.

Để sắp xếp một mảng, các phần tử trong mảng cần phải được so sánh với những phần tử còn lại. Cách tốt nhất để sắp xếp một mảng, theo thứ tự giảm dần, là chọn ra giá trị lớn nhất trong mảng và hoán vị nó với phần tử đầu tiên. Một khi điều này được thực hiện xong, giá trị lớn thứ hai trong mảng có thể được hoán vị với phần tử thứ hai của mảng, phần tử đầu tiên của mảng được bỏ qua vì nó đã là phần tử lớn nhất. Tương tự, các phần tử của mảng được loại ra tuần tự đến khi phần tử lớn thứ  $n$  được tìm thấy. Trong trường hợp mảng cần sắp xếp theo thứ tự tăng dần giá trị lớn nhất sẽ được hoán vị với phần tử cuối cùng của mảng.

Quan sát ví dụ một dãy số để hiểu được giải thuật. Hình 12.1 trình bày một mảng số nguyên cần được sắp xếp.

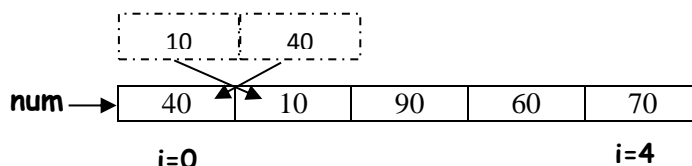


**Hình 12.1: Mảng num với chỉ số i (5 phần tử)**

Để sắp xếp mảng này theo thứ tự giảm dần,

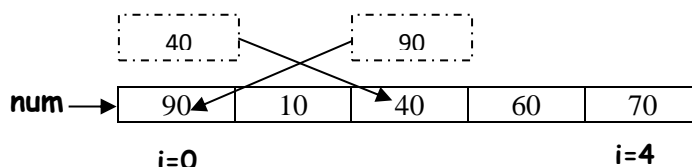
a. Chúng ta cần tìm phần tử lớn nhất và hoán vị nó vào vị trí phần tử đầu tiên. Xem như đây là lần thực hiện thứ nhất. Để đưa giá trị lớn nhất về vị trí đầu tiên, chúng ta cần so sánh phần tử thứ nhất với các phần tử còn lại. Khi phần tử đang được so sánh lớn hơn phần tử đầu tiên thì hai phần tử này cần phải được hoán vị.

Khởi đầu, ở lần thực hiện đầu tiên, phần tử ở vị trí thứ nhất được so sánh với phần tử ở vị trí thứ hai. Hình 12.2 biểu diễn sự hoán vị tại vị trí thứ nhất.



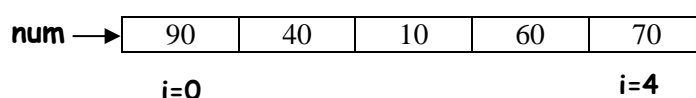
**Hình 12.2: Đảo vị trí phần tử thứ nhất với phần tử thứ hai**

Tiếp đó, phần tử thứ nhất được so sánh với phần tử thứ ba. Hình 12.3 biểu diễn sự hoán vị giữa phần tử thứ nhất và phần tử thứ ba.



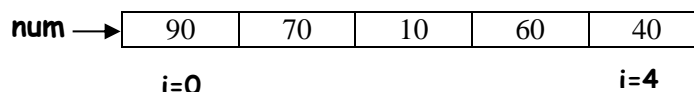
**Hình 12.3 Đảo vị trí phần tử thứ nhất với phần tử thứ ba**

Quá trình này được lặp lại cho đến khi phần tử thứ nhất được so sánh với phần tử cuối cùng của mảng. Mảng kết quả sau lần thực hiện đầu tiên được trình bày trong hình 12.4 bên dưới.



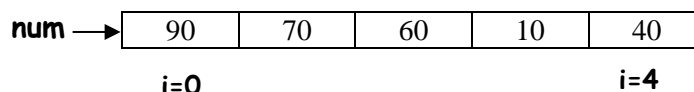
**Hình 12.4: Mảng sau lần thực hiện đầu tiên**

b. Bỏ qua phần tử đầu tiên, chúng ta cần tìm phần tử lớn thứ hai và hoán vị nó với phần tử thứ hai của mảng. Hình 12.5 biểu diễn mảng sau khi được thực hiện lần hai.



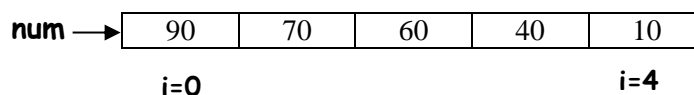
**Hình 12.5: Mảng sau lần thực hiện thứ hai**

c. Phần tử thứ ba phải được hoán vị với phần tử lớn thứ ba của mảng. Hình 12.6 biểu diễn mảng sau khi hoán vị phần tử lớn thứ ba.



**Hình 12.6: Mảng sau lần thực hiện thứ ba**

d. Phần tử thứ tư phải được hoán vị với phần tử lớn thứ tư của mảng. Hình 12.7 biểu diễn mảng sau khi hoán vị phần tử lớn thứ tư.



**Hình 12.7: Mảng sau lần thực hiện thứ tư**

e. Hình 12.7 cũng biểu diễn mảng đã được sắp xếp.

Để lập trình cho bài toán này, chúng ta cần hai vòng lặp, một để tìm phần tử lớn nhất trong mảng và một vòng lặp kia để lặp quá trình thực hiện n lần. Thực chất quá trình phải lặp n-1 lần cho một phần tử của mảng bởi vì phần tử cuối cùng sẽ không còn phần tử nào để so sánh với nó. Vì vậy, chúng ta khai báo hai biến i và j để thao tác với hai vòng lặp **for**. Vòng lặp **for** với chỉ số i được dùng để lặp lại quá trình xác định phần tử lớn nhất trong phần còn lại của mảng. Vòng lặp **for** với chỉ số j được dùng để tìm phần tử lớn thứ i của mảng trong các phần tử từ phần tử thứ i+1 đến phần tử cuối cùng của mảng. Theo cách đó, phần tử lớn thứ nhất thứ i trong phần còn lại của mảng sẽ được đưa vào vị trí thứ i.

Đoạn mã lệnh khai báo chỉ số và vòng lặp thực hiện n - 1 lần với i như là chỉ số:

```
int i, j;
for(i = 0; i < n - 1; i++)
{
```

Đoạn mã lệnh cho vòng lặp từ phần tử thứ i + 1 đến phần tử thứ n của mảng:

```
for(j = i + 1; j < n; j++)
{
```

Để hoán vị hai phần tử trong mảng chúng ta cần sử dụng một biến tạm. Bởi vì đây là thời điểm một phần tử của mảng được sao chép thành một phần tử khác, giá trị trong phần tử thứ hai sẽ bị mất. Để tránh mất giá trị của phần tử thứ hai, giá trị cần phải được lưu lại trong một biến tạm. Đoạn mã lệnh để hoán vị phần tử thứ i với phần tử lớn nhất trong phần còn lại của mảng là:

```
if(desnum[i] < desnum[j])
{
    temp = desnum[i];
    desnum[i] = desnum[j];
    desnum[j] = temp;
}
}
```

Các vòng lặp for cần được đóng lại và vì vậy hai dấu ngoặc đóng xuất hiện trong đoạn mã lệnh trên.

5. Hiện thị mảng đã được sắp xếp.

Chỉ số i có thể được dùng để hiển thị các giá trị của mảng như các câu lệnh trình bày bên dưới:

```
for(i = 0; i < n; i++)
printf("\n Number at [%d] is %d", i, desnum[i]);
```

Theo cách đó các phần tử của một mảng được sắp xếp. Hãy xem chương trình hoàn thiện dưới đây.

**1. Gọi trình soạn thảo mà bạn có thể viết chương trình C.**

**2. Tạo một tập tin mới.**

**3. Đưa vào mã lệnh sau:**

```
void main()
{
    int n;
    int num[100];
    int l;
    int desnum[100], k;
    int i, j, temp;
    printf("\nEnter the total number of marks to be entered : ");
    scanf("%d", &n);
    clrscr();
    for (l = 0; l < n; l++)
    {
        printf("\n Enter the marks of student %d : ", l + 1);
        scanf("%d", &num[l]);
    }
    for(k = 0; k < n; k++)
    desnum[k] = num[k];
    for(i = 0; i < n - 1; i++)
    {
        for(j = i + 1; j < n; j++)
        {
            if(desnum[i] < desnum[j])
            {
                temp = desnum[i];
                desnum[i] = desnum[j];
                desnum[j] = temp;
            }
        }
    }
    for(i = 0; i < n; i++)
        printf("\n Number at [%d] is %d", i, desnum[i]);
}
```

Để xem kết quả, thực hiện theo các bước liệt kê dưới đây:

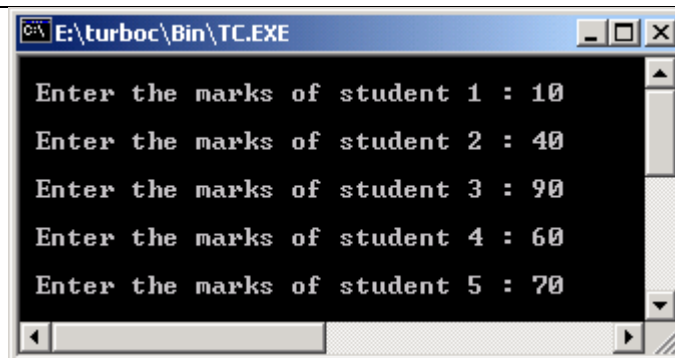
**1. Lưu tập tin với tên *arrayI.C*.**

**2. Biên dịch tập tin, *arrayI.C*.**

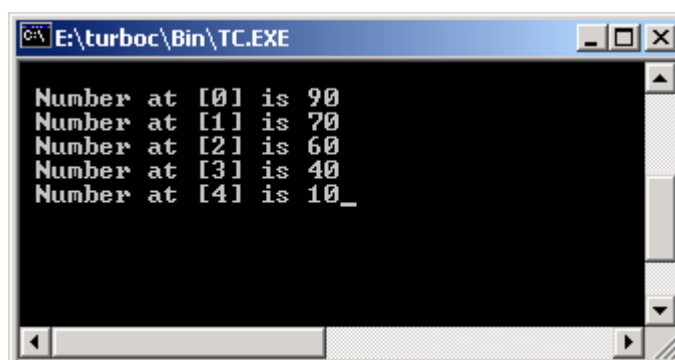
**3. Thực thi chương trình, *arrayI.C*.**

**4. Trở về trình soạn thảo.**

Ví dụ về kết quả thực thi của chương trình trên được trình bày trong hình 12.8 và 12.9.



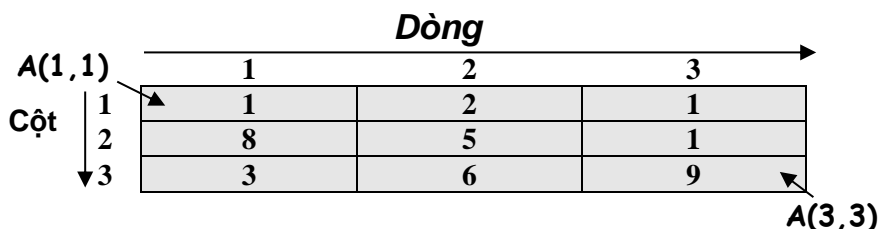
Hình 12.8: Kết quả xuất I của arrayI.C - Nhập vào các giá trị



Hình 12.9 : Kết quả xuất II của arrayI.C – Xuất ra các giá trị

### 12.1.2 Cộng ma trận sử dụng các mảng hai chiều

Các mảng có thể có nhiều chiều. Một ví dụ tiêu biểu của mảng hai chiều là ma trận. Một ma trận được tạo bởi các dòng và các cột. Giao điểm của mỗi dòng và cột có một giá trị. Hình 12.10 biểu diễn một ma trận.



Hình 12.10 : Ma trận A

Số dòng và cột trong ma trận khi được biểu diễn dạng (số dòng) x (số cột) được gọi là kích thước của ma trận. Kích thước của ma trận trong hình 12.10 là 3x3.

Chúng ta hãy xem ví dụ cộng ma trận để hiểu cách sử dụng của mảng hai chiều. Quan sát hai ma trận A và B trong hình 12.11.

<b>A</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>B</b>	<b>1</b>	<b>2</b>	<b>3</b>
<b>1</b>	1	2	1	<b>1</b>	2	1	4
<b>2</b>	8	5	1	<b>2</b>	3	4	2
<b>3</b>	3	6	9	<b>3</b>	8	9	0

Hình 12.11 : Ma trận A và B

Tổng của hai ma trận này là một ma trận khác. Nó được tạo từ việc cộng các giá trị tại mỗi dòng và cột tương ứng. Ví dụ, phần tử đầu tiên C(1,1) trong ma trận C sẽ là tổng của A(1,1) và B(1,1). Phần tử thứ hai C(1,2) sẽ là tổng của A(1,2) và B(1,2) ... Một qui luật quan trọng trong việc cộng các ma trận là kích thước

của các ma trận phải giống nhau. Nghĩa là, một ma trận 2x3 chỉ có thể được cộng với một ma trận 2x3. Hình 12.12 Biểu diễn ma trận A, B và C.

<b>A</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>B</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>C</b>	<b>1</b>	<b>2</b>	<b>3</b>
<b>1</b>	1	2	1	<b>1</b>	2	1	4	<b>1</b>	3	3	5
<b>2</b>	8	5	1	<b>2</b>	3	4	2	<b>2</b>	11	9	3
<b>3</b>	3	6	9	<b>3</b>	8	9	0	<b>3</b>	11	15	9

**Hình 12.12 : Ma trận A, B và C**

Để lập trình công việc này,

1. Khai báo hai mảng. Mã lệnh như sau,

```
int A[10][10], B[10][10], C[10][10];
```

2. Nhập vào kích thước của các ma trận. Mã lệnh là,

```
int row,col;
printf("\n Enter the dimension of the matrix : ");
scanf("%d %d",&row,&col);
```

3. Nhập các giá trị cho ma trận A và B.

Các giá trị của ma trận được nhập theo dòng. Trước tiên các giá trị của dòng thứ nhất được nhập vào. Kế đến các giá trị của dòng thứ hai được nhập, ... Bên trong một dòng, các giá trị của cột được nhập tuần tự. Vì vậy cần hai vòng lặp để nhập các giá trị của một ma trận. Vòng lặp thứ nhất đi qua từng dòng một, trong khi vòng lặp bên trong sẽ đi qua từng cột.

Đoạn mã lệnh như sau:

```
printf("\n Enter the values of the matrix A and B : \n");
int i, j;
for(i = 0; i < row; i++)
    for(j = 0; j < col; j++)
    {
        printf("A[%d,%d], B[%d,%d]:", row, col, row, col);
        scanf("%d %d", &A[i][j], &B[i][j]);
    }
```

4. Cộng hai ma trận. Hai ma trận có thể được cộng bằng cách sử dụng đoạn mã lệnh sau,

```
C[i][j] = A[i][j] + B[i][j];
```

Chú ý, dòng lệnh này cần đặt ở vòng lặp bên trong của đoạn lệnh đã nói ở trên. Một cách khác, hai vòng lặp có thể được viết lại để cộng ma trận.

5. Hiển thị ba ma trận. Mã lệnh sẽ như sau,

```
for(i = 0; i < row; i++)
    for(j = 0; j < col; j++)
    {
        printf("\nA[%d,%d]=%d, B[%d,%d] = %d, C[%d,%d]=%d \n",
            i, j, A[i][j], i, j, B[i][j], i, j, C[i][j]);
    }
```

Bên dưới là chương trình hoàn chỉnh.

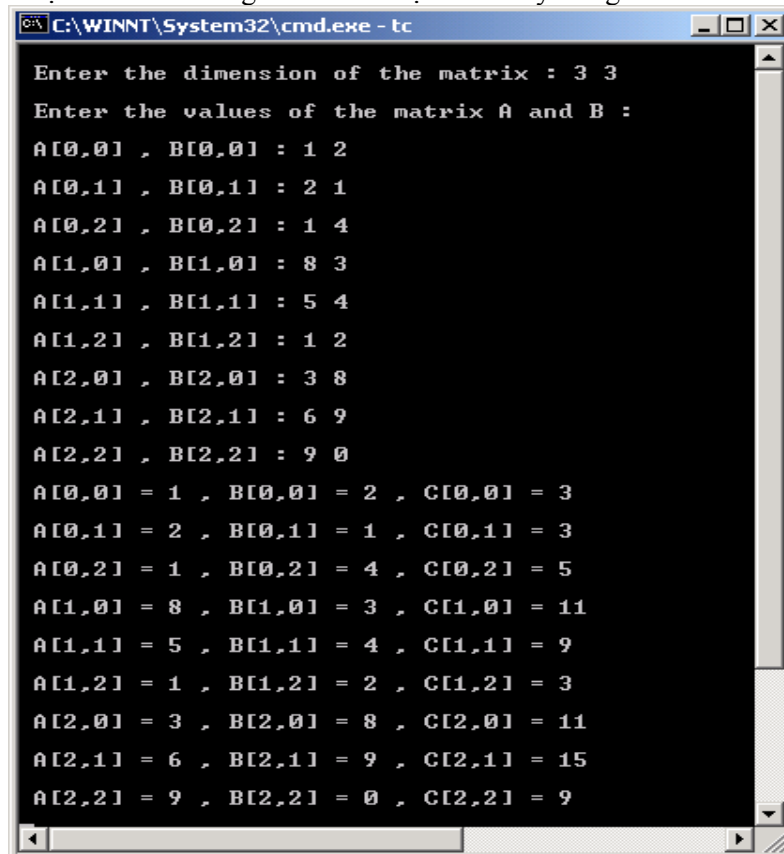
## 1. Tạo một tập tin mới.

## 2. Đưa vào mã lệnh sau:

```
void main()
{
    int A[10][10], B[10][10], C[10][10];
    int row, col;
    int i, j;
    printf("\n Enter the dimension of the matrix : ");
    scanf("%d %d", &row, &col);
    printf("\nEnter the values of the matrix A and B: \n");
    for(i = 0; i < row; i++)
        for(j = 0; j < col; j++)
        {
            printf("\n A[%d,%d], B[%d,%d]: ", i, j, i, j);
            scanf("%d %d", &A[i][j], &B[i][j]);
            C[i][j] = A[i][j] + B[i][j];
        }
    for(i = 0; i < row; i++)
        for(j = 0; j < col; j++)
        {
            printf("\nA[%d,%d]=%d, B[%d,%d]=%d, C[%d,%d]=%d\n",
                i, j, A[i][j], i, j, B[i][j], i, j, C[i][j]);
        }
}
```

3. Lưu tập tin với tên *arrayII.C*.
4. Biên dịch tập tin, *arrayII.C*.
5. Thực thi chương trình *arrayII.C*.
6. Trở về trình soạn thảo.

Một ví dụ về kết quả thực thi của chương trình trên được trình bày trong hình 12.13.



```
C:\WINNT\System32\cmd.exe - tc

Enter the dimension of the matrix : 3 3
Enter the values of the matrix A and B :
A[0,0] , B[0,0] : 1 2
A[0,1] , B[0,1] : 2 1
A[0,2] , B[0,2] : 1 4
A[1,0] , B[1,0] : 8 3
A[1,1] , B[1,1] : 5 4
A[1,2] , B[1,2] : 1 2
A[2,0] , B[2,0] : 3 8
A[2,1] , B[2,1] : 6 9
A[2,2] , B[2,2] : 9 0
A[0,0] = 1 , B[0,0] = 2 , C[0,0] = 3
A[0,1] = 2 , B[0,1] = 1 , C[0,1] = 3
A[0,2] = 1 , B[0,2] = 4 , C[0,2] = 5
A[1,0] = 8 , B[1,0] = 3 , C[1,0] = 11
A[1,1] = 5 , B[1,1] = 4 , C[1,1] = 9
A[1,2] = 1 , B[1,2] = 2 , C[1,2] = 3
A[2,0] = 3 , B[2,0] = 8 , C[2,0] = 11
A[2,1] = 6 , B[2,1] = 9 , C[2,1] = 15
A[2,2] = 9 , B[2,2] = 0 , C[2,2] = 9
```

Hình 12.13 : Kết quả I của arrayII.C – Nhập các giá trị



---

**Phần II – Trong thời gian 30 phút kế tiếp:**

1. Viết một chương trình C nhập một tập hợp số vào trong một mảng và đảo ngược mảng.

Để thực hiện điều này:

- a. Khai báo hai mảng.
  - b. Nhập các giá trị vào một mảng.
  - c. Thực hiện vòng lặp theo thứ tự ngược trên mảng này để sao chép các giá trị vào mảng thứ hai.
- Dùng một chỉ số khác cho mảng thứ hai, chỉ số này chạy từ 0.

---

**Bài tập tự làm**

1. Viết một chương trình C để tìm giá trị nhỏ nhất và giá trị lớn nhất trong một mảng.
2. Viết một chương trình để đếm số lượng nguyên âm và phụ âm trong một từ.

## Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Sử dụng con trỏ
- Sử dụng con trỏ với mảng.

Các bước trong bài học này được trình bày chi tiết, rõ ràng và cẩn thận. Điều này giúp ta hiểu rõ về công cụ lập trình. Thực hiện theo các bước sau thật cẩn thận.

## Phần I – Trong thời gian 1 giờ 30 phút đầu:

### 14.1 Con trỏ

Các biến con trỏ trong C chứa địa chỉ của một biến có bất kỳ kiểu nào. Nghĩa là, các con trỏ có thể là kiểu dữ liệu số nguyên hoặc ký tự. Một biến con trỏ số nguyên sẽ chứa địa chỉ của một biến số nguyên. Một con trỏ ký tự sẽ chứa địa chỉ của một biến kiểu ký tự.

#### 14.1.1 Đếm số nguyên âm trong một chuỗi sử dụng con trỏ

Các con trỏ có thể được sử dụng thay cho các chỉ số duyệt các phần tử trong một mảng. Ví dụ, một con trỏ kiểu chuỗi có thể được dùng để trỏ đến địa chỉ bắt đầu của một từ. Vì vậy một con trỏ được sử dụng để đọc các ký tự trong từ đó. Để minh họa điều này, chúng ta viết một chương trình C để đếm số nguyên âm trong một từ bằng cách sử dụng con trỏ. Các bước được liệt kê như sau:

1. Khai báo một biến con trỏ kiểu ký tự. Mã lệnh như sau,

```
char *ptr;
```

2. Khai báo một mảng ký tự và nhập vào cùng giá trị. Mã lệnh như sau,

```
char word[10];
printf("\n Enter a word : ");
scanf("%s", word);
```

3. Gán con trỏ ký tự tới chuỗi. Mã lệnh như sau,

```
ptr = &word[0];
```

Địa chỉ của ký tự đầu tiên của mảng ký tự, word, sẽ được lưu trong biến con trỏ, ptr. Nói cách khác, con trỏ ptr sẽ trỏ tới ký tự đầu tiên trong mảng ký tự word.

4. Lần lượt duyệt các ký tự trong từ để xác định đó là nguyên âm hay không. Trong trường hợp một nguyên âm được tìm thấy, tăng giá trị biến đếm nguyên âm. Đoạn mã lệnh như sau,

```
int i, vowcnt;
for(i = 0; i < strlen(word); i++)
{
    if((*ptr=='a') || (*ptr=='e') || (*ptr=='i') || (*ptr=='o')
    || (*ptr=='u') || (*ptr=='A') || (*ptr=='E') || (*ptr=='I')
    || (*ptr=='O') || (*ptr=='U'))
        vowcnt++;
    ptr++;
}
```

5. Hiển thị từ và số lượng nguyên âm trong từ. Đoạn mã lệnh sẽ như sau,

```
printf("\n The word is : %s \n The number of vowels in the
word is: %d ", word, vowcnt);
```

Dưới đây là chương trình hoàn chỉnh.

1. **Gọi trình soạn thảo chương trình C.**

2. **Tạo tập tin mới.**

3. **Đưa vào đoạn mã lệnh sau:**

```
void main()
{
    char *ptr;
    char word[10];
    int i, vowcnt=0;
    printf("\n Enter a word: ");
    scanf("%s", word);
    ptr = &word[0];
    for(i = 0; i < strlen(word); i++)
    {
        if((*ptr=='a') || (*ptr=='e') || (*ptr=='i') ||
(*ptr=='o') || (*ptr=='u') || (*ptr=='A') ||
(*ptr=='E') || (*ptr=='I') || (*ptr=='O') ||
(*ptr=='U'))
            vowcnt++;
        ptr++;
    }
    printf("\n The word is: %s \n The number of vowels in
the word is: %d ", word, vowcnt);
}
```

Xem kết quả, theo những bước sau:

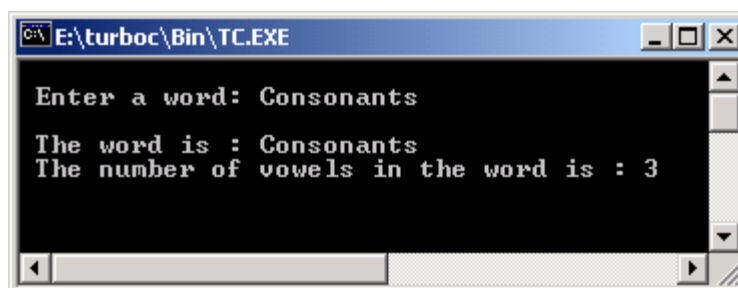
1. **Lưu tập tin với tên pointerI.C.**

2. **Biên dịch tập tin, pointerI.C.**

3. **Chạy chương trình, pointerI.C.**

4. **Trở về trình soạn thảo.**

Kết quả của chương trình trên được thể hiện như trong hình 14.1.



**Hình 14.1 : Kết quả của chương trình pointerI.C**

### 14.1.2 Sắp xếp một mảng theo thứ tự abc sử dụng con trỏ

Các con trỏ có thể được sử dụng để hoán vị nội dung của hai ô nhớ. Để minh họa điều này, chúng ta viết một chương trình C để sắp xếp một tập các chuỗi theo thứ tự abc.

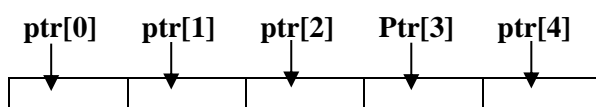
Có nhiều cách giải quyết chương trình này. Chúng ta hãy dùng một mảng của con trỏ ký tự để hiểu cách dùng mảng của con trỏ.

Để thực hiện chương trình này,

1. Khai báo mảng con trỏ ký tự chứa 5 chuỗi. Mã lệnh như sau,

```
char *ptr[5];
```

Mảng được mô tả trong hình 14.2.

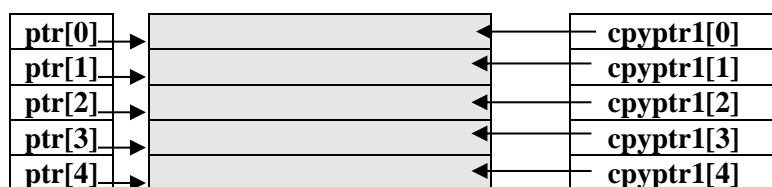


**Hình 14.2: Mảng con trỏ ký tự**

2. Nhập 5 chuỗi và gán các con trỏ trong mảng con trỏ đến các chuỗi. Đoạn mã lệnh như sau,

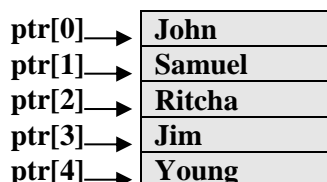
```
int i;
char cpyptr1[5][10];
for (i=0; i<5; i++)
{
    printf("\n Enter a string : ");
    scanf("%s", cpyptr1[i]);
    ptr[i]=cpyptr1[i];
}
```

Mảng được mô tả như hình 14.3.



**Hình 14.3 : Mảng con trỏ ký tự, ptr đang trỏ đến mảng ký tự cpyptr1**

Mỗi chuỗi sẽ được đọc vào bộ nhớ và một biến con trỏ sẽ được gán đến vị trí đó. Mảng được mô tả như trong hình 14.4.



**Hình 14.4 : Mảng sau khi được nhập các giá trị**

3. Lưu trữ mảng của các chuỗi trước khi sắp xếp.

Để thực hiện điều này, chúng ta cần tạo một bản sao của mảng các chuỗi. Đoạn mã lệnh như sau,

```
char cpyptr2[5][10];
for(i = 0; i < 5; i++)
    strcpy(cpyptr2[i], cpyptr1ptr[i]);
```

Ở đây hàm strcpy() được sử dụng để sao chép các chuỗi vào một mảng khác.

4. Sắp xếp mảng các chuỗi theo thứ tự abc. Mã lệnh là,

```
char *temp;
for(i = 0; i < 4; i++)
{
    for(j = i + 1; j < 5; j++)
    {
        if (strcmp(ptr[i], ptr[j]) > 0)
        {
            temp = ptr[i];
            ptr[i] = ptr[j];
            ptr[j] = temp;
        }
    }
}
```

5. Hiển thị các chuỗi ban đầu và các chuỗi đã được sắp xếp. Đoạn mã lệnh là,

```
print("\nThe Original list is ");
for(i = 0; i < 5; i++)
    printf("\n%s", cpyptr2[i]);
printf("\nThe Sorted list is ");
for(i = 0; i < 5; i++)
    printf("\n%s", ptr[i]);
```

Dưới đây là chương trình hoàn thiện.

1. **Tạo tập tin mới.**

2. **Đưa vào đoạn mã lệnh sau:**

```
void main()
{
    char *ptr[5];
    int i;
    int j;
    char cpyptr1[5][10], cpyptr2[5][10];
    char *temp;

    for(i = 0; i < 5; i++)
    {
        printf("\nEnter a string: ");
        scanf("%s", cpyptr1[i]);
        ptr[i] = cpyptr1[i];
    }
    for(i = 0; i < 5; i++)
        strcpy(cpyptr2[i], cpyptr1[i]);
    for(i = 0; i < 4; i++)
```

```

{
    for(j = i + 1; j < 5; j++)
    {
        if (strcmp(ptr[i], ptr[j]) > 0)
        {
            temp = ptr[i];
            ptr[i] = ptr[j];
            ptr[j] = temp;
        }
    }

    printf("\n The Original list is ");
    for(i = 0; i < 5; i++)
        printf("\n%s", cpyptr2[i]);

    printf("\n The Sorted list is ");
    for(i = 0; i < 5; i++)
        printf("\n%s", ptr[i]);
}
}

```

Để xem kết quả, thực hiện theo các bước sau:

1. Lưu tập tin với tên pointII.C
2. Biên dịch tập tin, pointII.C
3. Chạy chương trình, pointII.C
4. Trở về trình soạn thảo.

Kết quả của ví dụ trên được hiển thị ra như trong hình 14.5.

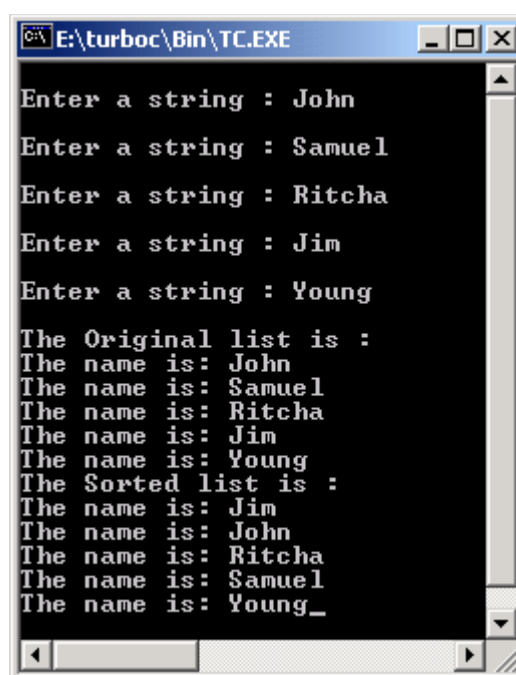


Figure 14.5: Kết quả của chương trình pointII.C

---

**Phần II – Trong thời gian 30 phút kế tiếp:**

1. Viết một chương trình C để nối hai chuỗi bằng cách sử dụng các con trỏ.

Để thực hiện điều này,

- a. Khai báo ba biến chuỗi.
- b. Khai báo ba con trỏ kiểu ký tự.
- c. Nhập các giá trị của hai chuỗi.
- d. Tạo ba con trỏ để trỏ đến ba biến chuỗi. Chuỗi thứ ba hiện tại không có bất kỳ giá trị gì.
- e. Lặp qua chuỗi thứ nhất và sao chép nội dung của chuỗi đó vào chuỗi thứ ba. Sử dụng các biến con trỏ để sao chép các giá trị.
- f. Sau khi sao chép chuỗi thứ nhất, lặp qua chuỗi thứ hai và chép nội dung của chuỗi vào cuối chuỗi ba. Sử dụng các biến con trỏ để sao chép giá trị.
- g. In ra chuỗi thứ ba.

---

**Bài tập tự làm**

1. Viết một chương trình C để đảo một mảng ký tự bằng cách sử dụng con trỏ.
2. Viết một chương trình để cộng hai ma trận sử dụng các con trỏ.

## Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Định nghĩa và gọi hàm
- Sử dụng các tham số trong hàm.

## Phần I – Trong thời gian 1 giờ 30 phút đầu:

### 16.1 Hàm

Như chúng ta đã biết, một hàm là một khối các lệnh thực hiện một tác vụ xác định. Trong bài này, chúng ta tập trung vào cách tạo và sử dụng hàm.

#### 16.1.1 Định nghĩa hàm

Một hàm được định nghĩa với một tên hàm, theo sau bởi cặp dấu ngoặc nhọn {} bên trong chứa một hay nhiều câu lệnh. Ví dụ,

```
argentina()
{
    statement 1;
    statement 2;
    statement 3;
}
```

#### 16.1.2 Gọi một hàm

Một hàm có thể được gọi từ chương trình chính bằng cách đưa ra tên của hàm theo sau bởi cặp dấu ngoặc () và một dấu chấm phẩy. Ví dụ,

```
argentina();
```

Bây giờ, xem chương trình hoàn thiện.

#### 1. Gọi trình soạn thảo chương trình C.

#### 2. Tạo tập tin mới.

#### 3. Đưa vào đoạn mã lệnh sau:

```
#include<stdio.h>
void main()
{
    printf("\nI am in main");
    italy();
    brazil();
    argentina();
}
italy()
{
    printf("\nI am in italy");
}
brazil()
{
    printf("\nI am in brazil");
}
```



```

}
argentina()
{
    printf("\nI am in argentina");
}

```

Để xem kết quả, thực hiện các bước sau:

1. Lưu tập tin với tên **functionI.C**.
2. Biên dịch tập tin, **functionI.C**.
3. Thực thi chương trình, **functionI.C**.
4. Trở về trình soạn thảo.

Kết quả của chương trình trên được minh họa như hình 16.1



Hình 16.1: Kết quả của **functionI.C**

## 1.2 Sử dụng các tham số trong hàm

**Các tham số** được sử dụng để truyền thông tin đến hàm. Các chuỗi định dạng và danh sách các biến được đặt bên trong cặp dấu ngoặc **()** của hàm là các tham số.

### 16.2.1 Định nghĩa một hàm có tham số

Một hàm được định nghĩa với một tên hàm theo sau là dấu ngoặc mở **(**, sau đó là các tham số và cuối cùng là dấu ngoặc đóng **)**. Bên trong hàm, có thể có một hoặc nhiều câu lệnh. Ví dụ,

```

calculatesum (int x, int y, int z)
{
    statement 1;
    statement 2;
    statement 3;
}

```

Xem chương trình hoàn thiện sau.

1. Tạo một tập tin mới.
2. Nhập vào mã lệnh sau:

```

#include<stdio.h>
void main()
{
    int a, b, c, sum;

```

```
printf("\nEnter any three numbers: ");
scanf("%d %d %d", &a, &b, &c);

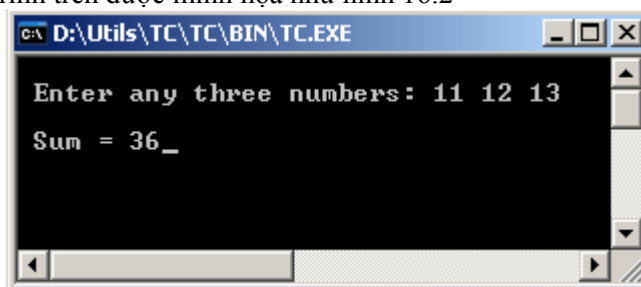
sum = calculatesum(a, b, c);

printf("\nSum = %d", sum);
}
calculatesum(int x, int y, int z)
{
    int d;

    d = x + y + z;
    return (d);
}
```

3. **Lưu tập tin với tên functionII.C.**
4. **Biên dịch tập tin, functionII.C.**
5. **Thực thi chương trình, functionII.C.**
6. **Trở về trình soạn thảo.**

Kết quả của chương trình trên được minh họa như hình 16.2



**Hình 16.2: Kết quả của functionII.C**

## **Phần II – Trong thời gian 30 kế tiếp:**

1. Viết một chương trình C nhập vào một số và tính bình phương của số đó bằng cách sử dụng hàm.

Để thực hiện điều này,

- a. Khai báo một hàm.
- b. Nhập vào một số.
- c. Truyền số đó đến hàm và hàm sẽ trả về bình phương của số đó.

## **Bài tập tự làm**

1. Viết một chương trình C để tính diện tích và chu vi hình tròn.
2. Viết một chương trình in ra giai thừa của một số nguyên.

### Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Sử dụng các hàm về chuỗi
- Truyền mảng vào hàm
- Truyền chuỗi vào hàm.

Các bước trong bài này được giải thích chi tiết, đầy đủ. Mục đích là nắm được những nội dung trong bài lý thuyết và sử dụng thành thạo được công cụ. Hãy làm theo các bước dưới đây một cách cẩn thận.

### Phần I – Trong thời gian 1 giờ 30 phút đầu:

#### 18.1 Các hàm về chuỗi

Các hàm xử lý chuỗi trong C nằm trong thư viện chuẩn **string.h**. Do đó bất kỳ chương trình nào có sử dụng đến các hàm xử lý chuỗi phải đưa thư viện này vào bằng lệnh `include`.

##### 18.1.1 Sắp xếp chuỗi sử dụng các hàm trong thư viện

Các hàm về chuỗi được dùng để thao tác trên các mảng ký tự. Chẳng hạn như, chiều dài của một chuỗi có thể được xác định bằng hàm **strlen()**. Chúng ta hãy viết một chương trình C để sắp xếp 5 chuỗi theo độ dài giảm dần. Các bước thực hiện được liệt kê như sau:

1. Như chúng ta đã học trong bài lý thuyết, trong C, để sử dụng các hàm về chuỗi, chúng ta cần phải đưa vào hai thư viện chuẩn: `stdio.h`, `string.h`. Câu lệnh sẽ là:

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

2. Khai báo một mảng ký tự để lưu 5 chuỗi. Câu lệnh sẽ là:

```
char str_arr[5][20];
```

3. Nhập vào 5 chuỗi trong vòng lặp **for**. Câu lệnh sẽ là:

```
for(i = 0; i < 5; i++)
{
    printf("\nEnter string %d: ", i + 1);
    scanf("%s", str_arr[i]);
}
```

4. So sánh chiều dài của mỗi chuỗi với các chuỗi khác, nếu chiều dài của chuỗi này nhỏ hơn chiều dài của một chuỗi đứng ở vị trí phía sau nó trong mảng, ta sẽ thực hiện đổi chỗ 2 chuỗi đó cho nhau. Câu lệnh sẽ là:

```
for(i = 0; i < 4; i++)
    for(j = i + 1; j < 5; j++)
    {
        if(strlen(str_arr[i]) < strlen(str_arr[j]))
```

```
        {
            strcpy(str, str_arr[i]);
            strcpy(str_arr[i], str_arr[j]);
            strcpy(str_arr[j], str);
        }
    }
```

Chuỗi **str** được sử dụng trong thao tác hoán đổi hai chuỗi.

5. Hiển thị các chuỗi theo thứ tự đã sắp xếp. Câu lệnh sẽ là:

```
printf("\nThe strings in descending order of length are:");
for(i = 0; i < 5; i++)
    printf("\n%s", str_arr[i]);
```

Chúng ta hãy xem chương trình hoàn chỉnh.

- 1. Mở chương trình soạn thảo mà bạn dùng để gõ chương trình C.**
- 2. Tạo một tập tin mới.**
- 3. Gõ vào các dòng lệnh sau đây:**

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

void main()
{
    int i, j;
    char str_arr[5][20], str[20];

    clrscr();
    for(i = 0; i < 5; i++)
    {
        printf("\nEnter string %d: ", i + 1);
        scanf("%s", str_arr[i]);
    }

    for(i = 0; i < 4; i++)
        for(j = i + 1; j < 5; j++)
        {
            if(strlen(str_arr[i]) < strlen(str_arr[j]))
            {
                strcpy(str, str_arr[i]);
                strcpy(str_arr[i], str_arr[j]);
                strcpy(str_arr[j], str);
            }
        }

    printf("\nThe strings in descending order of length are:");

    for(i = 0; i < 5; i++)
        printf("\n%s", str_arr[i]);

    getch();
}
```

Để xem kết quả, thực hiện các bước sau đây:

1. Lưu tập tin với tên stringI.C.
2. Biên dịch tập tin, stringI.C.
3. Thực thi chương trình, stringI.C.
4. Trở về chương trình soạn thảo.

Kết quả của chương trình trên được minh họa như sau:

```
Enter string 1: This
Enter string 2: sentence
Enter string 3: is
Enter string 4: not
Enter string 5: sorted
The strings in descending order of length are:
sentence
sorted
This
not
is
```

### **18.1.2 Sử dụng hàm để chuyển một mảng ký tự về kiểu chữ hoa**

Các chuỗi có thể được truyền vào hàm để thao tác. Khi chuỗi hay mảng các ký tự, được truyền vào hàm, thực ra là ta truyền địa chỉ của nó. Để minh họa điều này, chúng ta hãy viết một chương trình C để chuyển các chuỗi về kiểu chữ hoa. Việc chuyển đổi về kiểu chữ hoa sẽ được thực hiện bằng một hàm.

Các bước được liệt kê như sau:

1. Đưa vào các thư viện cần thiết. Câu lệnh sẽ là:

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

2. Khai báo một mảng để lưu trữ 5 chuỗi. Câu lệnh sẽ là:

```
char names[5][20];
```

3. Khai báo một hàm nhận vào một chuỗi như là một đối số. Câu lệnh sẽ là:

```
void upername(char name_arr[]);
```

4. Nhập 5 chuỗi đưa vào mảng. Câu lệnh sẽ là:

```
for(i = 0; i < 5; i++)
{
    printf("\nEnter string %d: ", i + 1);
    scanf("%s", names[i]);
}
```

5. Truyền mỗi chuỗi vào hàm để chuyển thành in hoa. Sau khi chuyển đổi, hiển thị chuỗi đã thay đổi. Câu lệnh sẽ là:

```
for(i = 0; i < 5; i++)
{
    upername(names[i]);
    printf("\nNew string %d: %s", i + 1, names[i]);
}
```

## 6. Định nghĩa hàm. Câu lệnh sẽ là:

```
void upername(char name_arr[])
{
    int x;

    for(x = 0; name_arr[x] != '\0'; x++)
    {
        if(name_arr[x] >= 97 && name_arr[x] <= 122)
            name_arr[x] = name_arr[x] - 32;
    }
}
```

Câu lệnh điều kiện bên trong vòng lặp kiểm tra giá trị ASCII của từng kí tự trong chuỗi. Nếu kí tự ở dạng chữ thường, nó sẽ được chuyển về dạng chữ hoa. Lưu ý rằng giá trị ASCII của 'A' là 65 và 'a' là 97.

Chúng ta hãy xem chương trình hoàn chỉnh.

### 1. Tạo một tập tin mới.

### 2. Gõ vào các dòng lệnh sau đây:

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

void main()
{
    int i;
    char names[5][20];
    void upername(char name_arr[]);
    clrscr();
    for(i = 0; i < 5; i++)
    {
        printf("\nEnter string %d: ", i + 1);
        scanf("%s", names[i]);
    }
    for(i = 0; i < 5; i++)
    {
        upername(names[i]);
        printf("\nNew string %d: %s", i + 1, names[i]);
    }
    getch();
}

void upername(char name_arr[])
{
    int x;

    for(x = 0; name_arr[x] != '\0'; x++)
    {
        if(name_arr[x] >= 97 && name_arr[x] <= 122)
            name_arr[x] = name_arr[x] - 32;
    }
}
```

Để xem kết quả, thực hiện các bước sau đây:

3. Lưu tập tin với tên stringII.C.
4. Biên dịch tập tin stringII.C.
5. Thực thi chương trình stringII.C.
6. Trở về chương trình soạn thảo.

Kết quả của chương trình trên được minh họa như sau:

```
Enter string 1: Sharon
Enter string 2: Christina
Enter string 3: Joanne
Enter string 4: Joel
Enter string 5: Joshua
New string 1: SHARON
New string 2: CHRISTINA
New string 3: JOANNE
New string 4: JOEL
New string 5: JOSHUA
```

## Phần II – Trong thời gian 30 phút kế tiếp:

1. Viết một chương trình C để hiển thị số lần xuất hiện của một ký tự nào đó trong một chuỗi. Dùng một vòng lặp để thực hiện thao tác này 5 lần.

Để làm điều này,

- a. Khai báo một biến ký tự và một mảng ký tự.
- b. Khai báo một hàm để nhận vào một mảng ký tự và một biến ký tự, và trả về một giá trị nguyên.
- c. Dùng một vòng lặp để nhập vào một chuỗi và một ký tự 5 lần.
- d. Nhận vào một chuỗi và một ký tự.
- e. Truyền chuỗi và ký tự vào hàm và nhận giá trị trả về bằng một biến nguyên.
- f. In giá trị trả về.
- g. Hàm trên thực hiện so sánh từng ký tự trong chuỗi với ký tự cần tìm. Tăng biến đếm lên một mỗi khi tìm thấy ký tự đó trong chuỗi. Cuối cùng, trả về giá trị của biến đếm cho hàm main().

## Bài tập tự làm

1. Viết một chương trình C để nhập vào 5 tên và một chức danh. Chèn chức danh đó vào phần đầu mỗi tên trong mảng. Hiển thị các tên đã sửa đổi.
2. Viết một chương trình C nhập vào nhiệt độ trung bình hằng năm của 5 năm qua cho 5 thành phố. Hiển thị nhiệt độ lớn nhất và nhỏ nhất của mỗi thành phố. Sử dụng hàm để xác định các nhiệt độ lớn nhất và nhỏ nhất.

### Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Sử dụng cấu trúc và các mảng cấu trúc
- Truyền tham số cấu trúc vào hàm
- Sắp xếp mảng.

Các bước trong bài này được giải thích chi tiết, đầy đủ. Mục đích là nắm được những nội dung trong bài lý thuyết và sử dụng thành thạo được công cụ. Hãy làm theo các bước dưới đây một cách cẩn thận.

### Phần I – Trong thời gian 1 giờ 30 phút đầu:

#### 20.1 Cấu trúc

Một cấu trúc là một nhóm các mẫu dữ liệu có thể có kiểu khác nhau. Mỗi cấu trúc phải được định nghĩa trước khi nó được sử dụng trong khai báo biến. Một định nghĩa cấu trúc có thể bao gồm một thành phần là một cấu trúc khác. Việc khởi tạo cấu trúc tương tự như việc khởi tạo mảng.

##### 20.1.1 Các mảng cấu trúc và sắp xếp

Trong C, có thể tạo mảng cấu trúc. Cũng như với mảng, dữ liệu trong các mảng cấu trúc có thể được sắp xếp theo hai phương pháp Selection sort và Bubble sort. Chúng ta hãy viết một chương trình C để cài đặt một hệ thống quản lý thư viện cơ bản. Hệ thống lưu trữ danh mục sách và ghi nhận các giao dịch mượn và trả sách. Sử dụng hệ thống này, ta có thể thêm thông tin chi tiết của một sách, ghi nhận các giao dịch mượn/trả sách và sắp xếp các ghi nhận này. Các bước để tạo hệ thống được liệt kê như sau:

1. Định nghĩa một cấu trúc để lưu trữ chi tiết sách. Câu lệnh sẽ là:

```
struct book_st{
    int book_cd;
    char book_nm[30];
    char author[30];
    int copies;
};
```

2. Định nghĩa một cấu trúc để lưu trữ các giao dịch mượn/trả sách. Lưu ý rằng ngày mượn/trả cũng sẽ là một cấu trúc và cũng phải được định nghĩa. Câu lệnh sẽ là:

```
struct date_st { int month;
                int day;
                int year; };

struct tran_st { int book_code;
                char tran_type;
                struct date_st tran_dt; };
```

3. Khai báo các biến có hai kiểu cấu trúc trên. Để thực hành, ta giả sử rằng cần lưu trữ chi tiết của 5 quyển sách và 10 giao dịch. Câu lệnh sẽ là:

```
struct book_st books[5];
struct tran_st trans[10];
```



4. Viết một vòng lặp để hiển thị menu các thao tác mà chương trình có thể thực hiện.. Câu lệnh sẽ là:

```
while(choice != 4)
{
    clrscr();
    printf("\nSelect from Menu\n1. Add book names\n2.
        Record Issue/Return\n3. Sort Transactions\n4.
        Exit\n\nEnter choice: ");
    scanf("%d", &choice);
    .
    .
    .
}
```

5. Nếu thao tác được chọn là thêm sách, thì nhập các thông tin chi tiết của sách trong một vòng lặp. Câu lệnh sẽ là:

```
for(i = 0; i < 5 && addflag == 'y'; i++)
{
    books[i].book_cd = i + 1;
    printf("\n\nBook code: %d\n\nBook name: ", i + 1);
    scanf("%s", books[i].book_nm);
    printf("\nAuthor: ");
    scanf("%s", books[i].author);
    printf("\nNumber of copies: ");
    scanf("%d", &books[i].copies);
    printf("\n\nContinue? (y/n): ");
    scanf(" %c", &addflag);
}
```

6. Nếu thao tác được chọn là thêm giao dịch, đặt một vòng lặp để nhập các thông tin chi tiết của giao dịch. Câu lệnh sẽ là:

```
for(i = 0; i < 10 && addflag == 'y'; i++)
{
    printf("\n\nBook code: ");
    scanf("%d", &trans[i].book_code);
    printf("\nIssue or Return?(I/R): ");
    scanf(" %c", &trans[i].tran_type);
    printf("\nDate: ");
    scanf("%d %d %d",
        &trans[i].tran_dt.month, &trans[i].tran_dt.day,
        &trans[i].tran_dt.year);
    printf("\n\nContinue? (y/n): ");
    scanf("%c", &addflag);
}
```

7. Nếu thao tác được chọn là sắp xếp các giao dịch, thì truyền tham số mảng cấu trúc vào hàm. Hàm sẽ sắp xếp mảng theo mã sách sử dụng phương pháp bubble sort. Câu lệnh sẽ là:

```
for(i = 0; i < 10; i++)
    for(j = i + 1; j < 10; j++)
    {
        if(trans[i].book_code > trans[j].book_code)
        {
            tempran=trans[i];
            trans[i]=trans[j];
            trans[j]=tempran;
        }
    }
```

8. Hiển thị số giao dịch cho mỗi quyển sách trong hàm sắp xếp. Câu lệnh sẽ là:

```
for(i = 0, j = 0; i < 10; j = 0)
{
    tempcode = tran[i].book_code;
    while(tran[i].book_code == tempcode && i < 10)
    {
        j++;
        i++;
    }
    printf("\nBook code %d had %d transactions",
        tempcode, j);
}
```

Tiến hành thực hiện các bước sau đây để có chương trình hoàn chỉnh

1. **Mở chương trình soạn thảo mà bạn dùng để gõ chương trình C.**

2. **Tạo một tập tin mới**

3. **Gõ vào các dòng lệnh sau đây:**

```
#include<stdio.h>

struct book_st {    int book_cd;
                   char book_nm[30];
                   char author[30];
                   int copies; };

struct date_st {    int month;
                   int day;
                   int year; };

struct tran_st {    int book_code;
                   char tran_type;
                   struct date_st tran_dt; };

void main()
{
    int choice = 1, i;
    char addflag;
    struct book_st books[5];
    struct tran_st trans[10];
    while(choice != 4)
    {
        clrscr();
        printf("\nSelect from Menu\n1. Add book names\n2.
        Record Issue/Return\n3. Sort Transactions\n4.
        Exit\n\nEnter choice: ");
        scanf("%d", &choice);

        if(choice == 1)
        {
            addflag = 'y';
            clrscr();
            for(i = 0; i < 5 && addflag == 'y'; i++)
            {
                books[i].book_cd = i + 1;
                printf("\n\nBook code: %d\n\nBook name:",
                    i+1);
                scanf("%s", books[i].book_nm);
                printf("\nAuthor: ");
                scanf("%s", books[i].author);
            }
        }
    }
}
```

```

        printf("\nNumber of copies: ");
        scanf("%d", &books[i].copies);
        printf("\n\nContinue? (y/n): ");
        scanf(" %c", &addflag);
    }
}
else if(choice == 2)
{
    addflag = 'y';

    clrscr();
    for(i = 0; i < 10 && addflag == 'y'; i++)
    {
        printf("\n\nBook code: ");
        scanf("%d", &trans[i].book_code);
        printf("\nIssue or Return?(I/R): ");
        scanf(" %c", &trans[i].tran_type);
        printf("\nDate: ");
        scanf("%d %d %d",
                &trans[i].tran_dt.month,
                &trans[i].tran_dt.day,
                &trans[i].tran_dt.year);
        printf("\n\nContinue? (y/n): ");
        scanf(" %c", &addflag);
    }
}
else if(choice == 3)
{
    sorttran(trans);
}

}
}
sorttran(struct tran_st tran[10])
{
    int i, j, tempcode;
    struct tran_st temptran;

    clrscr();
    for(i = 0; i < 10; i++)
        for(j = i + 1; j < 10; j++)
        {
            if(tran[i].book_code > tran[j].book_code)
            {
                temptran = tran[i];
                tran[i] = tran[j];
                tran[j] = temptran;
            }
        }
    for(i = 0, j = 0; i < 10; j = 0)
    {
        tempcode = tran[i].book_code;
        while(tran[i].book_code == tempcode && i < 10)
        {
            j++;
            i++;
        }
        printf("\nBook code %d had %d transactions",
                tempcode, j);
    }
    getch();
}

```

Để xem kết quả, thực hiện các bước sau đây:

1. **Lưu tập tin với tên structI.C.**
2. **Biên dịch tập tin, structI.C.**
3. **Thực thi chương trình, structI.C.**
4. **Trở về chương trình soạn thảo.**

Mẫu kết quả của chương trình như sau:

Select from Menu

1. Add book names
2. Record Issue/Return
3. Sort Transactions
4. Exit

Enter choice:

Nếu nhập vào 1, mẫu kết xuất của chương trình sẽ là:

Book code: 1  
Book name: Detective  
Author: Hailey  
Number of copies: 3

Continue? (y/n): y

Nếu nhập vào 2, mẫu kết xuất của chương trình sẽ là:

Book code: 1  
Issue or Return? (I/R): I  
Date: 2 22 03

Continue? (y/n): y

Nếu nhập vào 3, mẫu kết xuất của chương trình sẽ là:

Book code 1 had 3 transactions  
Book code 2 had 1 transactions  
Book code 3 had 2 transactions  
Book code 4 had 0 transactions  
Book code 5 had 4 transactions

## Phần II – Trong thời gian 30 phút kế tiếp:

- Viết một chương trình C để lưu trữ các thông tin về sinh viên trong một cấu trúc. Dữ liệu phải bao gồm mã sinh viên, tên sinh viên, khóa học đã đăng ký và năm đăng ký. Viết một hàm để hiển thị các thông tin chi tiết của các sinh viên đã nhập học trong một năm học nào đó. Viết một hàm khác để xác định và hiển thị thông tin chi tiết của một sinh viên khi biết mã của sinh viên đó.

Yêu cầu:

- Định nghĩa một cấu trúc để lưu trữ thông tin chi tiết của sinh viên.
- Khai báo và khởi tạo biến cấu trúc với thông tin chi tiết của 10 sinh viên.
- Viết vòng lặp để hiển thị một menu các thao tác mà chương trình có thể thực hiện.
- Nhận vào lựa chọn danh mục và gọi hàm thích hợp với tham số là mảng cấu trúc.
- Trong hàm dùng để hiển thị thông tin chi tiết của các sinh viên nhập học trong một năm, viết chương trình để thực hiện nhập vào năm học cần được hiển thị thông tin, sau đó sử dụng vòng lặp để kiểm tra năm nhập học của từng sinh viên, nếu trùng với năm cần hiển thị thông tin yêu cầu thì hiển thị thông tin của sinh viên đó. Ngoài ra, hàm này còn cho phép người dùng có thể tiếp tục thực hiện việc hiển thị thông tin của những năm khác cho đến khi họ không muốn sử dụng chức năng này nữa.
- Hàm dùng để hiển thị thông tin chi tiết của sinh viên cho phép nhập vào mã của sinh viên, dùng một vòng lặp để kiểm tra mã của mỗi sinh viên, nếu mã của sinh viên nào trùng với mã đã được nhập thì hiển thị thông tin chi tiết của sinh viên đó. Ngoài ra, hàm này còn cho phép người dùng có thể tiếp tục thực hiện việc hiển thị thông tin của những sinh viên khác cho đến khi họ không muốn sử dụng chức năng này nữa.

## Bài tập tự làm

- Viết một chương trình C để lưu trữ 5 độ dài trong một mảng cấu trúc. Mỗi độ dài phải bao gồm 3 thông tin về yards, feet và inches. Sắp xếp và hiển thị các độ dài.
- Viết một chương trình C để lưu trữ thông tin chi tiết của nhân viên trong một mảng cấu trúc. Thông tin của một nhân viên phải bao gồm mã nhân viên, tên, lương và ngày vào làm. Ngày vào làm phải được lưu trong một cấu trúc khác. Chương trình phải thực hiện các thao tác sau đây dựa trên sự lựa chọn trong menu các chức năng của chương trình:

- Tăng lương theo các luật sau:

Salary Range	Percentage increase
$\leq 2000$	15%
$> 2000$ and $\leq 5000$	10%
$> 5000$	No increase

- Hiển thị thông tin chi tiết của các nhân viên đã làm việc trong công ty từ 10 năm trở lên.

### Mục tiêu:

Kết thúc bài học này, bạn có thể:

- Thực hiện các thao tác trên tập tin văn bản và tập tin nhị phân
- Mở và đóng tập tin
- Đọc từ tập tin và ghi vào tập tin
- Sử dụng con trỏ tập tin.

Các bước được cho trong bài này được giải thích cặn kẽ, dễ hiểu và tư duy cẩn thận từ đầu đến cuối. Bài đã được viết để đáp ứng được mục tiêu học và để có thể hiểu hoàn toàn về công cụ. Xin hãy thực hiện theo các bước một cách cẩn thận.

### Phần I – Trong thời gian 1 giờ 30 phút đầu:

#### 20.2 Quản lý tập tin trong C

C cung cấp một giao diện đồng nhất cho việc quản lý nhập và xuất. Các phương pháp truy cập tập tin cũng giống như các phương pháp quản lý các thiết bị khác. Giải pháp cho tính đồng nhất này là trong C không có kiểu tập tin. C xem tất cả các tập tin là stream.

##### 22.1.1 Đọc, ghi và truy cập dữ liệu trong tập tin

Có một số hàm xử lý tập tin trong tập tin header `stdio.h`. Chúng ta hãy viết một chương trình C sử dụng những hàm này. Chương trình tạo một hệ thống ngân hàng đơn giản. Các chi tiết khách hàng được nhập vào và lưu trong một tập tin gọi là **customer**. Chi tiết của các giao dịch như gửi tiền và rút tiền được kiểm tra hợp lệ trên tập tin **customer**. Các giao dịch hợp lệ được ghi nhận trong tập tin **trans**. Một báo cáo về các khách hàng có số vốn thấp được in ra. Các bước được liệt kê như sau:

1. Định nghĩa một structure để lưu trữ dữ liệu về khách hàng và giao dịch. Câu lệnh sẽ là:

```
struct cust_st
{
    int acc_no;
    char cust_nm[30];
    float bal;
};

struct tran_st
{
    int acc_no;
    char trantype;
    float amt;
};
```

2. Hiện thị một danh mục để thực hiện các thao tác khác nhau dựa trên lựa chọn của người dùng. Câu lệnh sẽ là:

```
while(choice != 4)
{
    clrscr();
    printf("\nSelect choice from menu\n\n1. Accept
customer details\n2. Record Withdrawal/Deposit
transaction\n3. Print Low Balance Report\n4.
Exit\n\nEnter choice: ");
    scanf(" %d", &choice);
```

```
        .  
        .  
    }
```

3. Gọi các hàm tương ứng dựa vào lựa chọn của người dùng. Câu lệnh sẽ là:

```
if(choice == 1)  
    addcust();  
else if(choice == 2)  
    rectran();  
else if(choice == 3)  
    prnlowbal();
```

4. Trong hàm thêm chi tiết của khách hàng, định nghĩa một con trỏ tập tin để kết hợp với tập tin **customer**. Khai báo một biến cấu trúc để nhập dữ liệu của khách hàng. Câu lệnh sẽ là:

```
FILE *fp;  
struct cust_st custdata;
```

5. Mở tập tin **customer** theo chế độ append để có thể thêm các mẫu tin mới. Xác nhận rằng thao tác mở tập tin đã được thực hiện. Câu lệnh sẽ là:

```
if((fp = fopen("customer", "a+")) == NULL)  
{  
    printf("\nERROR opening customer file");  
    getch();  
    return;  
}
```

6. Nhập dữ liệu khách hàng vào biến cấu trúc và ghi dữ liệu vào tập tin **customer**. Câu lệnh sẽ là:

```
fwrite(&custdata, sizeof(struct cust_st), 1, fp);
```

7. Đóng tập tin **customer** sau khi nhập dữ liệu. Câu lệnh sẽ là:

```
fclose(fp);
```

8. Trong hàm dùng để ghi các giao dịch, định nghĩa biến con trỏ để trỏ đến tập tin **customer** và tập tin **trans**. Và định nghĩa biến cấu trúc để nhập vào dữ liệu của giao dịch và đọc dữ liệu khách hàng. Câu lệnh sẽ là:

```
FILE *fp1, *fp2;  
struct cust_st custdata;  
struct tran_st trandata;
```

9. Mở hai tập tin theo chế độ thích hợp. Tập tin **customer** phải mở để đọc và cập nhật, trong khi tập tin **trans** phải cho phép thêm các mẫu tin mới. Câu lệnh sẽ là:

```
if((fp1=fopen("customer", "r+w"))==NULL)  
{  
    printf("\nERROR opening customer file");  
    getch();  
    return;  
}  
  
if((fp2 = fopen("trans", "a+")) == NULL)  
{  
    printf("\nERROR opening transaction file");  
    getch();  
    return;  
}
```

10. Nhập vào số tài khoản cho giao dịch và bảo đảm rằng nó tồn tại trong tập tin customer. Câu lệnh sẽ là:

```
while((fread(&custdata, size, 1, fp1)) == 1 && found == 'n')
{
    if(custdata.acc_no == trandata.acc_no)
    {
        found='y';
        break;
    }
}
```

11. Để bảo đảm nhập vào một kiểu giao dịch hợp lệ, câu lệnh sẽ là:

```
if(trandata.trantype!='D' && trandata.trantype!='d'
    && trandata.trantype!='W' && trandata.trantype!='w')
    printf("\t\tInvalid transaction type, please reenter");
```

12. Đối với các giao dịch rút tiền, phải bảo đảm rằng số tiền rút ra phải sẵn có trong tài khoản của khách hàng. Nếu sẵn có, cập nhật số tiền còn lại trong tài khoản. Cũng cần cập nhật số tiền trong tài khoản cho các giao dịch gửi tiền. Câu lệnh sẽ là:

```
if(trandata.trantype=='W' || trandata.trantype=='w')
{
    if(trandata.amt>custdata.bal)
        printf("\nAccount balance is %.2f. Please
reenter withdrawal amount.", custdata.bal);
    else
    {
        custdata.bal-=trandata.amt;
        .
        .
    }
}
else
{
    custdata.bal+=trandata.amt;
    .
    .
}
```

13. Ghi mẫu tin chứa giao dịch mới vào tập tin trans và cập nhật mẫu tin của khách hàng trong tập tin customer. Câu lệnh sẽ là:

```
fwrite(&trandata, sizeof(struct tran_st), 1, fp2);
fseek(fp1, (long)(-size), 1);
fwrite(&custdata, size, 1, fp1);
```

Lưu ý rằng trong suốt quá trình kiểm tra số tài khoản của khách hàng, mẫu tin đọc cuối cùng là của khách hàng đang thực hiện giao dịch. Vì vậy, con trỏ tập tin của tập tin **customer** phải nằm ở cuối của mẫu tin cần cập nhật. Con trỏ tập tin sẽ được đặt lại vị trí về đầu của mẫu tin sử dụng hàm `fseek()`. Ở đây **size** là một biến số nguyên chứa kích cỡ của cấu trúc cho dữ liệu khách hàng.

14. Đóng hai tập tin sau khi đã nhập giao dịch. Câu lệnh sẽ là:

```
fclose(fp1);
```



```
fclose(fp2);
```

15. Trong hàm hiển thị các tài khoản có số vốn ít, định nghĩa con trỏ tập tin để kết hợp với tập tin **customer**. Khai báo một biến cấu trúc để đọc dữ liệu của khách hàng. Câu lệnh sẽ là:

```
FILE *fp;
struct cust_st custdata;
```

16. Sau khi mở tập tin ở chế độ đọc, đọc mỗi mẫu tin khách hàng và kiểm tra số vốn. Nếu nó ít hơn 250, in mẫu tin ra. Câu lệnh sẽ là:

```
while((fread(&custdata, sizeof(struct cust_st), 1, fp))==1)
{
    if(custdata.bal<250)
    {
        .
        .
        printf("\n%d\t%s\t%.2f", custdata.acc_no,
            custdata.cust_nm, custdata.bal);
    }
}
```

17. Đóng tập tin customer. Câu lệnh sẽ là:

```
fclose(fp);
```

Chúng ta hãy nhìn vào chương trình hoàn chỉnh.

1. **Mở chương trình soạn thảo mà bạn dùng để gõ chương trình C.**

2. **Tạo một tập tin mới.**

3. **Gõ vào các dòng lệnh sau đây:**

```
#include<stdio.h>
struct cust_st
{
    int acc_no;
    char cust_nm[30];
    float bal;
};
struct tran_st
{
    int acc_no;
    char tran_type;
    float amt;
};
void main()
{
    int choice = 1;

    while(choice != 4)
    {
        clrscr();
        printf("\nSelect choice from menu\n\n1. Accept
            customer details\n2. Record Withdrawal/Deposit
            transaction\n3. Print Low Balance Report\n4.
            Exit\n\nEnter choice: ");
        scanf(" %d", &choice);
        if(choice == 1)
            addcust();
        else if(choice == 2)
```

```

        rectran();
    else if(choice == 3)
        prnlowbal();
    }
}
addcust()
{
    FILE *fp;
    char flag = 'y';
    struct cust_st custdata;
    clrscr();

    if((fp = fopen("customer", "a+")) == NULL)
    {
        printf("\nERROR opening customer file");
        getch();
        return;
    }
    while(flag == 'y')
    {
        printf("\n\nEnter Account number: ");
        scanf(" %d", &custdata.acc_no);
        printf("\nEnter Customer Name: ");
        scanf("%s", custdata.cust_nm);
        printf("\nEnter Account Balance: ");
        scanf(" %f", &custdata.bal);
        fwrite(&custdata, sizeof(struct cust_st), 1, fp);
        printf("\n\nAdd another? (y/n): ");
        scanf(" %c", &flag);
    }
    fclose(fp);
}
rectran()
{
    FILE *fp1, *fp2;
    char flag = 'y', found, val_flag;
    struct cust_st custdata;
    struct tran_st trandata;
    int size = sizeof(struct cust_st);
    clrscr();
    if((fp1 = fopen("customer", "r+w")) == NULL)
    {
        printf("\nERROR opening customer file");
        getch();
        return;
    }
    if((fp2 = fopen("trans", "a+")) == NULL)
    {
        printf("\nERROR opening transaction file");
        getch();
        return;
    }
    while(flag == 'y')
    {
        printf("\n\nEnter Account number: ");
        scanf("%d", &trandata.acc_no);
        found='n';
        val_flag = 'n';
        rewind(fp1);
        while((fread(&custdata, size, 1, fp1))==1 &&
                found=='n')
        {
            if(custdata.acc_no == trandata.acc_no)
            {
                found = 'y';
            }
        }
    }
}

```

```

        break;
    }
}
if(found == 'y')
{
    while(val_flag == 'n')
    {
        printf("\nEnter Transaction type (D/W): ");
        scanf(" %c", &trandata.trantype);
        if(trandata.trantype!='D' &&
           trandata.trantype!='d' &&
           trandata.trantype!='W' &&
           trandata.trantype!='w')
            printf("\t\tInvalid transaction type,
                    please reenter");
        else
            val_flag = 'y';
    }
    val_flag = 'n';
    while(val_flag == 'n')
    {
        printf("\nEnter amount: ");
        scanf(" %f", &trandata.amt);
        if(trandata.trantype=='W' ||
           trandata.trantype=='w')
        {
            if(trandata.amt > custdata.bal)
                printf("\nAccount balance is
%.2f. Please reenter withdrawal amount.", custdata.bal);
            else
            {
                custdata.bal -= trandata.amt;
                val_flag = 'y';
            }
        }
        else
        {
            custdata.bal += trandata.amt;
            val_flag = 'y';
        }
    }
    fwrite(&trandata, sizeof(struct tran_st), 1, fp2);
    fseek(fp1, (long)(-size), 1);
    fwrite(&custdata, size, 1, fp1);
}
else
    printf("\nThis account number does not exist");

    printf("\nRecord another transaction? (y/n): ");
    scanf(" %c", &flag);
}
fclose(fp1);
fclose(fp2);
}
prnlowbal()
{
    FILE *fp;
    struct cust_st custdata;
    char flag = 'n';
    clrscr();
    if((fp = fopen("customer", "r")) == NULL)
    {
        printf("\nERROR opening customer file");
    }
}

```

```

        getch();
        return;
    }
    printf("\nReport on account balances below 250\n\n");
    while((fread(&custdata, sizeof(struct cust_st), 1, fp)) == 1)
    {
        if(custdata.bal < 250)
        {
            flag = 'y';
            printf("\n%d\t%s\t%.2f", custdata.acc_no,
                custdata.cust_nm, custdata.bal);
        }
    }
    if(flag == 'n')
        printf("\nNo account balances found below 250");
    getch();
    fclose(fp);
}

```

Để xem kết quả, thực hiện các bước sau đây:

1. **Lưu tập tin với tên filesI.C.**
2. **Biên dịch tập tin, filesI.C.**
3. **Thực thi chương trình, filesI.C.**
4. **Trở về chương trình soạn thảo.**

Kết xuất của chương trình như sau:

Select choice from menu

1. Accept customer details
2. Record Withdrawal/Deposit transaction
3. Print Low Balance Report
4. Exit

Enter choice:

Một mẫu kết xuất của hàm thêm vào chi tiết của khách hàng như sau:

Enter Account number: 123

Enter Customer Name: E.Wilson

Enter Account Balance: 2000

Add another? (y/n):

Một mẫu kết xuất của hàm thêm vào chi tiết của giao dịch như sau:

Enter Account number: 123

Enter Transaction type (D/W): W

Enter amount: 1000

Record another transaction? (y/n):

---

Một mẫu kết xuất của hàm hiển thị báo cáo các tài khoản có vốn thấp như sau:

Report on account balances below 250

104	Jones	200
113	Sharon	150
120	Paula	200

## Phần II – Trong thời gian 30 phút kế tiếp:

1. Viết một chương trình C để hiển thị sự khác nhau giữa hai tập tin nhập vào như là đối số của dòng lệnh. Với mỗi sự khác nhau, hiển thị vị trí tìm thấy sự khác nhau và các ký tự của hai tập tin tại vị trí đó. Cũng cần phải bảo đảm rằng người sử dụng đã nhập vào số lượng đối số hợp lệ. Cuối cùng, hiển thị tổng số sự khác nhau đã tìm thấy.

Để làm điều này,

- a. Khai báo các biến argv và argc để nhận vào đối số từ dòng lệnh.
- b. Khai báo con trỏ trỏ đến hai tập tin.
- c. Kiểm tra tính hợp lệ của argc để bảo đảm rằng đã nhập đúng số đối số.
- d. Mở hai tập tin ở chế độ đọc.
- e. Đặt một vòng lặp để đọc từng ký tự từ hai tập tin cho đến khi đến cuối cả hai tập tin.
- f. Nếu các ký tự là khác nhau, hiển thị chúng cùng với vị trí của chúng. Tăng số đếm sự khác nhau lên 1.
- g. Nếu đi đến cuối của một tập tin, in các ký tự còn lại trong tập tin kia như là sự khác biệt.
- h. Kiểm tra số đếm sự khác nhau để hiển thị các thông báo thích hợp.
- i. Đóng hai tập tin.

## Bài tập tự làm

---

1. Viết một chương trình C để sao chép nội dung của một tập tin vào một tập tin khác loại trừ các từ **a**, **an**, và **the**.
2. Viết một chương trình C để nhập vào hai chuỗi số. Lưu trữ mỗi chuỗi ở hai tập tin riêng biệt. Sắp xếp chuỗi trong mỗi tập tin. Trộn hai chuỗi vào một, sắp xếp và lưu lại chuỗi kết quả vào một tập tin mới. Hiển thị nội dung của tập tin mới.