ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐH KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP MINH HỌA
HỌC PHẦN NHẬP MÔN LẬP TRÌNH

Mục lục:

Tuần 3. CÁC KHÁI NIỆM CƠ BẢN VỀ KỸ THUẬT LẬP TRÌNH

CÁC BÀI TẬP CƠ BẢN

Bài tập 1:
Viết chương trình in ra các dòng chữ sau đây:

```
1. In C, lowercase letters are significant.
2. main is where program execution begins.
3. Opening and closing braces enclose program statements in a routine.
4. All program statements must be terminated by a semicolon.
```

```
#include <stdio.h>
int main (void)
{
      printf ("\t1. In C, lowercase letters are significant.\n");
      printf ("\t2. main is where program execution begins.\n");
      printf ("\t3. Opening and closing braces enclose program statements in a
routine.\n");
      printf ("\t4. All program statements must be terminated by a
semicolon.\n");
      return 0;
}
```

Chú ý:

1. Ngôn ngữ C phân biệt chữ hoa và chữ thường.
2. Mỗi chương trình luôn có một và chỉ một hàm main. Hàm main sẽ là nơi đầu tiên chương trình thực hiện.
3. Mỗi khi có mở ngoặc thì phải có đóng ngoặc. vd: {…} và (…)
4. Các dòng lệnh phải kết thúc bằng dấu chấm phẩy ';'

Bài tập 2:
Viết chương trình tính ra kết quả của phép trừ 15 cho 87, và xuất kết quả ra màn hình.

```
#include <stdio.h>
int main (void)
{
      int x = 15;
      int y = 87;
      int z = x - y;
      printf ("%d - %d = %d", x, y, z);
      return 0;
}
```

Bài tập 3:
Viết đoạn chương trình sau đây mà không có các ký tự bôi đen, sau đó biên dịch chương trình (F7) và xem thông báo lỗi. **Ghi chú lại các lỗi mà chương trình thông báo**. Sau đó, sửa lại chương trình cho đúng và biên dịch lại.
Ghi chú: thông báo lỗi sẽ hiện ra ở cửa sổ phía dưới của Visual C++. Nhấn **F4** lần lượt nhảy đến các lỗi.

```
#include <stdio.h>
#define TWENTYFIVE      25;
int main ()
{
```

```
        int sum;
        /* COMPUTE RESULT */
        sum = TWENTYFIVE + 37 - 19;
        /* DISPLAY RESULTS */
        printf ("The answer is %i\n", sum);
        return 0;
}
```

Bài tập 4:
Viết đoạn chương trình sau đây và dự đoán kết quả của chương trình. Sau đó biên dịch và chạy chương trình để xem kết quả chính xác. Nếu kết quả khác với mình dự đoán thì phân tích xem tại sao lại như vậy.

```
#include <stdio.h>
int main ()
{
        int answer, result;
        answer = 100;
        result = answer - 10;
        printf ("The result is %i\n", result + 5);
        return 0;
}
```

```
The result is 95
```

Bài tập 5:
Dự đoán kết quả của chương trình sau và giải thích tại sao. Chạy chương trình và so sánh kết quả thật sự với kết quả dự đoán.

```
#include <stdio.h>
#define PRINT(format,x) printf ("x = %"#format"\n", x)
int main (void)
{
        int integer = 5;
        char character = '5';
        PRINT(d, character); PRINT(d, integer);
        PRINT(c, character); PRINT(c, integer=53);
        return 0;
}
```

```
x = 53
x = 5
x = 5
x = 5
```

Bài tập 6:
Dự đoán kết quả của chương trình sau và giải thích tại sao. Chạy chương trình và so sánh kết quả thật sự với kết quả dự đoán.

```
#include <stdio.h>
#define PR(x)      printf("x = %.8g\t", (double)x)
#define PRINT4(x1,x2,x3,x4)    PR(x1); PR(x2); PR(x3); PR(x4)
int main (void)
{
        double d;
        float f;
        long l;
        int i;
```

```
        i = l = f = d = 100/3; PRINT4(i, l, f, d);
        i = l = f = d = 100/3. ; PRINT4(i, l, f, d);
        return 0;
}
```

```
x = 33  x = 33  x = 33  x = 33  x = 33 x = 33  x = 33 x = 33.333332 x = 33.333333
```

Bài tập 7:
Viết chương trình tính giá trị biểu thức sau và giải thích kết quả.
$3x^3 - 5x^2 + 6$
Với $x = 2.55$.

```
#include <stdio.h>
int main (void)
{
        float x = 2.55;
        float y = 3*x*x*x - 5*x*x + 6;
        printf ("%f", y);
        return 0;
}
```

```
23.231623
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Viết chương trình in lên màn hình như sau:
```
************************
*     THAO CHUONG BANG  *
*        NGON NGU C     *
************************
```

2. Viết chương trình nhập vào năm sinh, in ra tuổi.
   Ví dụ nhập 1988 in ra:
```
Ban sinh nam 1988 vay ban 19 tuoi.
```

3. Viết chương trình thực hiện các yêu cầu sau (không dùng hàm chuyển đổi):
   a. Nhập vào một kí tự và in ra mã ASCII tương ứng với kí tự đó.
   b. Nhập vào một số nguyên (1 → 255) và in ra kí tự có mã ASCII tương ứng.

4. Nhập vào bán kính của hình tròn, tính và in ra chu vi, diện tích của hình tròn đó.

5. Viết chương trình nhập vào 2 số nguyên. Xuất ra min, max.
   Ví dụ:
   Nhập vào 5 và 7
   Xuất ra: min =5, max = 7

6. Tìm hiểu ý nghĩa các thông báo lỗi thường gặp.

7. Tìm hiểu bộ thư viện trợ giúp MSDN.

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Nhập vào bán kính đáy R và chiều cao h của hình trụ tròn, tính diện tích đáy, diện tích xung quanh và thể tích của hình trụ tròn theo công thức (các số liệu là số thực, giá trị PI đã được C định nghĩa sẵn bằng hằng số M_PI):
   a. $S_{đáy} = \pi R^2$
   b. $S_{xungquanh} = 2\pi Rh$
   c. $V = S_{đáy} h$

2. Nhập vào số thực x, tính và in ra các giá trị y1, y2, lầy 2 số lẻ:
   a. $y_1 = 4(x^2 + 10x\sqrt{x} + 3x + 1)$
   b. $y_2 = \dfrac{\sin(\pi x^2) + \sqrt{x^2 + 1}}{e^{2x} + \cos\left(\dfrac{\pi}{4}x\right)}$

3. Nhập số tiền nguyên N đồng, đổi ra xem được bao nhiêu tờ 10 đồng, 5 đồng, 2 đồng và 1 đồng.
   Ví dụ:
   N = 543đ = 54 tờ 10đ + 0 tờ 5đ + 1 tờ 2đ + 1 tờ 1đ

4. Nhập vào số nguyên có 3 chữ số, tính tổng 3 chữ số đó.
   Ví dụ:
   Số 543 có tổng 3 chữ số là: 5 + 4 + 3 = 12

5. Viết chương trình nhập giờ, phút, giây và thực hiện kiểm tra tính hợp lệ của dữ liệu nhập vào.

6. Viết chương trình nhập 2 giờ (giờ, phút, giây) và thực hiện tính '+' và '-' của 2 giờ này.

Tuần 4. CÁC CẤU TRÚC LẬP TRÌNH - CẤU TRÚC CHỌN

CÁC BÀI TẬP CƠ BẢN

Bài tập 1

Viết chương trình nhập vào một số x, nếu x = 100 thì xuất ra thông báo "Gia tri cua x la 100", ngược lại, xuất ra thông báo "Gia tri của x khac 100".

```
#include "stdafx.h"
#include <iostream.h>
int main(int argc, char* argv[])
{
	int x;
	cout << "Nhap x = ";
	cin >> x;
	if(x == 100)
		cout << "\nGia tri cua x la 100 ";
	if(x != 100)
		cout << "\nGia tri cua x khac 100 ";
	return 0;
}
```

Bài tập 2

Giải phương trình bậc 1: ax + b = 0

```
#include "stdafx.h"
#include <iostream.h>
int main(int argc, char* argv[])
{
	float x, a, b;
	cout << "Nhap a = ";
	cin >> a;
	cout << "Nhap b = ";
	cin >> b;
	if(a == 0)
	{
		if(b == 0)
			cout << "\nPhuong trinh co vo so nghiem. " << endl;
		else
			cout << "\nPhuong trinh vo nghiem. " << endl;
	}
	else
	{
		cout << "\nPhuong trinh co nghiem duy nhat: x = " << -b/a << endl;
	}
	return 0;
}
```

Bài tập 3

Nhập vào 1 tháng, năm, cho biết tháng đó có bao nhiêu ngày.

```
// Thang co 31 ngay: 1, 3, 5, 7, 8, 10, 12
// Thang co 30 ngay: 4, 6, 9, 11
// Thang 2 co 28 hoac 29 ngay
```

```c
#include <stdio.h>
#include <conio.h>
void main()
{
	//khai bao bien
	int ngay, thang, nam;
	int nhuan;
	//nhap du lieu
	printf("Nhap vao mot thang: ");
	scanf("%d",&thang);
	printf("Nhap vao mot nam: ");
	scanf("%d",&nam);
	//kiem tra nam nhuan
	nhuan = 0;
	if ((nam%400 == 0) || (nam%4 == 0 && nam%100 != 0))
		nhuan = 1;
	ngay = 0;
	switch (thang)
	{
		case 1:
		case 3:
		case 5:
		case 7:
		case 8:
		case 10:
		case 12:
			ngay = 31;
			break;
		case 4:
		case 6:
		case 9:
		case 11:
			ngay = 30;
			break;
		case 2:
			if (nhuan == 1) ngay = 29;
			else ngay = 28;
			break;
	}
	printf("So ngay cua thang %d cua nam %d la: %d",thang, nam, ngay);
	getch();
}
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Giải phương trình bậc 2: $ax^2 + bx + c = 0$

2. Nhập vào 3 số a, b, c. In ra màn hình 3 số này theo thứ tự tăng dần.

3. Nhập vào 4 số a, b, c, d.
   a. In ra số lớn nhất và số nhỏ nhất.
   b. In ra 2 số không phải số lớn nhất và số nhỏ nhất.

4. Nhập vào độ dài 3 cạnh a, b, c của 1 tam giác.
   a. Cho biết 3 cạnh đó có lập thành một tam giác không ?
   b. Nếu có, cho biết loại tam giác này (thường, cân, vuông, đều, vuông cân).

5. Nhập 1 chữ cái, nếu là chữ thường thì đổi thành chữ hoa, ngược lại đổi thành chữ thường.

6. Tính tiền đi taxi từ số km đã được nhập vào, biết:
   - 1 km đầu giá 15000đ
   - Từ km thứ 2 đến km thứ 5 giá 13500đ
   - Từ km thứ 6 trở đi giá 11000đ
   - Nếu đi hơn 120km sẽ được giảm 10% trên tổng số tiền.

7. Xếp loại các học sinh trong lớp. Nhập vào họ tên, điểm toán, lý, hóa của các học sinh. Tính điểm trung bình 3 môn và phân loại như sau:
   - suất xắc: đtb >=9.0
   - giỏi: 9.0 > đtb >= 8.0
   - khá: 8.0 > đtb >= 6.5
   - trung bình: 6.5 > đtb >= 5.0
   - yếu: 5.0 > đtb >= 3.0
   - kém: 3.5 > đtb

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Viết chương trình nhập vào ngày, tháng, năm. Hãy cho biết ngày kế tiếp và ngày trước của ngày đó. (có code tham khảo bên dưới)

```
#include <stdio.h>
#include <conio.h>
void main()
{
	//khai bao bien
	int ngay, thang, nam;
	int ngaytruoc, ngayke;
	int nhuan;
	//nhap du lieu
	printf("Nhap vao mot ngay: ");
	scanf("%d",&ngay);
	printf("Nhap vao mot thang: ");
	scanf("%d",&thang);
	printf("Nhap vao mot nam: ");
	scanf("%d",&nam);
	//kiem tra nam nhuan
	nhuan = 0;
	if ((nam%400 == 0) || (nam%4 == 0 && nam%100 != 0))
		nhuan = 1;
```

```
ngaytruoc = ngay-1;
ngayke = ngay+1;
switch (thang)
{
	case 1:
	case 5:
	case 7:
	case 10:
	case 12:
	case 4:
	case 6:
	case 9:
	case 11:
		if (ngay == 30)
		{
			ngaytruoc = 29;
			ngayke = 1;
		}
		else
		{
			ngaytruoc = ngay-1;
			ngayke = ngay+1;
		}
		break;
	case 2:
		if (nhuan == 1)
		{
			if (ngay == 29)
			{
				ngaytruoc = 28;
				ngayke = 1;
			}
			else if (ngay == 1)
			{
				ngaytruoc = 31;
				ngayke = 2;
			}
		}
		else
			if (ngay == 28)
			{
				ngaytruoc = 27;
				ngayke = 1;
			}
			else if (ngay == 1)
			{
				ngaytruoc = 31;
				ngayke = 2;
			}
		break;
		case 3:
		if (ngay == 31)
		{
			ngaytruoc = 30;
			ngayke = 1;
		}
		else if (ngay == 1)
		{
			if (nhuan == 1)
			{
				ngaytruoc = 29;
```

```
                              ngayke = 2;
                        }
                        else
                        {
                              ngaytruoc = 28;
                              ngayke = 2;
                        }
                  }
                  break;
                  case 8:
                        if (ngay == 31)
                  {
                        ngaytruoc = 30;
                        ngayke = 1;
                  }
                  else if (ngay == 1)
                  {
                        ngaytruoc = 31;
                        ngayke = 2;
                  }
                  break;
      }
      printf("Ngay truoc cua ngay %d cua thang %d cua nam %d la: %d",ngay,
thang, nam, ngaytruoc);
      printf("\n");
      printf("Ngay ke tiep cua ngay %d cua thang %d cua nam %d la: %d",ngay,
thang, nam, ngayke);
      getch();
}
```

Tuần 5. VÒNG LẶP WHILE

CÁC BÀI TẬP CƠ BẢN

Bài 1: Hãy tính tổng s = 1 + 2 + 3 ..... + n

Cách 1:

```
#include <stdio.h>
void main()
{
	int n;
	long s = 0;
	printf("nhap vao n ");
	scanf("%d", &n);
	while (i <= n)
	{
		s += i;
		i++;
	}
	printf("ket qua la: s= %ld", s);
}
```

Bài 2: Tính tổng các số chia hết cho 4 và không chia hết cho 5 nhỏ hơn n

```
#include <stdio.h>
void main()
{
	int n;
	long s = 0;
	printf("nhap vao n ");
	scanf("%d", &n);
	for (int i = 1;i <=n;i++)
		if(!(i%4) && i%5)
			s += i;
	printf("ket qua la: s= %ld", s);
}
```

Bài 3: Tìm số nguyên tố lớn nhất nhỏ hơn n, 0<n<50. (số nguyên tố là số >=2 và chỉ có 2 ước số là 1 và chính nó)

```
#include <stdio.h>
void main()
{
	int k,n;
	do
	{
		printf("nhap so nguyen n: ");
		scanf("%d",&n);
	} while (n<=0 || n>=50);
	k=n-1;
	while (k>1)
	{
		int t=2;
		while (k%t!=0)
			t++;
		if(t==k)
```

```
            {
                  printf("so nguyen to lon nhat nho hon %d la %d\n", n, k);
                  break;
            }
            k--;
      }
      if(k<=1)
            printf("khong co so nguyen to nao nho hon %d",n);
}
```

Bài 4: Nhập vào 1 số nguyên dương, xuất ra số ngược lại. VD: nhập 123, xuất ra 321

```
#include <stdio.h>
void main()
{
      int n;
      do
      {
            printf("nhap so nguyen duong n: ");
            scanf("%d",&n);
      } while (n<=0);
      int don_vi = n%10;
      while (don_vi!=0)
      {
            printf("%5d", don_vi);
            n = n/10;
            don_vi = n%10;
      }
}
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Tính S = $1^3 + 2^3 + 3^3 + \ldots\ldots + N^3$
2. Tính S = $1^2 + 2^2 + 3^2 + \ldots\ldots + N^2$
3. Tính S = 1 + 1/2 + 1/3 + …. + 1/n
4. Tính S = 1/(1x2) + 1/(2x3) + 1/(3x4) + ….. + 1/(n x (n+1))
5. Tính S = 1 + 1.2 + 1.2.3 + …. + 1.2.3….n
6. Tính S = $1 + x + x^2 + \ldots.. + x^n$
7. Tính S = 1! + 2! + 3! +….. + n!
8. Tìm số nguyên dương n nhỏ nhất sao cho 1 + 2 + 3 + …… + n > 1000
9. Tìm và in lên màn hình tất cả các số nguyên trong phạm vi từ 10 đến 99 sao cho tích của 2 chữ số bằng 2 lần tổng của 2 chữ số đó.
10. Tìm các ước số chung nhỏ nhất của 2 số nguyên dương
11. Kiểm tra 1 số có phải là số nguyên tố hay không.
12. In ra tất cả các số nguyên tố nhỏ hơn số n được nhập vào từ bàn phím.

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Kiểm tra xem các chữ số của số nguyên dương n có giảm dần/ tăng dần từ trái sang phải không.
2. Kiểm tra xem 1 số nguyên dương n có phải là số đối xứng/ số toàn số lẻ/ số toàn số chẵn không
3. Tìm chữ số lớn nhất, nhỏ nhất của số nguyên dương n.
4. $S = \sqrt{2*N + \sqrt{2*(N-1) + ... + \sqrt{4+\sqrt{2}}}}$
5. $S = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{n!}$
6. $S = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ...(-1)^n \frac{x^{2n+1}}{(2n+1)!}$ với $-\infty < x < \infty$
7. $S = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + ...(-1)^n \frac{x^{2n}}{(2n)!}$ với $-\infty < x < \infty$
8. $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots = \sum_{n=0}^{\infty} x^n$ với $-1 < x < 1$
9. $\cos(x) = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$ với $-\infty < x < \infty$
10. $\operatorname{arccotan}(x) = \frac{\pi}{2} - x + \frac{1}{3}x^3 - \frac{1}{5}x^5 + \frac{1}{7}x^7 - \frac{1}{9}x^9 + \ldots = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1}$
11. $\arctan(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \ldots = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1} x^{2n-1}$ với $-1 < x < 1$
12. $e^x = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \ldots = \sum_{n=0}^{\infty} \frac{1}{n!} x^n$ với $-\infty < x < \infty$
13. $\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \ldots = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} x^n$ với $-\infty < x < \infty$
14. $\ln\left(\frac{1+x}{1-x}\right) = 2x + \frac{2}{3}x^3 + \frac{2}{5}x^5 + \frac{2}{7}x^7 + \ldots = \sum_{n=1}^{\infty} \frac{2}{2n-1} x^{2n-1}$ với $-\infty < x < \infty$
15. $\sin(x) = x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$ với $-\infty < x < \infty$
16. Nhập số nguyên dương *n*. (i) Kiểm tra có phải là số nguyên tố; (ii) Tìm các thừa số nguyên tố của nó; (iii) In các chữ số từ phải qua trái, (iv) In các chữ số từ trái qua phải (chú ý số 0); (v) In ra tất cả các cặp số nguyên dương *a* và *b* ($a \neq b$) sao cho: $a^2 + b^2 < n$. (vi) Nhập số nguyên *k*. In ra *k* số thập phân đứng sau dấu thập phân (chú ý: 10^k có thể tràn số).

17. Nhập 2 số nguyên dương *a*, *b* khác không. (i) Tìm USCLN(*a*, *b*); (ii) Tìm hai số nguyên *x* và *y* sao cho: USCLN(*a*, *b*) = *a* * *x* + *b* * *y*.
18. Nhập số nguyên dương *n*. Cho biết đó là (i) số đối xứng, (ii) số gần đối xứng, (iii) các chữ số xếp tăng dần hay giảm dần không? (iv) tổng các chữ số cho đến khi nhỏ hơn 10, (v) chữ số lớn và nhỏ nhất.
19. Xác định phần tử thứ *k* của dãy Fibonacci: $f_0 = 0, f_1 = 1, f_k = f_{k-1} + f_{k-2}$, với $k \geq 2$.
20. In ra bình phương của *n* số nguyên dương đầu tiên, nhưng (i) chỉ dùng phép '+' và phép '–'; (ii) Chỉ dùng phép '+'.

Tuần 6. VÒNG LẶP FOR

CÁC BÀI TẬP CƠ BẢN

Bài 1: Viết chương trình tính tổng n số tự nhiên đầu tiên

S = 1 + 2 + 3 + ... + n

```
#include "stdafx.h"
#include "stdio.h"
void main()
{
	int n;
	long S = 0;
	printf(" Nhap gia tri n : ");
	scanf("%d",&n);
	if ( n <= 0)
	{
		printf("Gia tri n khong hop le.\n");
		return;
	}
	for (int i=1;i<=n;i++)
		S = S + i;
	printf("Tong n so tu nhien dau tien la : S = %ld \n",S);
}
```

Bài 2: Viết chương trình cho phép người dùng nhập một số nguyên N > 1. Cho biết N có phải là số nguyên tố hay không ?

Gợi ý:
Để kiểm tra N có phải là số nguyên tố hay không thì ta kiểm tra xem N có ước số trong đoạn [2, n-1] không ? Nếu có ước số trong đoạn đó thì N không phải là số nguyên tố.

```
#include "stdafx.h"
#include "stdio.h"
void main()
{
	int n,i;
	printf("Nhap gia tri N :");
	scanf("%d",&n);
	if (n <= 1)
	{
		printf(" Gia tri N khong hop le.\n");
		return;
	}
	for (i=2; i<=n -1; i++)
	{
		if ( n%i == 0)
		{
			printf("N khong phai la so nguyen to.\n");
			return;
		}
	}
	printf(" N la so nguyen to. \n");
}
```

Bài 3: Viết chương trình vẽ một hình chữ nhật bằng các dấu '*' với chiều dài và chiều rộng do người dùng nhập vào.

```
#include "stdafx.h"
#include "stdio.h"
void main()
{
	int m,n,i,j;
	printf("Nhap chieu dai hinh chu nhat : m = ");
	scanf("%d",&m);
	printf("Nhap chieu rong hinh chu nhat : n = ");
	scanf("%d",&n);
	printf("\n");
	for (i=0; i<n; i++)
	{
		for (j=0; j<m; j++)
			printf("*");
		printf("\n");
	}
}
```

Bài 4: Viết chương trình đảo ngược một dãy số với các cách sử dụng vòng lặp For khác nhau.
Cách 1:

```
#include "stdafx.h"
#include "stdio.h"
int x[] = {1,2,3,4,5};
int n = sizeof(x)/sizeof(int);
void main()
{
	int i,j;
	int c;
	for (i=0, j=n-1; i<j; ++i,--j)
	{
		c= x[i];
		x[i] = x[j];
		x[j] = c;
	}
	printf("\n Day ket qua la :\n");
	for (i=0; i<n; i++)
		printf(" %d", x[i]);
}
```

Cách 2:

```
#include "stdafx.h"
#include "stdio.h"
int x[] = {1,2,3,4,5};
int n = sizeof(x)/sizeof(int);
void main()
{
	int i,j;
	int c;
	for (i=0, j=n-1; i<j; c=x[i], x[i]=x[j], x[j]=c, i++, j--)
	{
		// than FOR rong
	}
	printf("\n Day ket qua la :\n");
	for (i = -1; ++i < n ;) // vang thanh phan thu 3
		printf(" %d", x[i]);
}
```

Cách 3:

```
#include "stdafx.h"
#include "stdio.h"
int x[] = {1,2,3,4,5};
int n = sizeof(x)/sizeof(int);
void main()
{
	int i=0,j=n-1;
	int c;
	for ( ; ; ) // cau lenh FOR vang ca 3 thanh phan
	{
		c= x[i];
		x[i] = x[j];
		x[j] = c;
		if (++i >= --j) break;
	}
	printf("\n Day ket qua la :\n");
	for (i=0; i-n; ) // thay quan he i<n bang bieu thuc i-n
		printf(" %d", x[i++]);
}
```

Cách 4:

```
#include "stdafx.h"
#include "stdio.h"
int x[] = {1,2,3,4,5};
int n = sizeof(x)/sizeof(int);
void main()
{
	int i=0,j=n-1;
	int c;
	for ( ; c=x[i],x[i]=x[j],x[j]=c, ++i<--j; );

	printf("\n Day ket qua la :\n");
	for ( i=0 ; printf(" %d",x[i]), ++i-n ; );
}
```

Bài 5: Viết chương trình tìm Ước số chung lớn nhất (USCLN) của hai số a và b, sử dụng vòng lặp FOR.

```
#include "stdafx.h"
#include "stdio.h"
void main()
{
	int a,b;

	printf("Nhap gia tri a : a =");
	scanf("%d",&a);
	printf("Nhap gia tri b : b =");
	scanf("%d",&b);
	for ( ; a != b ; )
	{
		if (a>b)
			a = a-b;
		else
			b=b-a;
	}
	printf("\n Uoc so chung lon nhat cua hai so la : %d \n",a);
}
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Xuất tất cả các ký tự từ a đến z, A đến Z, 0 đến 9
2. Xuất ra bảng mã ASCII: gồm 2 cột: ký tự và mã ASCII, yêu cầu hiển thị thành từng trang một,
3. Tính tổng các số nguyên tố nhỏ hơn 1000
4. Viết chương trình nhập vào 1 số nguyên , hãy viết cách đọc số nguyên đó
5. Viết chương trình in bảng cửu chương ra màn hình
6. Cần có tổng 20000 từ 3 loại tiền 10000, 20000, 50000. Hãy cho biết tất cả các phương án đó.
7. Các bài toán vẽ hình: tam giác, hình chữ nhật, cây thông,…
8. Liệt kê tất cả các ước số của số nguyên dương n. Cho biết có bao nhiêu ước số và tìm tổng của tất cả các số ước số đó.
9. Tìm BSCNN của 2 số nguyên dương a, b.
10. Kiểm tra 1 số có phải là số nguyên tố không.
11. Tìm chữ số đảo ngược của số nguyên dương n
12. Tìm chữ số lớn nhất/ nhỏ nhất của số nguyên dương n
13. Đếm số lượng chữ số, tính tổng các chữ số của số nguyên dương n
14. Đếm số lượng chữ số lẻ/ chẵn của số nguyên dương n.
15. Tính dãy Fibonacci:

    $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

Làm lại các bài khó của chương trước với vòng lặp for

Tuần 7. CHƯƠNG TRÌNH CON

1. Viết hàm để xác định số nhỏ hơn trong 2 số, sau đó sử dụng hàm này để xác định số nhỏ hơn trong 3 số.
2. Viết hàm tính ước số chung lớn nhất và bội số chung nhỏ nhất của hai số nguyên dương a,b.
3. Viết hàm tính giá trị n! , với n là số nguyên dương và n > 1.

   n ! = 1 x 2 x ... x (n-1) x n
4. Viết hàm tính X^n không dùng đệ quy.
5. Viết chương trình tính hàm tổ hợp $C(n,k) = \frac{n!}{k!(n-k)!}$ trong đó cần cài đặt hàm tính n!.
6. Viết hàm tính chu vi và diện tích hình chữ nhật khi biết độ dài 2 cạnh. Sau đó vẽ hình chữ nhật ra màn hình bằng các dấu *. Hàm tính chu vi, diện tích và hàm vẽ hình chữ nhật phải độc lập nhau.

   * * * * * * * * * * * * * * * *

   * *

   * *

   * *

   * *

   * * * * * * * * * * * * * * * *
7. Viết chương trình con xuất ra tam giác Pascal như sau :

   1

   1 1

   1 2 1

   1 3 3 1

   1 4 6 4 1
8. Viết hàm nhập vào tháng bằng số rồi in ra tên tháng bằng chữ ra màn hình.
9. Viết hàm để kiểm tra một ngày nào đó có hợp lệ hay không, kiểm tra năm nhuần.
10. Viết hàm đổi ngày tháng năm thành thứ trong tuần.
11. Viết hàm để nhận biết một số nguyên dương có phải là số nguyên tố hay không.
12. Viết chương trình in ra tất cả các số nguyên tố nhỏ hơn số nguyên dương M cho trước (sử dụng hàm kiểm tra số nguyên tố đã cài đặt ở trên).
13. Viết hàm kiểm tra một số nguyên dương có phải là số chính phương hay không. Xuất tất cả các số chính phương trong khoảng A,B.

14. Một số tự nhiên được gọi là số hoàn thiện nếu nó bằng tổng tất cả các ước số của nó, kể cả 1. Hãy viết hàm kiểm tra một số có phải là số hoàn thiện hay không, và in ra tất cả các số hoàn thiện nhỏ hơn số N cho trước.
15. Viết hàm tính tổng nghịch đảo của n số nguyên.
16. Viết hàm đếm số các số chẵn trong khoảng từ M đến N, tính tổng các số đó.
17. Tính Sin của giá trị x bất kì theo công thức :

    $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$ So sánh kết quả với hàm sin(double) đã có.
18. Viết chương trình con xuất ra màn hình dãy số Fibonanci cấp n, xác định theo công thức :

    Fib(1) = 1

    Fib(2) = 1

    Fib(n) = Fib(n-1) + Fib(n-2) với n> 2.
19. Ta có các loại tiền 50.000, 20.000, 10.000, 5.000, 2.000, 1.000. Viết hàm cho biết số tờ của từng loại tiền để tổng của chúng bằng một số tiền nào đó mà người dùng nhập vào. Cho biết tất cả các phương án có thể có, sau đó thông biết phương án nào cho kết quả có số tờ ít nhất.
20. Cho trước mảng số nguyên n phần tử và số M. Tìm tập hợp các phần tử trong A sao cho tổng của chúng bằng M.

Tuần 8. CHƯƠNG TRÌNH CON (tt)

CÁC BÀI TẬP THÊM VỀ CHƯƠNG TRÌNH CON

1. Viết hàm đổi một số hệ 10 sang hệ 16 và ngược lại.
2. Viết hàm làm tròn một số thực với 2 tham số đầu vào : số cần phải làm tròn và số chữ số phần thập phân có nghĩa sau khi làm tròn.
3. Viết chương trình đảo vị trí các kí số trong một số. Dữ liệu input là một số nguyên dương n, giá trị của n sẽ thay đổi sau khi gọi thực hiện chương trình con đảo kí số.

   Ví dụ :

```
void main()
{
	int n = 12345;
	DaoKiSo(n);
	// n == 54321
}
```

4. Viết chương trình con rút gọn một phân số.
5. Viết hàm tính khoảng cách giữa 2 điểm trong hệ tọa độ vuông góc khi biết tọa độ của chúng.
6. Viết hàm tính chu vi diện tích của một hình chữ nhật, hình tam giác trong hệ trục tọa độ vuông góc khi biết tọa độ các đỉnh.
7. Trong hệ tọa độ Đề-các vuông góc, cho hai điểm A, B có tọa độ lần lược là (X1, Y1) và (X2, Y2) . Viết chương trình xác định hai hệ số a,b trong phương trình đường thẳng y = ax + b đi qua 2 điểm A, B đó.
8. Cho 3 điểm A, B, C với các tọa độ tương ứng (X1, Y1) , (X2, Y2) và (X3, Y3). Viết chương trình xác định trọng tâm của tam giác đó.
9. Cho trước trong hệ tọa độ cuông góc các điểm A, B, C và một điểm X có tọa độ bất kì. Hãy xác định xem X có nằm trong tam giác hay không.
10. Viết chương trình in theo trật tự tăng dần tất cả các phân số tối giản trong khoảng (0,1) có mẫu số không vượt quá 7.
11. Viết chương trình con đổi chữ thường thành chữ hoa.

Tuần 9. KIỂU MẢNG MỘT CHIỀU VÀ MỘT SỐ KỸ THUẬT CƠ BẢN

CÁC BÀI TẬP CƠ BẢN

1. Tính tổng tất cả các phần tử trên mảng.

```
#include "stdafx.h"
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <stdlib.h>
#include <time.h>

void NhapMang(int[], int);
void XuatMang(int[], int);

int TinhTongCacPhanTu(int[], int);

int main(int argc, char* argv[])
{
	int n = 10;
	int a[10];

	NhapMang(a, n);
	XuatMang(a, n);

	int s = TinhTongCacPhanTu(a, n);
	cout << "\nTong cac phan tu trong mang = " << s << "\n";

	return 0;
}

/************************************************************/
int TinhTongCacPhanTu(int a[], int n)
{
	int s = 0;
	for(int i=0; i<n; i++)
		s = s + a[i];

	return s;
}

/********************* Cac ham nhap xuat *********************/
void NhapMang(int a[], int n)
{
	srand((unsigned int)time(NULL));

	cout << "\n... Phat sinh tu dong cac phan tu trong mang...\n";
	for(int i=0; i<n; i++)
	{
		a[i] = rand()%90 + 10;
	}
}

void XuatMang(int a[], int n)
{
	cout << "\nCac phan tu hien co trong mang: ";

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

```
	{
		cout << a[i] << "  ";
	}

	cout << "\n";
}
/*************************************************************/
```

2. Đếm số lần xuất hiện một phần tử x bất kỳ.

```
#include "stdafx.h"
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <stdlib.h>
#include <time.h>

void NhapMang(int[], int);
void XuatMang(int[], int);

int DemSoLanXuatHienMotPhanTu(int[], int, int);

int main(int argc, char* argv[])
{
	int n = 10;
	int a[10];

	NhapMang(a, n);
	XuatMang(a, n);

	int x = 0;
	cout << "\nNhap phan tu x muon tim: ";
	cin >> x;
	int so_lan_xuat_hien = DemSoLanXuatHienMotPhanTu(a, n, x);
	cout << "\nSo lan xuat hien phan tu " << x << " la "
		<< so_lan_xuat_hien << " lan\n";

	return 0;
}

/*************************************************************/
int DemSoLanXuatHienMotPhanTu(int a[], int n, int x)
{
	int so_lan_xuat_hien = 0;
	for(int i=0; i<n; i++)
	{
		if(a[i] == x)
			so_lan_xuat_hien++;
	}

	return so_lan_xuat_hien;
}
/********************** Cac ham nhap xuat **********************/
/* ... */
```

3. Trộn 2 mảng một chiều a, b các phần tử xen kẽ nhau thành một mảng một chiều (a, b có thể có số phần tử khác nhau).

```
/*************************************************************/
/* ... */
```

```
void Tron2Mang(int[], int, int[], int, int[]);

int main(int argc, char* argv[])
{
	int n = 5, m = 7;
	int a[5];
	int b[7];
	int c[100];

	srand((unsigned int)time(NULL));

	cout << "\nMang a :";
	NhapMang(a, n);
	XuatMang(a, n);

	cout << "\nMang b :";
	NhapMang(b, m);
	XuatMang(b, m);

	Tron2Mang(a, n, b, m, c);
	cout << "\nMang c la ket qua tron 2 mang a, b :";
	XuatMang(c, n + m);

	return 0;
}

/*************************************************************/
void Tron2Mang(int a[], int n, int b[], int m, int c[])
{
	int min = (n>m ? m:n);

	int i = 0, j = 0;
	for(i=0; i<min; i++, j+=2)
	{
		c[j] = a[i];
		c[j+1] = b[i];
	}

	while(i<n)
	{
		c[j++] = a[i++];
	}

	while(i<m)
	{
		c[j++] = b[i++];
	}
/* ... */
/*************************************************************/
```

4. Xóa một phần tử bất kỳ trên mảng.

```
/*************************************************************/
/* ... */

void Xoa1PhanTu(int[], int&, int);

int main(int argc, char* argv[])
{
	int n = 10;
	int a[10];
```

```
	srand((unsigned int)time(NULL));

	cout << "\nMang a ban dau :";
	NhapMang(a, n);
	XuatMang(a, n);

	int x;
	cout << "\nNhap phan tu x muon xoa: ";
	cin >> x;

	cout << "\nMang a sau khi xoa phan tu " << x << " : ";
	Xoa1PhanTu(a, n, x);
	XuatMang(a, n);

	return 0;
}

/*************************************************************/
void Xoa1PhanTu(int a[], int &n, int x)
{
	int b[100];
	for(int i=0; i<n; i++)
		b[i] = a[i];

	int m = 0;
	for(i=0; i<n; i++)
	{
		if(b[i] != x)
			a[m++] = b[i];
	}

	n = m;
}

/* ... */
/*************************************************************/
```

5. Tạo ra một mảng gồm n phần tử là các số liên tiếp trong dãy Fibonaci.

```
/*************************************************************/
/* ... */

void TaoMangFibonaci(int[], int);

int main(int argc, char* argv[])
{
	int n = 10;
	int a[100];

	cout << "\Nhap so phan tu cua mang (n): ";
	cin >> n;

	cout << "\nMang a Fibonaci: ";
	TaoMangFibonaci(a, n);
	XuatMang(a, n);

	return 0;
}

/*************************************************************/
void TaoMangFibonaci(int a[], int n)
```

```
{
	int f1 = 0;
	int f2 = 1;
	int f;

	a[0] = f1;
	a[1] = f2;
	for(int i=2; i<n; i++)
	{
		f = f1 + f2;
		a[i] = f;

		f1 = f2;
		f2 = f;
	}
}

/* ... */
/*************************************************************************/
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Đếm số lần xuất hiện của các số nguyên dương.
2. Tính tổng tất cả các phần tử không âm.
3. Nối 2 mảng một chiều thành một.
4. Đếm số phần tử là số nguyên tố và tính tổng các phần tử này.
5. Đếm số phần tử là số chính phương và tính tổng các phần tử này.

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Trộn 2 mảng một chiều có cùng độ dài thành một mảng một chiều với mỗi phần tử của mảng mới là tổng của 2 phần tử tương ứng từ 2 mảng cho trước.
2. Xóa n phần tử liên tục trên mảng bắt đầu từ một vị trí x cho trước.
3. Nhập vào 2 mảng có cùng kích thước, tạo mảng mới gồm các phần tử là UCLN của 2 phần tử tương ứng.
4. Tính tổng giai thừa của các phần tử trong mảng cho trước.
5. Nhập vào 2 mảng một chiều, xóa trên 2 mảng này tất cả các phần tử trùng nhau của 2 mảng.

Tuần 10. TÌM KIẾM VÀ SẮP XẾP TRÊN MẢNG MỘT CHIỀU

CÁC BÀI TẬP CƠ BẢN

1. Tìm một phần tử x bất kỳ trên mảng theo kiểu tuần tự.

```
/*********************************************************************/
/* ... */

int TimKiem(int[], int, int);

int main(int argc, char* argv[])
{
	int n = 10;
	int a[10];

	srand((unsigned int)time(NULL));

	NhapMang(a, n);
	XuatMang(a, n);

	int x;
	cout << "\nNhap phan tu x tim: ";
	cin >> x;

	int kq = TimKiem(a, n, x);
	if(kq == -1)
		cout << "\nKhong tim thay x";
	else
		cout << "\nTim thay x tai vi tri so " << kq
		     << " (chi so duoc tinh bat dau tu 0)";

	cout << "\n\n";

	return 0;
}

/*************************************************************/

/* Neu tim thay x thi tra ve vi tri xuat hien lan dau tien
   cua x trong mang. Neu khong tim thay x thi tra ve gia tri -1*/
int TimKiem(int a[], int n, int x)
{
	int vitri = -1;
	for(int i=0; (i < n) && (vitri == -1); i++)
	{
		if(x == a[i])
		{
			vitri = i;
		}
	}

	return vitri;
}

/* ... */
/*********************************************************************/
```

2. Sắp xếp các phần tử trên mảng tăng dần hoặc giảm dần theo yêu cầu.

```
/* ..................................................................... */
void SapXep(int[], int, bool);
void HoanVi(int&, int&);

int main(int argc, char* argv[])
{
	int n = 10;
	int a[100];

	srand((unsigned int)time(NULL));

	cout << "\nMang a ban dau :";
	NhapMang(a, n);
	XuatMang(a, n);

	cout << "\nMang a sau khi sap xep tang dan:";
	SapXep(a, n, true);
	XuatMang(a, n);

	cout << "\nMang a sau khi sap xep giam dan:";
	SapXep(a, n, false);
	XuatMang(a, n);

	return 0;
}

void SapXep(int a[], int n, bool bSapTang)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=i+1; j<n; j++)
		{
			// bSapTang == true -> sap tang
			// bSapTang == false -> sap giam

			if(bSapTang == true)
			{
				if(a[j] < a[i])
					HoanVi(a[i], a[j]);
			}
			else
			{
				if(a[j] > a[i])
					HoanVi(a[i], a[j]);
			}
		}
	}
}

void HoanVi(int &a, int &b)
{
	int temp = a;
	a = b;
	b = temp;
}

/* ..................................................................... */
```

3. Sắp xếp các phần tử trên mảng sao cho các số dương tăng dần và các số âm giảm dần.

```
/*****************************************************************/
/* ... */

void SapXepDuongTangAmGiam(int a[], int n)
{
	for(int i=0; i<n-1; i++)
	{
		for(int j=i+1; j<n; j++)
		{
			if(a[j]>0 && a[i]>0)
			{
				if(a[j] < a[i])
					HoanVi(a[i], a[j]);
			}

			if(a[j]<0 && a[i]<0)
			{
				if(a[j] > a[i])
					HoanVi(a[i], a[j]);
			}
		}
	}
}

/* ... */

void NhapMang(int a[], int n)
{
	cout << "\n... Phat sinh tu dong cac phan tu trong mang...\n";
	for(int i=0; i<n; i++)
	{
		a[i] = rand()%90 - 30;
	}
}

/* ... */
/*****************************************************************/
```

4. Đảo ngược mảng (phần tử đầu tiên sẽ về cuối mảng, phần tử cuối mảng đưa lên đầu…).

```
/*****************************************************************/
/* ... */

void DaoNguocMang(int[], int);

int main(int argc, char* argv[])
{
	int n = 11;
	int a[11];

	srand((unsigned int)time(NULL));

	NhapMang(a, n);
	XuatMang(a, n);

	cout << "\nMang a sau dao nguoc: ";
	DaoNguocMang(a, n);
	XuatMang(a, n);
```

```
	return 0;
}

/*************************************************************/
void DaoNguocMang(int a[], int n)
{
	for(int i=0; i<n/2; i++)
	{
		HoanVi(a[i], a[n-i-1]);
	}
}

/* ... */
/*********************************************************************/
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Tìm một phần tử x bất kỳ trên mảng theo kiểu nhị phân.
2. Kiểm tra xem mảng có tăng dần hay giảm dần không.
3. Đếm số mảng con tăng dần hoặc giảm dần trong mảng.
4. Cho mảng n phần tử và k < n. In ra tổng lớn nhất của k phần tử liên tiếp xuất hiện trên mảng.
5. Đếm số lượng các phần tử khác nhau xuất hiện trong mảng.
6. Cũng với yêu cầu cho biết số lượng phần tử khác nhau, nhưng biết rằng, các giá trị xuất hiện nằm trong khoảng từ 1 $\rightarrow$ k. (tạo mảng từ 1 $\rightarrow$ k, ban đầu bằng 0).
7. Mảng x và y chứa hoành độ và tung độ của các điểm trên mặt phẳng hai chiều. In ra khoảng cách xa nhất giữa 2 điểm.
8. Mảng a chứa hệ số của đa thức $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$. Nhập x. Tính giá trị đa thức.
9. Cho 2 mảng a và b có m và n phần tử. Nhập số q (nguyên dương). Tìm tổng a[i] + b[j] nhỏ nhất nhưng lớn hơn q.

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Sắp xếp các phần tử trên mảng sao cho các số dương tăng dần và ở đầu mảng, các số âm giảm dần và ở cuối mảng, các số 0 ở giữa.
2. Sắp xếp các phần tử trên mảng sao cho các số chẵn tăng dần, các số lẻ giảm dần.
3. Sắp xếp các phần tử trên mảng sao cho các số chẵn tăng dần và ở đầu mảng, các số lẻ giảm dần và ở cuối mảng.
4. Kiểm tra xem có tồn tại mảng con tăng dần hay giảm dần không. Nếu có, in mảng con tăng dần dài nhất xuất hiện trong mảng. Nếu có nhiều mảng cùng dài nhất thì chỉ cần in ra một.

5. Cho mảng có n phần tử. Nhập m là số nguyên dương nhỏ hơn n. Chia mảng làm 2 đoạn a[0] $\rightarrow$ a[m – 1] và a[m] $\rightarrow$ a[n – 1]. Không dùng thêm mảng phụ. Chuyển chỗ các phần tử để thành a[m] $\rightarrow$ a[n – 1] $\rightarrow$ a[0] $\rightarrow$ a[m – 1].
6. Mảng a (k phần tử) và b (l phần tử) chứa hệ số của 2 đa thức. Tính tích của 2 đa thức trên.
7. Cho 2 mảng a và b có m và n phần tử. Các phần tử trong mỗi mảng là khác nhau. Tìm số lượng phần tử chung. Mở rộng: giả sử có phần tử trùng. (while)

Tuần 11. MẢNG 2 CHIỀU

CÁC BÀI TẬP CƠ BẢN

Bài 1: Viết chương trình nhập, xuất một mảng số nguyên hai chiều có m dòng và n cột. Xác định phần tử lớn nhất và nhỏ nhất trong mảng.

```
#include "stdafx.h"
#include "stdio.h"
#define max_dong 100
#define max_cot 100
void main()
{
	int m,n;
	int a[max_dong][max_cot];
	int i,j;

	// nhap cac phan tu cho mang 2 chieu co m dong va n cot
	printf(" Nhap so dong : m = ");
	scanf("%d",&m);
	printf("Nhap so cot : n = ");
	scanf("%d",&n);
	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			printf("\n A[%d,%d] = ",i+1,j+1);
			scanf("%d",&a[i][j]);
		}

	int max = a[0][0], min=a[0][0], x_max=0, y_max=0, x_min=0, y_min=0;

	// tim phan tu lon nhat va nho nhat trong mang
	for (i=0;i<m;i++)
		for (j=0;j<n;j++)
		{
			if (max <= a[i][j])
			{
				max = a[i][j];
				x_max = i;
				y_max = j;
			}
			if (min >= a[i][j])
			{
				min = a[i][j];
				x_min = i;
				y_min = j;
			}
		}
	// xuat cac phan tu cua mang 2 chieu
	printf("\n Cac phan tu cua mang A:\n");
	for (i=0;i<m;i++)
	{
		for (j=0; j<n; j++)
			printf(" %d ",a[i][j]);
		printf("\n");
	}
	// xuat vi tri va gia tri cac phan tu lon nhat, nho nhat
```

```
	printf("\n Phan tu lon nhat cua mang la %d tai vi tri (%d,%d)",
max,x_max+1,y_max+1);
	printf("\n Phan tu nho nhat cua mang la %d tai vi tri (%d,%d)",
min,x_min+1,y_min+1);
}
```

Bài 2: Viết chương trình sắp xếp ma trận các số thực tăng dần từ trên xuống dưới và từ trái sang phải bằng hai phương pháp dùng và không dùng mảng phụ.

Cách 1 : Không sử dụng mảng phụ.

```
#include "stdafx.h"
#include "stdio.h"
#define max_dong 100
#define max_cot 100
void main()
{
	int m,n;
	int a[max_dong][max_cot];
	int i,j;

	// nhap cac phan tu cho mang 2 chieu co m dong va n cot
	printf(" Nhap so dong : m = ");
	scanf("%d",&m);
	printf("Nhap so cot : n = ");
	scanf("%d",&n);
	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			printf("\n A[%d,%d] = ",i+1,j+1);
			scanf("%d",&a[i][j]);
		}
	for (int k=0; k<=m*n-2; k++)
		for (int l=k+1; l<=m*n-1; l++)
		if (a[k/n][a%n] > a[l/n][l%n])
		{
			int temp = a[k/n][k%n];
			a[k/n][k%n] = a[l/n][l%n];
			a[l/n][l%n] = temp;
		}
	// xuat cac phan tu cua mang 2 chieu
	printf("\n Cac phan tu cua mang A:\n");
	for (i=0;i<m;i++)
	{
		for (j=0; j<n; j++)
			printf(" %d ",a[i][j]);
		printf("\n");
	}

}
```

Cách 2 : Sử dụng mảng phụ.

```
#include "stdafx.h"
#include "stdio.h"
#define max_dong 50
#define max_cot 50
void main()
{
	int m,n;
	int a[max_dong][max_cot];
```

```
	int i,j;

	// nhap cac phan tu cho mang 2 chieu co m dong va n cot
	printf(" Nhap so dong : m = ");
	scanf("%d",&m);
	printf("Nhap so cot : n = ");
	scanf("%d",&n);
	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			printf("\n A[%d,%d] = ",i+1,j+1);
			scanf("%d",&a[i][j]);
		}

	//Do ma tran ra mang mot chieu b
	int b[max_dong*max_cot];
	int k = 0;
	for (i=0;i<m;i++)
		for (j=0;j<n;j++)
		{
			b[k] = a[i][j];
			k = k+1;
		}

	// Sap xep mang mot chieu b
	for (i=0; i<k-1; i++)
		for (j=i+1; j<k; j++)
			if (b[i] > b[j])
			{
				int tmp = b[i];
				b[i] = b[j];
				b[j] = tmp;
			}

	// Do mang mot chieu b tro lai mang hai chieu a
	k = 0;
	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			a[i][j] = b[k];
			k = k+1;
		}

	// xuat cac phan tu cua mang 2 chieu
	printf("\n Cac phan tu cua mang A:\n");
	for (i=0;i<m;i++)
	{
		for (j=0; j<n; j++)
			printf(" %d ",a[i][j]);
		printf("\n");
	}

}
```

Bài 3: Cho một mảng số nguyên A có m dòng và n cột. Một phần tử được gọi là điểm yên ngựa nếu phần tử đó là phần tử nhỏ nhất trong dòng và lớn nhất trong cột. Viết chương trình xác định tất cả các đểm yên ngựa có thể có.

```
#include "stdafx.h"
#include "stdio.h"
```

```
#define max_dong 50
#define max_cot 50
void main()
{
	int m,n;
	int a[max_dong][max_cot];
	int i,j;

	// nhap cac phan tu cho mang 2 chieu co m dong va n cot
	printf(" Nhap so dong : m = ");
	scanf("%d",&m);
	printf("Nhap so cot : n = ");
	scanf("%d",&n);
	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			printf("\n A[%d,%d] = ",i+1,j+1);
			scanf("%d",&a[i][j]);
		}

	// tim kiem cac phan tu yen ngua
	for (i= 0; i<m; i++)
		for (j=0; j<n; j++)
		{
			bool IsMinRow = true, IsMaxCol = true;

			// kiem tra a[i][j] co phai la phan tu trong dong hay khong ?
			for (int k = 0; k<n; k++)
				if (a[i][k] < a[i][j])
				{
					IsMinRow = false;
					break;
				}
			// kiem tra a[i][j] co phai la phan tu lon nhat trong cot khong ?
			for (int l=0; i<m; l++)
				if (a[l][j] > a[i][j])
				{
					IsMaxCol = false;
					break;
				}
			// neu a[i][j] thao dieu kien --> a la phan tu yen ngua
			if (IsMaxCol && IsMinRow)
				printf("\n A[%d,%d] = %d la phan tu yen
ngua.",i+1,j+1,a[i][j]);
		}
}
```

Bài 4: Cho ma trận các số thực A(m x n). Hãy xây dựng ma trận B(m x n) từ ma trận A sao cho B[i][j] = số lượng phần tử dương xung quanh A[i][j] trong ma trận A (B[i][j] tối đa là 8 và nhỏ nhất là 0).

```
#include "stdafx.h"
#include "stdio.h"
#define max_dong 50
#define max_cot 50
void main()
{
	int m,n;
	float a[max_dong][max_cot];
	int i,j;

	// nhap cac phan tu cho mang 2 chieu co m dong va n cot
```

```
	printf(" Nhap so dong : m = ");
	scanf("%d",&m);
	printf("Nhap so cot : n = ");
	scanf("%d",&n);
	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			printf("\n A[%d,%d] = ",i+1,j+1);
			scanf("%f",&a[i][j]);
		}

	// phat sinh mang b
	float b[max_dong][max_cot];

	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			int count = 0;
			for (int k=-1; k<=1; k++)
				for (int l=-1; l<=1; l++)
					if ((i+k>=0)&&(i+k<m)&&(j+l>=0)
&&(j+l<n)&&(a[i+k][j+l] > 0))
						count = count + 1;
			b[i][j] = count;
		}

	// xuat cac phan tu cua mang 2 chieu
	printf("\n Cac phan tu cua mang B:\n");
	for (i=0;i<m;i++)
	{
		for (j=0; j<n; j++)
			printf(" %f ",b[i][j]);
		printf("\n");
	}
}
```

Bài 5: Hãy sắp xếp ma trận sao cho dòng có tổng nhỏ hơn nằm ở trên và dòng có tổng dòng lớn hơn nằm ở dưới.

```
#include "stdafx.h"
#include "stdio.h"
#define max_dong 50
#define max_cot 50
void main()
{
	int m,n;
	int a[max_dong][max_cot];
	int i,j;

	// nhap cac phan tu cho mang 2 chieu co m dong va n cot
	printf(" Nhap so dong : m = ");
	scanf("%d",&m);
	printf("Nhap so cot : n = ");
	scanf("%d",&n);
	for (i=0; i<m; i++)
		for (j=0; j<n; j++)
		{
			printf("\n A[%d,%d] = ",i+1,j+1);
			scanf("%d",&a[i][j]);
		}
```

```
	// su dung mang B luu gia tri tong cac dong cua matrix A
	int b[max_dong];
	int k = 0;

	// tinh tong cac dong va luu vao B
	for (i=0; i<m; i++)
	{
		int tong = 0;
		for (j=0; j<n ;j++)
			tong = tong + a[i][j];
		b[k++] = tong;
	}

	// sap xep lai mang A theo thong tin tong dong trong B
	for (i=0; i<k-1; i++)
		for (j=i+1; j<k; j++)
		{
			if (b[i] > b[j])
			{
				// hoan vi trong B
				int tmp;
				tmp = b[i];
				b[i] = b[j];
				b[j] = tmp;

				// hoan vi dong trong A
				for (int k = 0; k<n; k++)
				{
					tmp = a[i][k];
					a[i][k] = a[j][k];
					a[j][k] = tmp;
				}
			}
		}
	// xuat cac phan tu cua mang 2 chieu
	printf("\n Cac phan tu cua mang A sau khi thay doi :\n");
	for (i=0;i<m;i++)
	{
		for (j=0; j<n; j++)
			printf(" %d ",a[i][j]);
		printf("\n");
	}
}
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Tính tổng tất cả các phần tử trên mảng.
2. Đếm số lần xuất hiện một phần tử x bất kỳ.
3. Đếm số lần xuất hiện của các số nguyên dương.
4. Tính tổng tất cả các phần tử không âm.
5. Tính tổng các phần tử trên đường chéo chính.
6. Tính tổng các phần tử trên đường chéo phụ.
7. Sắp xếp các phần tử trên mảng tăng dần trên trên từng dòng.
8. Tính tổng và tích 2 ma trận.

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Sắp xếp các phần tử trên mảng tăng dần trên từng cột và giảm dần trên từng dòng.
2. Sắp xếp các phần tử trên mảng tăng dần lần lượt trên dòng và trên cột.
3. Tính tổng và tích của n ma trận.
4. Xoay 1 ma trận theo chiều bất kỳ.
5. Xoay 1 ma trận theo chiều bất kỳ n bước.
6. Xóa một hàng hoặc một cột bất kỳ trên mảng 2 chiều.
7. Nhập vào 2 mảng 2 chiều, tìm tất cả các phần tử trùng nhau của 2 mảng và thay vào đó là số 0.
8. Nhập vào 2 ma trận cùng kích thước n x m, in ra ma trận tổng.
9. Nhập vào 2 ma trận cùng kích thước n x m, tính ma trận tích.
10. Nhập vào 1 ma trận, xuất ra ma trận nghịch đảo.
11. Tìm giá phần tử lớn nhất và nhỏ nhất trên từng dòng, từng cột, và trên toàn ma trận.

Tuần 12. 13. KIỂU KÝ TỰ VÀ KIỂU CHUỖI

CÁC BÀI TẬP CƠ BẢN

1. Nhập một chuỗi S từ bàn phím. Kiểm tra xem chuỗi có phải là chuỗi đối xứng.

```
Ví dụ:
Nhập: S = "aBCdCBa"
Xuất: Đối xứng
Nhập: S = "aBCdBCa"
Xuất: Không dối xứng
```

Ý tưởng:
Giả sử chuỗi là chuỗi đối xứng. Nếu phát hiện 1 vị trí mà đối xứng với nó (cùng vị trí nhưng tính từ cuối chuỗi) thì chuỗi không còn đối xứng nữa.

Chương trình:

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

// Nên định nghĩa MAX để có thể thay đổi kích thước mảng nhanh chóng
#define MAX 100

// Nên đặt prototype ở đây để biết hàm nhận vào cái gì và trả về cái gì
// Đầu vào: Chuỗi cần kiểm tra
// Đầu ra:  1 nếu là chuỗi đối xứng, 0 nếu là chuỗi không đối xứng
int LaChuoiDoiXung(char []);

void main()
{
	char str[MAX];

	printf("Nhap 1 chuoi: ");
	// Nên sử dụng gets thay vì scanf để có thể nhập chuỗi có khoảng trắng
	gets(str);

	int kq = LaChuoiDoiXung(str);
	if (kq==1)
		printf("Chuoi \"%s\" la chuoi doi xung.", str); // Sử dụng \" để xuất "
	else
		printf("Chuoi \"%s\" khong phai la chuoi doi xung.", str);
	getch();
}

int LaChuoiDoiXung(char str[MAX])
{
	int Flag = 1;			// cứ giả sử đây là chuỗi đối xứng
	int l = strlen(str);	// strlen để lấy độ dài chuỗi. Nên tính 1 lần để sử
dụng lại
	for (int i=0; i<l/2; i++)
		if (str[i]!=str[l-i-1])
			Flag = 0;	// phát hiện không đối xứng thì cập nhật cờ

	return Flag;
}
```

Ghi chú:

Có thể không cần sử dụng biến Flag để kiểm tra mà có thể **return 0**; ngay khi kiểm tra thấy có vị trí không còn đối xứng. Cụ thể như sau:

```
int LaChuoiDoiXung(char str[MAX])
{
	int l = strlen(str);	// strlen để lấy độ dài chuỗi. Nên tính 1 lần để sử dụng lại
	for (int i=0; i<l/2; i++)
		if (str[i]!=str[l-i-1])
			return 0;	// phát hiện không đối xứng thì trả về 0 ngay!

	return 1;	// chưa lần nào phát hiện vị trí không đối xứng ➔ chuỗi đối xứng
}
```

2. Nhập một chuỗi S từ bàn phím. Tìm ký tự xuất hiện nhiều nhất trong chuỗi đó và số lần xuất hiện.

```
Ví dụ:
Nhập: S = "Nguyen Thi B"
Xuất: n 2 lần
```

Ý tưởng:

- Dùng 2 biến lưu ký tự xuất hiện nhiều nhất và số lần xuất hiện nhiều nhất đó.
- Ban đâu ký tự xuất hiện nhiều nhất chưa có nên số lần xuất hiện nhiều nhất là 0.
- Xét 1 ký tự tại vị trí i. Đếm xem từ vị trí i về sau ký tự này xuất hiện bao nhiêu lần. Nếu số lần xuất hiện này nhiều hơn số lần hiện tại thì ta cập nhật ký tự xuất hiện nhiều nhất và số lần xuất hiện nhiều nhất trùng với ký tự vừa xét.

Chương trình:

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

// Nên định nghĩa MAX để có thể thay đổi kích thước mảng nhanh chóng
#define MAX 100

// Nên đặt prototype ở đây để biết hàm nhận vào cái gì và trả về cái gì
// Đầu vào: Chuỗi cần kiểm tra và biến chứa ký tự xuất hiện nhiều nhất tìm được
// Đầu ra:  Số lần xuất hiện nhiều nhất của ký tu. Bằng 0 nếu không có ký tự nào
int KyTuXuatHienNhieuNhat(char [], char &);

void main()
{
	char str[MAX];

	printf("Nhap 1 chuoi: ");
	// Nên sử dụng gets thay vì scanf để có thể nhập chuỗi có khoảng trắng
	gets(str);

	char chr;
	int max = KyTuXuatHienNhieuNhat(str, chr);
	if (max!=0) // Trường hợp max  khác 0 ⇔ Chuỗi không rỗng
		printf("Ky tu %c xuat hien nhieu nhat la %d lan.", chr, max);
	else
		printf("Chuoi rong! Khong co ky tu xuat hien nhieu nhat.");
	getch();
}
```

```
int KyTuXuatHienNhieuNhat(char str[], char &chr)
{
	int i, j, length = strlen(str);
	char curchr;		// Ký tự đang xét
int curcount;		// Số lần xuất hiện của ký tự đang xét
int max = 0;		// Số lần xuất hiện nhiều nhất ban đầu là 0 (chưa có)


	for (i=0; i<length; i++)
	{
		curchr = str[i];
		curcount = 1;

		// Lấy ký tự thứ i ra kiểm tra với các ký tự sau i
		for (j=i+1;j<length; j++)
			if (str[j] == str[i])
				curcount++;

		// Tìm được số lần xuất hiện nhiều hơn max thì cập nhật lại số lần và ký
tự
		if (max < curcount)
		{
			max = curcount;
			chr = curchr;
		}
	}

	return max;
}
```

3. Nhập họ tên của một người từ bàn phím. Hãy chuẩn hóa chuỗi họ tên này. (Xóa các khoảng trắng thừa và ký tự đầu tiên của họ, chữ lót và tên phải viết hoa, các ký tự còn lại viết thường).

```
Ví du:
Nhập: "     NgUyen    VaN  A     "
Xuất: "Nguyen Van A"
```

Ý tưởng:

- Xóa các khoảng trắng dư ở bên trái chuỗi.
- Xóa các khoảng trắng dư ở bên phải chuỗi.
- Nếu tìm được 2 ký tự khoảng trắng dính nhau trong chuỗi thì xóa bớt 1 khoảng trắng.
- Cho tất cả thành chữ thường.
- Cập nhật ký tự đầu tiên và các ký tự mà trước nó là khoảng trắng thành chữ hoa.

Chương trình:

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

// Nên định nghĩa MAX để có thể thay đổi kích thước mảng nhanh chóng
#define MAX 200

// Nên đặt prototype ở đây để biết hàm nhận vào cái gì và trả về cái gì
// Đầu vào: Chuỗi cần chuẩn hóa
// Đầu ra:  không có (chuỗi đưa vào được cập nhật sau khi chuẩn hóa)
void ChuanHoaChuoi(char []);

void main()
{
	char str[MAX];
```

```
	printf("Nhap ho ten: ");
	// Nên sử dụng gets thay vì scanf để có thể nhập chuỗi có khoảng trắng
	gets(str);

	ChuanHoaChuoi(str);

	printf("Chuoi sau khi chuan hoa: %s", str);
	getch();
}

void ChuanHoaChuoi(char str[])
{
	int i;

	// Xóa các khoảng trắng dư bên trái
// Nếu ký tự đầu tiên vẫn là khoảng trắng thì dịch chuyển chuỗi qua trái 1 ký tự
	while (str[0]==' ')
	{
		for (i=0; i<strlen(str)-1; i++)
			str[i] = str[i+1];
		str[strlen(str)-1] = '\0';// Cập nhật lại ký tự kết thúc chuỗi lùi lại
	}

	// Xóa các khoảng trắng dư bên phải
// Nếu ký tự cuối cùng vẫn là khoảng trắng thì dời lùi ký tự kết thúc chuỗi 1 ký tự
	while (str[strlen(str)-1]==' ')
	{
		str[strlen(str)-1] = '\0';
	}

	// Xóa các khoảng trắng dư ở giữa
	// Nếu tìm thấy khoảng trắng và ký tự kế tiếp cũng là khoảng trắng
	// Thì xóa khoảng trắng thừa đó (dời chuỗi qua trái 1 ký tự tại khoảng trắng
đó)
	i = 0;
	while (i<strlen(str)-1)
	{
		if (str[i]==' ')
			if (str[i+1]==' ')
			{
				for (int j=i+1; j<strlen(str)-1; j++)
					str[j] = str[j+1];
				str[strlen(str)-1] = '\0';
			}
			else
				i++;
		else
			i++;
	}

	// Biến tất cả ký tự của chuỗi thành chữ thường
	strlwr(str);

	// Chữ đầu tiên là chữ hoa
	str[0] = str[0]-32;

	// Chữ sau khoảng trắng sẽ được đổi thành chữ hoa
	for (i=0; i<strlen(str)-1; i++)
		if (str[i]==' ')
			str[i+1] = str[i+1]-32;
}
```

Lưu ý:
- Có thể viết 1 hàm xóa 1 ký tự tại vị trí x trong chuỗi để sử dụng chung (xem như Bài tập)
- Ký tự kết thúc chuỗi là ký tự '\0'
- Do ký tự thường đứng sau ký tự hoa 32 ký tự trong bảng max ASCII nên muốn đổi ký tự thường sang hoa thì ta trừ giá trị của ký tự thường với 32. Đơn giản hơn là sử dụng hàm toupper trong thư viện ctype.h.

4. Viết chương trình nhập một số nguyên, xuất lại số đó ở dạng chuỗi nhưng có dấu "," ngăn cách hàng triệu, ngàn.

```
Ví dụ:
Nhập: N = 123456789
Xuất: S = "123,456,789"
```

Ý tưởng:
Chuyển số N sang chuỗi S. Đi lùi từ cuối chuỗi và cứ 3 ký tự sẽ chèn dấu phẩy vào.

Chương trình:

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

// Nên định nghĩa MAX để có thể thay đổi kích thước mảng nhanh chóng
#define MAX 20

// Nên đặt prototype ở đây để biết hàm nhận vào cái gì và trả về cái gì
// Đầu vào: Chuỗi cần chèn, ký tự chèn, vị trí
// Đầu ra:  không có (chèn trực tiếp vào chuỗi đưa vào)
void ChenKyTuVaoChuoi(char [], char, int);

void main()
{
	unsigned int N;
	printf("Nhap mot so N: ");
	scanf("%d", &N);

	char str[MAX];
	itoa(N, str, 10); // Đổi số N sang chuỗi str cơ số 10

	for (int i=strlen(str)-3;i>0;i=i-3)
		ChenKyTuVaoChuoi(str, ',', i);		// Gọi hàm chèn ký tự , vào chuỗi

	printf("Chuoi so sau khi xu ly la: %s", str);
	getch();
}

void ChenKyTuVaoChuoi(char str[], char chr, int pos)
{
	int length = strlen(str);

	for (int i=length; i>pos; i--)		// Đẩy chuỗi sang phải sau vị trí pos
		str[i] = str[i-1];

	str[pos] = chr;				// Đưa ký tự chr vào vị trí pos
	str[length+1] = '\0';			// Cập nhật vị trí kết thúc chuỗi
}
```

5. Nhập 2 chuỗi S1 và S2 chỉ gồm các ký số từ bàn phím. Xuất ra tổng của 2 số đó.

```
Ví dụ:
Nhập: S1 = “2912” S2 = “176”
Xuất: S = “3088”
```

Ý tưởng 1:
Đổi 2 chuỗi sang 2 số. Cộng 2 số đó rồi đổi ngược sang chuỗi. Cách này chỉ làm trong trường hợp chuỗi số ngắn ➔ biến số còn đủ sức chứa.

Ý tưởng 2:
Thực hiện phép cộng như cộng bằng tay. Để tiện ta thêm các số 0 vào đầu chuỗi số ngắn để 2 chuỗi số dài bằng nhau.

Chương trình:

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

#define MAX 200

void main()
{
	char a[MAX];
	char b[MAX];
	char c[MAX];

// Chuỗi số này liên tục nên ta sử dụng ngay hàm scanf thay vì hàm gets
	printf("Nhap so thu nhat: ");
	scanf(“%s”, &a);
	printf("Nhap so thu hai: ");
	scanf(“%s”, &b);

/*
Chèn số 0 vào đầu chuỗi ngắn hơn
	Đối với chuỗi ngắn hơn, ta thực hiện 3 bước:
	- Đảo chuỗi
	- Thêm số 0 vào đến khi độ dài chuỗi ngắn = chuỗi dài
	- Đảo chuỗi lại
	*/
	if (strlen(a)<strlen(b))
	{
		strrev(a);
		while (strlen(a)<strlen(b))
			strcat(a, "0");
		strrev(a);
	}
	else
	{
		strrev(b);
		while (strlen(b)<strlen(a))
			strcat(b, "0");
		strrev(b);
	}

	int na, nb, nc, clength = 0;
	int temp = 0;
	for (int i=strlen(a)-1; i>=0; i--)
	{
		na = a[i] - 48;   // Đổi ký tự số sang số ta trừ 48. Ví dụ: ‘1’ - 48 = 1
		nb = b[i] - 48;
		nc = na + nb + temp;
```

```
            temp = nc/10;      // Lấy phần nhớ
            nc = nc%10;
            c[clength++] = nc + 48;
      }
      if (temp>0)
            c[clength++] = temp + 48;      // Nếu còn nhớ thì đưa tiếp vào
      c[clength] = '\0';                   // Kết thúc chuỗi

      strrev(c);
      printf(" %s + %s = %s", a, b, c);
      getch();
}
```

Nếu không thêm các số 0 vào đầu chuỗi số ngắn, ta có thể làm như sau:

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

#define MAX 200

void main()
{
      char a[MAX];
      char b[MAX];
      char c[MAX];
      int temp;

      printf("Nhap so thu nhat: ");
      scanf("%s", &a);
      printf("Nhap so thu hai: ");
      scanf("%s", &b);

      strrev(a);
      strrev(b);

      int ablength;
      ablength = strlen(a) < strlen(b) ? strlen(b) : strlen(a);
      /*
Câu lệnh trên tương đương với câu lệnh if sau (xem lại toán tử 3 ngôi):
if (strlen(a) < strlen(b))
      ablength = strlen(b);
else
      ablength = strlen(a);
      /*

      int na, nb, nc, clength = 0;
      temp = 0;
      for (int i=0; i<ablength; i++)
      {
            na = i < strlen(a) ? a[i]-48 : 0;
            nb = i < strlen(b) ? b[i]-48 : 0;

            nc = na + nb + temp;
            temp = nc/10;
            nc = nc%10;

            c[clength++] = nc + 48;
      }

      if (temp>0)
            c[clength++] = temp + 48;
      c[clength] = '\0';
```

```
        strrev(a);
        strrev(b);
        strrev(c);

        printf("%s + %s = %s", a, b, c);
        getch();
}
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Nhập họ tên của một người từ bàn phím. Hãy chuẩn hóa chuỗi họ tên này. (Xóa các khoảng trắng thừa và ký tự đầu tiên của họ, chữ lót và tên phải viết hoa, các ký tự còn lại viết thường). Ví du:

   Nhập: " NgUyen VaN A "

   Xuất: "Nguyen Van A"
2. Không sử dụng các hàm có sẵn. Viết chương trình xóa N ký tự tại vị trí i trong chuỗi S.

   Ví dụ:

   Nhập: S = "Nguyen Van A" i = 2 N = 3 (Xóa 3 ký tự tại ký tự 2 trong chuỗi S)

   Xuất: S = "Nen Van A"
3. Viết chương trình nhập một số nguyên, xuất lại số đó ở dạng chuỗi nhưng có dấu "," ngăn cách hàng triệu, ngàn.

   Ví dụ:

   Nhập: N = 123456789

   Xuất: S = "123,456,789"
4. Nhập một chuỗi S từ bàn phím. Kiểm tra xem chuỗi có phải là chuỗi đối xứng.

   Ví dụ:

   Nhập: S = "aBCdCBa"

   Xuất: Đối xứng

   Nhập: S = "aBCdBCa"

   Xuất: Không dối xứng
5. Nhập một chuỗi S từ bàn phím. Tìm ký tự xuất hiện nhiều nhất trong chuỗi đó và số lần xuất hiện.

   Ví dụ:

   Nhập: S = "Nguyen Thi B"

   Xuất: n 2 lần
6. Nhập một chuỗi S từ bàn phím và một ký tự C. Đếm xem ký tự C xuất hiện bao nhiêu lần trong chuỗi S đó.

   Nhập: "Nguyen Van A" C = 'u'

Xuất: 1 lần

7. Lập trình nhập vào từ bàn phím danh sách học sinh một lớp, sắp xếp lại danh sách theo thứ tự abc của Tên, nếu trùng Tên thì sắp xếp theo thứ tự abc của Họ.
8. Viết chương trình nhập từ bàn phím 2 xâu ký tự S1 và S2. Hãy xét xem S1 có xuất hiện bao nhiêu lần trong S2 (hoặc ngược lại S2 xuất hiện bao nhiêu lần trong S1) và tại những vị trí nào?
9. Viết chương trình nhập một xâu S chỉ gồm các chữ cái thường. Hãy lập xâu S1 nhận được từ xâu S bằng cách sắp xếp lại các ký tự theo thứ tự abc.
10. Cho xâu S chỉ gồm các dấu "(" và ")". Hãy kiểm tra xem S có là một biểu thức ( ) hợp lệ hay không.

    Lập trình tính giá trị của một số viết dưới dạng LA MÃ.
11. Ví dụ: MDCLXVI = 1666. M:1000 ; D:500 ; C:100; L:50; X :10 ; V:5 ; I :1

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Không sử dụng các hàm có sẵn. Viết chương trình chèn chuỗi S2 vào chuỗi S1 tại vị trí i trong chuỗi S1.

   Ví dụ:

   Nhập: S1 = "Nguyen Van A"

   S2 = "Le "

   i = 8 (Chèn chuỗi S2 vào chuỗi S1 tại vị trí 8)

   Xuất: S1 = "Nguyen LeVan A"
2. Nhập 2 chuỗi S1 và S2 chỉ gồm các ký số từ bàn phím. Xuất ra tổng của 2 số đó.

   Ví dụ:

   Nhập: S1 = "2912" S2 = "176"

   Xuất: S = "3088"
3. Không sử dụng các hàm có sẵn. Kiểm tra xem chuỗi S2 có nằm trong chuỗi S1 hay không và tại vị trí nào.

   Ví dụ:

   Nhập: S1 = "Nguyen Van A" S2 = "Van"

   Xuất: S2 nằm trong S1 tại vị trí 8.
4. Nhập một chuỗi S từ bàn phím. Đếm xem có bao nhiêu từ có nhiều hơn n ký tự có trong chuỗi S.

   Nhập: "Nguyen Van A" n = 2

   Xuất: 2 từ
5. Cho xâu ký tự S. Xét xem có hay không một xâu X sao cho S là ghép của một số lần liên tiếp của X. Ví du S = abcabc thì X là abc, còn nếu S = abcab thì không có xâu X.

6. Nhập từ bàn phím một số nguyên dương N<=1000 và in ra màn hình xâu ký tự S độ dài N chỉ gồm các ký tự 0 và 1 sao cho S không có xâu con nào xuất hiện 3 lần liên tiếp trong nó.
   Nhập từ bàn phím hai số nhị phân X và Y. Hãy in ra màn hình số nhị phân Z có giá trị lớn nhất có thể được mà Z nhận được từ X bằng cách gạch đi một số chữ số nhị phân nào đó và Z cũng nhận được từ Y bằng cách gạch đi một số chữ số nhị phân nào đó.

7. Cho một số N rất lớn (không thể biểu diễn dưới dạng thập phân một các thông thường) được biểu diễn dưới dạng chuỗi nhị phân. Hãy tìm phần dư của phép chia N cho 15 trong hệ thập phân.
8. Cho N xâu ký tự A1, A2, …, AN, N<=100, độ dài của xâu Ai không quá 10, và một xâu ký tự S. Hãy tìm mọi cách biểu diễn S dưới dạng ghép của các xâu ký tự Ai, mỗi xâu Si có thể xuất hiện trong biểu diễn dó nhiều lần.
9. Xâu M gọi là xâu con của S nếu ta có thể nhận được M từ S bằng cách xóa đi một số ký tự của S. Cho hai xâu S1, S2, hãy tìm xâu con M dài nhất vừa là xâu con của S1, vừa là xâu con của S2.

Tuần 13. ĐỆ QUY

CÁC BÀI TẬP CƠ BẢN

1. Nhập một số nguyên n. Sử dụng đệ quy tính n! (n! = 1*2*…*n)

```
Ví dụ:
Nhập: n = 0
Xuất: 0! = 1
Nhập: n = 4
Xuất: 4! = 24
```

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

// Nên đặt prototype ở đây để biết hàm nhận vào cái gì và trả về cái gì
// Đầu vào: n
// Đầu ra:  n!
int GiaiThua(int);

void main()
{
	int n;

	printf("Nhap n: ");
	scanf("%d", &n);

	int kq = GiaiThua(n);

	printf("%d! = %d", n, kq);
	getch();
}

int GiaiThua(int n)
{
	if (n==0)	// Rất quan trọng. Đây là điểm thoát của hàm đệ quy.
		return 1;
	else		// Có thể bỏ else vì nếu rơi vào trường hợp n=0 thì đã return.
		return n*GiaiThua(n-1);
}
```

Cần chú ý sức chứa của biến khi khai báo. Hàm GiaiThua trả về kết quả kiểu **int** có sức chứa không đủ trong các trường hợp n lớn. Lúc này sẽ xảy ra hiện tượng tràn số và dẫn đến cho ra kết quả sai. Có thể thay kiểu trả về int bằng kiểu khác có sức chứa lớn hơn như **float** hoặc **double**.

2. Nhập một số nguyên n. Sử dụng đệ quy in dãy Fibonacy có độ dài n.

```
Ví dụ:
Nhập: n = 2
Xuất: 1 1
Nhập: n = 5
Xuất: 1 1 2 3 5
```

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

// Nên đặt prototype ở đây để biết hàm nhận vào cái gì và trả về cái gì
// Đầu vào: n
// Đầu ra:  phần tử thứ n của dãy Fibonacy
int Fibonacy(int);

void main()
{
	int n;

	printf("Nhap n: ");
	scanf("%d", &n);

	for (int i=0; i<n; i++)
		printf("%4d", Fibonacy(i));
	getch();
}

int Fibonacy(int n)
{
	if (n==0 || n==1) // Rất quan trọng. Đây là điểm thoát của hàm đệ quy.
		return 1;
	else  // Có thể bỏ else vì nếu rơi vào trường hợp n=0 hoặc n = 1 thì đã return
		return Fibonacy(n-1) + Fibonacy(n-2);
}
```

3. Nhập một mảng n số nguyên. Sử dụng đệ quy tính tổng giá trị các phần tử có trong mảng.

```
Ví dụ:
Nhập: n = 1 và mảng a = 4
Xuất: Tổng = 4
Nhập: n = 5 và mảng a = -1 4 6 3 0
Xuất: Tổng = 12
```

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

#define MAX 100

// Đầu vào: biến chứa mảng các số nguyên và số phần tử
// Đầu ra:  tổng các phần tử của mảng
int TongMang(int [], int);

void main()
{
	int a[MAX];
	for (i=0; i<n; i++)
	{
		printf("Nhap phan tu a[%d]: ", i);
		scanf("%d", &a[i]);
	}

	if (n>0)
	{
		printf("Mang a vua nhap la: ");
		for (i=0; i<n; i++)
			printf("%4d", a[i]);
	}

	int kq = TongMang(a, n);
	printf("Tong cac phan tu cua mang bang %d", kq);
	getch();
}

int TongMang(int a[], int n)
{
	if (n==0)   // Đây là điểm thoát của hàm đệ quy khi mảng rỗng.
		return 0;
	else  // Mảng không rỗng thì lấy phần tử cuối + tổng phần đầu
		return a[n-1] + TongMang(a, n-1);
}
```

4. Nhập 2 số nguyên dương a và b. Sử dụng đệ quy tính ước số chung lớn nhất của 2 số đó.

```
Nhập: a = 3 b = 4
Xuất: USCLN = 12
```

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

// Đầu vào: 2 số nguyên dương
// Đầu ra:  ước số chung lớn nhất của 2 số nguyên dương đó
int USCLN(int, int);

void main()
{
	int a, b;

	printf("Nhap 2 so nguyen a va b: ");
	scanf("%d%d", &a, &b);

	int c = USCLN(a,b);
	printf("Uoc so chung lon nhat cua %d va %d la: %d", a, b, c);
	getch();
}

int USCLN(int a, int b)
{
	if (a==b)   // Đây là điểm thoát của hàm đệ quy a=b
		return a;
	if (a>b)
		return USCLN(a-b, b);
	return USCLN(a, b-a);
}
```

5. Nhập số nguyên dương N. Sử dụng đệ quy tính in dãy nhị phân của số N đó.

```
Nhập: N = 5
Xuất: 101
```

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

// Đầu vào: số nguyên n
// Đầu ra:  dãy biểu diện nhị phân của số nguyên dương n
void InNhiPhan(int);

void main()
{
	int n;

	printf("Nhap n: ");
	scanf("%d", &n);

	InNhiPhan(n);
	getch();
}

void InNhiPhan(int n)
{
	if (n/2==0) // Đây là điểm thoát của hàm đệ quy khi không thể chia n tiếp
		printf("%d", n%2);
	else		// Muốn bỏ else ở đây thì phải them return; ở điều kiện n/2==0
	{
		InNhiPhan(n/2);   // Chú ý thứ tự in: in phần còn lại trước
		printf("%d", n%2);	// … rồi in số dư vừa tìm được
	}
}
```

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ TRUNG BÌNH

1. Nhập một số nguyên N. Tính N!

   Nhập: N = 4

   Xuất: N! = 24

2. Nhập 2 số nguyên a và b. Sử dụng đệ quy tính ước số chung lớn nhất của 2 số đó.

   Nhập: a = 3 b = 4

   Xuất: USCLN = 12

3. Nhập một mảng gồm N số nguyên. Sử dụng đệ quy tính tổng N số nguyên đó.

   Nhập: [1, 5, 0, 6]

   Xuất: S = 12

4. Nhập một mảng gồm số N số nguyên. Sử dụng đệ quy kiểm tra xem có phải là mảng tăng dần.

   Nhập: [0, 1, 5, 6]

   Xuất: Tăng dần

5. Nhập một mảng gồm số N số nguyên. Sử dụng đệ quy kiểm tra xem có phải là mảng đối xứng.

   Nhập: [0, 1, 5, 1, 0]

   Xuất: Mảng đối xứng

6. Nhập một số N từ bàn phím. In ra số đó theo thứ tự ngược lại.

   Nhập: N = 1234

   Xuất: N = 4321

7. Nhập một số nguyên N từ bàn phím. Tính giá trị biểu thức:

   $S = \sqrt{2*N + \sqrt{2*(N-1) + ... + \sqrt{4+\sqrt{2}}}}$

   Nhập: N = 4

   Xuất: S = 3,299 (S = $\sqrt{8+\sqrt{6+\sqrt{4+\sqrt{2}}}}$)

8. Nhập n.

   Tính $S = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ...(-1)^n \frac{x^{2n+1}}{(2n+1)!}$.

   Kiểm tra kết quả tìm được khi n lớn với sinx

9. Nhập n. Tính $S = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + ...(-1)^n \frac{x^{2n}}{(2n)!}$.

   Kiểm tra kết quả tìm được khi n lớn với cosx

10. QuickSort
11. Xây dựng một dãy gồm N ký tự từ 3 ký tự 1, 2, 3 sao cho không có hai dãy con liên tiếp nào giống nhau. Ví dụ N = 5: 12321 (đúng). Các dãy 12323, 12123 là không đúng.

CÁC BÀI TẬP THÊM CÓ ĐỘ KHÓ CAO

1. Cho n số tự nhiên x1, x2,…, xn. Hãy tìm UCLN (x1, x2,…,xn) bằng cách sử dụng: UCLN(x1,x2,…,xn)= UCLN(UCLN(x1,x2,…,x(n-1)),xn)
2. Bài toán mã đi tuần
3. Bài toán 8 quân hậu
4. Tìm tất cả các hoán vị của một mảng có n phần tử
5. Dãy Fibonacci
6. Cho n quả cân có trọng lượng m1, m2,…,mn. Hãy tìm cách đặt một số quả cân sao cho cân được cân bằng.
7. Cho 1 va li có thể tích V và có n đồ vật có các giá trị a1, a2,…, an và thể tích tương ứng là V1, V2,…,Vn. Hãy tìm cách xếp các đồ vật vào vali sao cho giá trị của các hàng hóa là cao nhất