حل مسائل ساختمان داده با C++

تالیف:

دکتر رمضان عباس نژاد ورزی

فن‌آوری نوین

سرشناسه	: ۱۳۴۸ -عباس نژاد، رمضان،
عنوان و نام پدیدآور	: حل مسائل ساختمان داده با C++/ تالیف رمضان عباس‌نژادورزی.
مشخصات نشر	: بابل: فناوری نوین، ۱۳۹۷
مشخصات ظاهری	: ۳۶۸ ص.: مصور، جدول.
شابک	: ۵۷۰۰۰۰ ریال: 978-600-7272-33-6
وضعیت فهرست نویسی	: .کتابنامه:ص.۳۶۴
موضوع	: سی ++ (زبان برنامه‌نویسی کامپیوتر) -- مسائل، تمرین‌ها و غیره (عالی)
موضوع	: C++ (Computer program language) -- Problems, exercises, etc. (Higher)
موضوع	: سی ++ (زبان برنامه‌نویسی کامپیوتر) -- راهنمای آموزشی (عالی)
موضوع	: C++ (Computer program language) -- Study and teaching (Higher)
رده بندی کنگره	: QA۷۶/۷۳
رده بندی دیویی	: ۰۰۵/۱۳۳
شماره کتابشناسی ملی	: ۵۸۱۱۲۳۱

فن‌آوری نوین www.fanavarinovin.net

بابل، صندوق پستی ۴۷۱۶۷-۷۳۴۴۸ **تلفن: ۳۲۲۵۶۶۸۷-۰۱۱**

حل مسائل ساختمان داده با C++

تألیف: دکتر رمضان عباس‌نژادورزی

ناشر: فن‌آوری نوین

چاپ اول: تابستان ۱۳۹۸

جلد: ۲۰۰

شابک: 978-600-7272-33-6

حروفچینی و صفحه‌آرایی: فن‌آوری نوین قیمت: ۵۷۰۰۰ تومان

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ تلفکس: ۶۶۴۰۰۱۴۴-۶۶۴۰۰۲۲۰

فهرست مطالب

مقدمه

ساختمان داده، یکی از مباحث بسیار مهم در دروس مهندسی کامپیوتر است. کتاب‌های زیادی در زمینه ساختمان داده با زبان C++ ترجمه و تالیف شده است که جای تقدیر و تشکر دارد. اما، جای کتاب حل مسائل ساختمان داده با C++ که بتواند الگوریتم‌های مختلف ساختمان داده را پیاده‌سازی کند، خالی است. کتاب حاضر الگوریتم‌های ساختمان داده را به زبان C++ پیاده‌سازی می‌کند.

این کتاب شامل ۷ فصل است که عبارت‌اند از:

فصل اول، تمرین‌های مباحثی نظیر **ساختار داده‌ها و الگوریتم‌های بازگشتی** را حل کرده است.

فصل دوم، مسائل مباحثی مانند **ماتریس‌های یک بعدی، دو بعدی و ماتریس‌های اسپارس** را پیاده‌سازی کرده است.

فصل سوم، تمرین‌های مباحثی مانند **صف و پشته** را حل نموده است.

فصل چهارم، مسائل **لیست‌های پیوندی** را پیاده‌سازی می‌کند.

فصل پنجم، تمرین‌های **درخت‌ها** را حل نمود.

فصل ششم، مسائل **گراف‌ها** را پیاده‌سازی کرده است.

فصل هفتم، مسائل **جست‌وجو و مرتب‌سازی** را پیاده‌سازی کرده است.

برای آموزش ساختمان داده با C++ می‌توانید به ساختمان داده با C++ از همین انتشارات مراجعه کنید.

از تمامی اساتید و دانشجویان عزیز تقاضا داریم، هرگونه اشکال، ابهام در متن کتاب، پیشنهاد و انتقادات را به آدرس پست الکترونیک **fanavarienovin@gmail.com** ارسال نمایند.

در پایان امیدوارم این اثر موردتوجه جامعه انفورماتیک کشور، اساتید و دانشجویان عزیز قرار گیرد.

مولفین

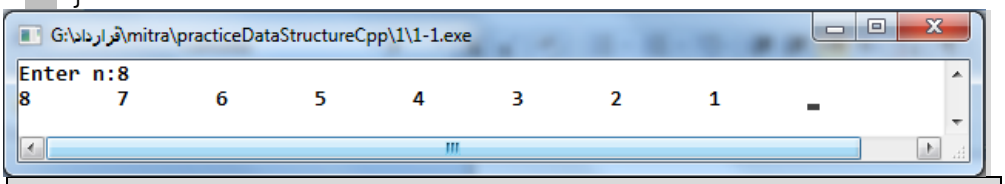
fanavarienovin@gmail.com

فصل ۱

ساختار داده‌ها و الگوریتم‌های بازگشتی

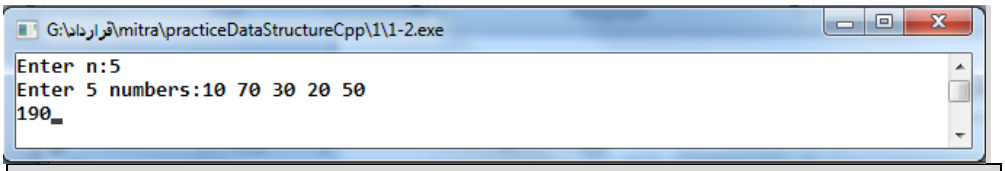
مثال ۱– ۱. تابع بازگشتی که N را به‌عنوان پارامتر دریافت می‌کند و از n تا یک را نمایش می‌دهد:

```
#include <iostream.h>
#include <conio.h>
void display(int n) {
   if (n == 0)
       return;
   else {
     cout << n <<  "\t";
     display(n-1);
   }
}
void main() {
    int n;
    cout << "Enter n:";
    cin >> n;
    display(n);
    getch();
}
```



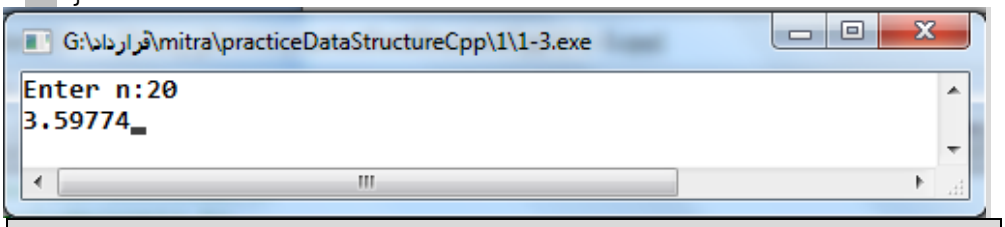
مثال ۲–۱. تابع بازگشتی که مجموع عناصر یک آرایه را برمی‌گرداند.

```
#include <iostream.h>
#include <conio.h>
int listSum(int numList[], int n) {
    if (n == 0)
       return numList[0];
    else
       return numList[n-1] + listSum(numList, n - 1);
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   int *a = new int[n];
   cout << "Enter " << n << " numbers:";
   for(int i = 0; i < n; i++)
      cin >> a[i];
   cout << listSum(a, n);
   getch();
}
```



مثال ۳–۱. تابع بازگشتی که حاصل عبارت زیر را محاسبه می‌کند و برمی‌گرداند.

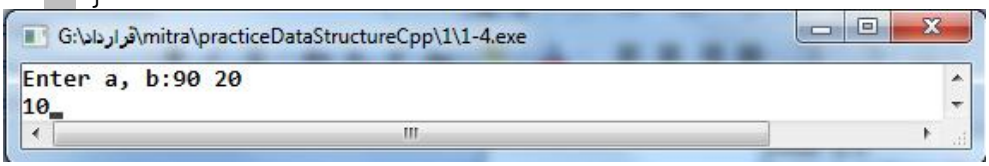
```
#include <iostream.h>
#include <conio.h>
float harmonicSum(int n) {
    if (n < ۲ )
       return ۱;
    else
       return (۱.۰ / n + harmonicSum(n - ۱));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   cout << harmonicSum(n);
   getch();
}
```



مثال ۴–۱. تابع بازگشتی که بزرگ‌ترین مقسوم‌علیه مشترک بین دو عدد را برمی‌گرداند.

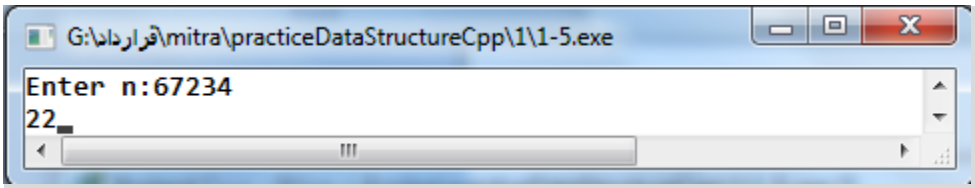
```
#include <iostream.h>
#include <conio.h>
int min(int a, int b) {
  return (a < b ? a : b );
}
int max(int a, int b) {
  return (a > b ? a : b );
}
int gcd(int a, int b) {
    int low = min(a, b);
    int high = max(a, b);
    if (low == 0)
       return high;
    else if (low == 1)
       return 1;
    else
       return gcd(low, high % low);
}
void main() {
```

```
    int a, b;
    cout << "Enter a, b:";
    cin >> a >> b;
    cout <<  gcd(a, b);
    getch();
}
```



مثال ۵–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، مجموع ارقام آن را برمی‌گرداند.

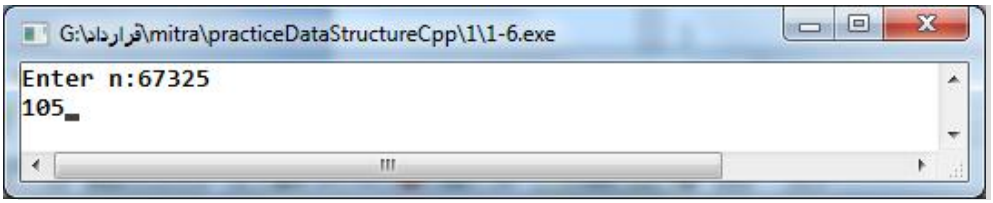
```
#include <iostream.h>
#include <conio.h>
int  sumDigit(int n) {
   if (n == 0)
        return 0;
   else
        return (n % 10 + sumDigit(n / 10));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   cout <<  sumDigit(n);
   getch();
}
```



مثال ۶–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، حاصل‌ضرب ارقام فرد آن را برمی‌گرداند.

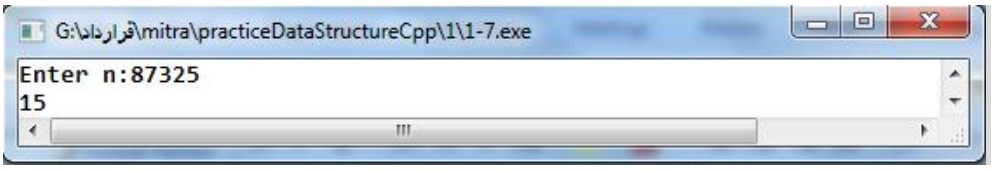
```
#include <iostream.h>
#include <conio.h>
int mulOddDigit(int n) {
   if  (n == 0)
        return 1;
   else if (n % 10 % 2 == 1 )
        return (n % 10 * mulOddDigit(int(n / 10)));
   else
        return (mulOddDigit(n / 10));
}
```

```
void main() {
    int n;
   cout << "Enter n:";
   cin >> n;
   cout <<  mulOddDigit(n);
   getch();
}
```



مثال ۷–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، حاصل جمع ارقام بالای ۶ آن را برمـــی-گرداند.

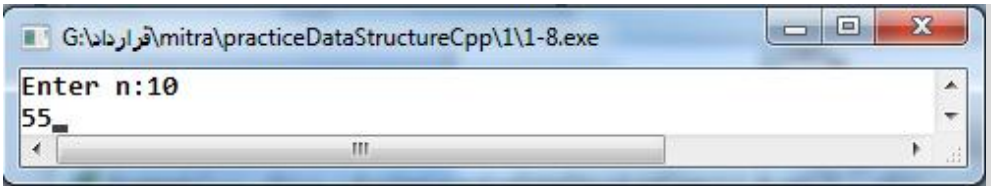
```
#include <iostream.h>
#include <conio.h>
int sumDigitG6(int n) {
   if  (n == 0 )
        return 0;
   else if (n % 10 > 6 )
         return (n % 10 + sumDigitG6(n / 10));
   else
         return (sumDigitG6(int(n / 10)));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   cout <<  sumDigitG6(n);
   getch();
}
```



مثال ۸–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، nامین جمله فیبوناچی را برمی‌گرداند.

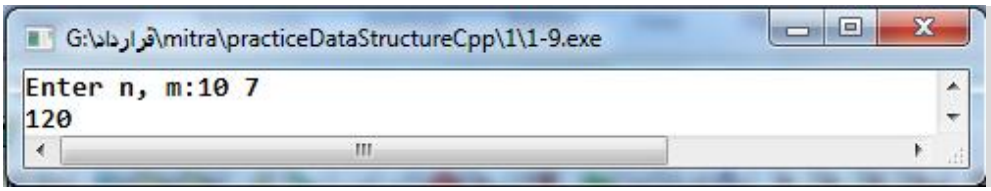
```
#include <iostream.h>
#include <conio.h>
int Fibo(int n) {
   if (n == 1 || n == 2 )
        return 1;
```

```
   else
        return (Fibo(n -1) + Fibo(n - 2));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   cout <<  Fibo(n);
   getch();
}
```



مثال ۹–۱. تابع بازگشتی که m و n را به‌عنوان پارامتر دریافت کرده، $\binom{n}{m} = C(n,\ m)$ را برمی‌گرداند.

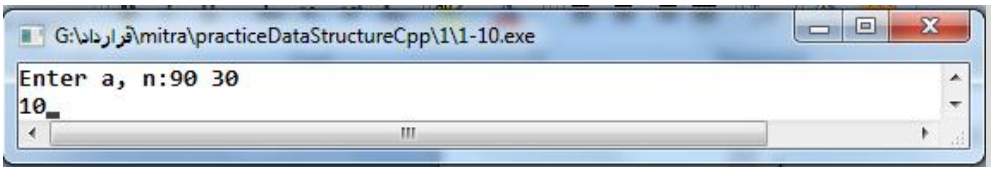
```
#include <iostream.h>
#include <conio.h>
int  C(int n, int m) {
   if (n == m  ||  m == 0)
        return 1;
   else
        return (C(n -1, m -1) + C(n -1, m));
}
void main() {
   int n, m;
   cout << "Enter n, m:";
   cin >> n >> m;
   cout <<  C(n, m);
   getch();
}
```



مثال ۱۰–۱. تابع بازگشتی که a و n را به‌عنوان پارامتر دریافت کرده، حاصل عبارت $\sqrt{a+\sqrt{a+\sqrt{a+\cdots}}}$ را برای n مرتبه برمی‌گرداند.

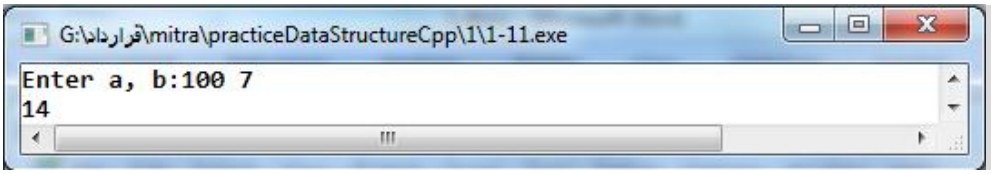
```
#include <iostream.h>
#include <conio.h>
#include <math.h>
double  S( double a, double n) {
   if (n == 1)
```

```
        return sqrt(a);
    else
        return (sqrt(a +  S(a, n -1)));
}
void main() {
    double a, n;
    cout << "Enter a, n:";
    cin >> a >> n;
    cout <<  S(a, n);
    getch();
}
```



مثال ۱۱–۱. تابع بازگشتی که a و b را به‌عنوان پارامتر دریافت کرده، a تقسیم بر b را برمی‌گرداند.

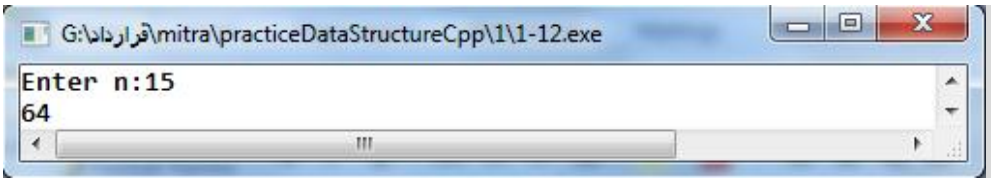
```
#include <iostream.h>
#include <conio.h>
int Div(int a, int b) {
    if (a < b)
        return 0;
    else
        return (1 + Div(a - b, b));
}
void main() {
    int a, b;
    cout << "Enter a, b:";
    cin >> a >> b;
    cout <<  Div(a, b);
    getch();
}
```



مثال ۱۲–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، حاصل جمع اعداد فرد ۱ تا n را برمی‌گرداند.

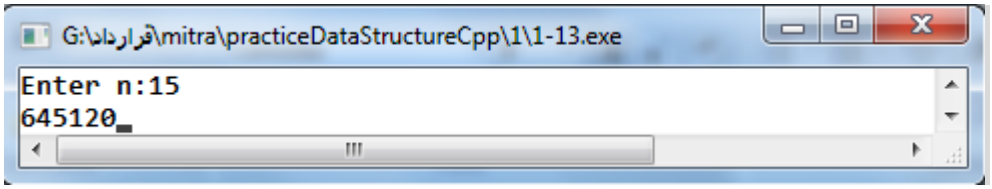
```
#include <iostream.h>
#include <conio.h>
int sumOdd (int n) {
    if (n < 1)
        return 0;
```

```
   else if (n % 2 == 1)
        return (n + sumOdd(n - 2));
   else
        return (sumOdd(n - 1));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   cout <<  sumOdd(n);
   getch();
}
```



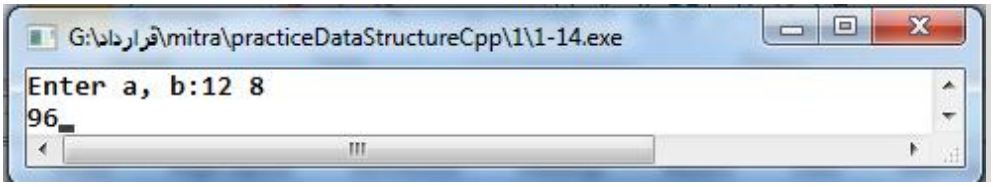
مثال ۱۳–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، حاصل‌ضرب اعداد زوج ۲ تا n را برمی‌گرداند.

```
#include <iostream.h>
#include <conio.h>
long  MulEven(int n) {
   if (n < 2)
        return 1;
   else if (n % 2 == 0)
        return (n * MulEven(n - 2));
   else
        return (MulEven(n - 1));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   cout <<  MulEven(n);
   getch();
}
```



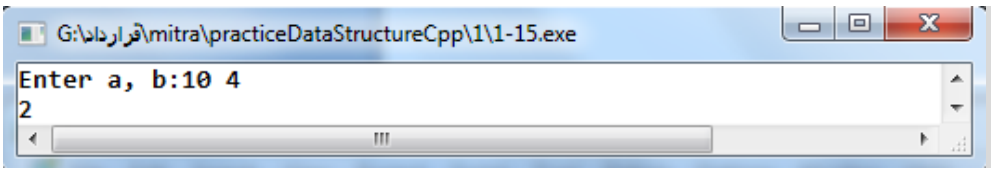
مثال ۱۴–۱. تابع بازگشتی که اعداد صحیح a و b بزرگ‌تر از صفر را دریافت کرده، a ضرب در b را برمی‌گرداند.

```
#include <iostream.h>
#include <conio.h>
int  Mul(int a, int b) {
   if (b == ۱)
        return a;
   else
        return (a + Mul(a, b - ۱));
}
void main() {
       int a, b;
       cout << "Enter a, b:";
   cin >> a >> b;
       cout <<  Mul(a, b);
   getch();
}
```



مثال ۱۵–۱. تابع بازگشتی که a و b (اعداد صحیح) را به‌عنوان پارامتر دریافت کرده، باقی‌مانده تقسیم صحیح a بر b را برمی‌گرداند.

```
#include <iostream.h>
#include <conio.h>
int  Mod(int a, int b) {
   if (a < b)
        return a;
   else
        return(Mod(a - b, b));
}
void main() {
   int a, b;
   cout << "Enter a, b:";
   cin >> a >> b;
   cout <<  Mod(a, b);
   getch();
}
```

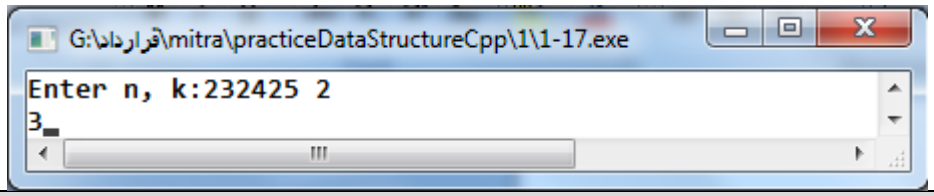


مثال ۱۶–۱. تابع بازگشتی که k و n را به‌عنوان پارامتر دریافت کرده، مجموع تمام اعداد مضرب k کوچک‌تر یا مساوی n را برمی‌گرداند.

```
#include <iostream.h>
#include <conio.h>
int MK(int n, int k) {
    if (n < k)
         return 0;
    else if (n % k == 0)
         return (n + MK(n - k, k));
    else
         return (MK(n - 1, k));
}
void main() {
   int n, k;
   cout << "Enter n, k:";
   cin >> n >> k;
   cout <<  MK(n, k);
   getch();
}
```

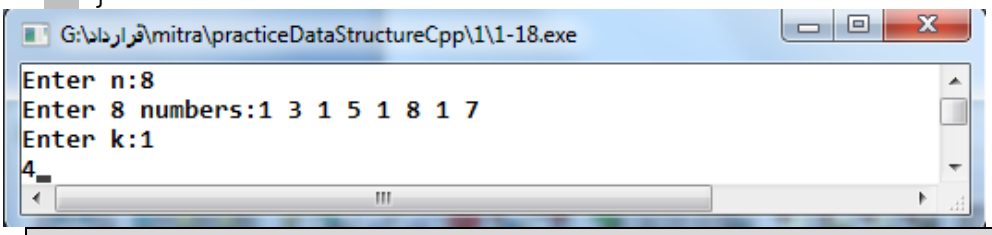
مثال ۱۷–۱. تابع بازگشتی که n و k را به‌عنوان پارامتر دریافت کرده، تعداد ارقام k آن را برمی‌گرداند.

```
 #include <iostream.h>
#include <conio.h>
int countK(int n, int k) {
    if (n == 0)
         return 0;
    else if ( n % 10 == k)
         return (1 + countK(n /  10, k));
    else
         return (countK(n / 10, k));
}
void main() {
   int n, k;
   cout << "Enter n, k:";
   cin >> n >> k;
   cout <<  countK(n, k);
   getch();
}
```



مثال ۱۸–۱. تابع بازگشتی که تعداد تکرار یک مقدار در یک آرایه را برمی‌گرداند.

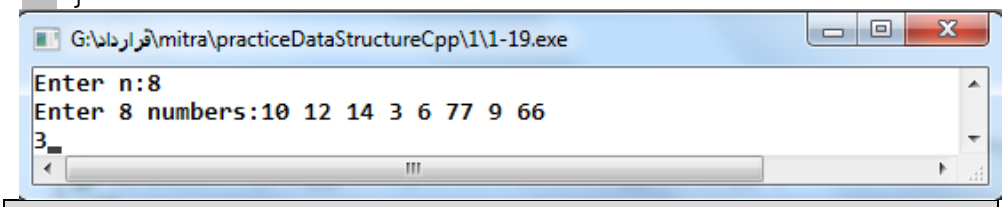
```
#include <iostream.h>
#include <conio.h>
int listCount(int numList[], int n, int k) {
    if (n == -1)
       return 0;
    else if ( numList[n-1] == k )
       return (1 + listCount(numList, n - 1, k));
    else
       return(listCount(numList, n - 1, k));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   int *a = new int[n];
   cout << "Enter " << n << " numbers:";
   for(int i = 0; i < n; i++)
      cin >> a[i];
   int k;
   cout << "Enter k:";
   cin >> k;
   cout << listCount(a, n, k);
   getch();
}
```



مثال ۱۹–۱. تابع بازگشتی که کوچک‌ترین عنصر یک آرایه را برمی‌گرداند.

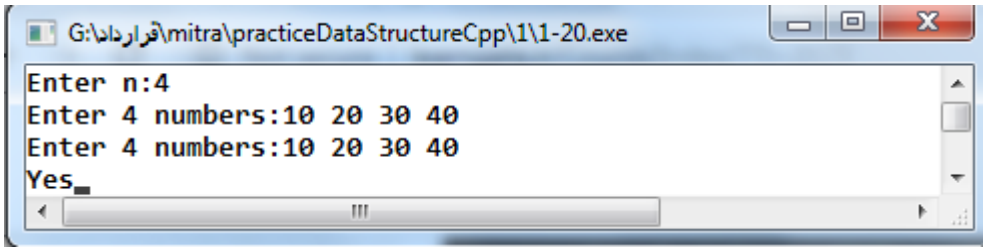
```
#include <iostream.h>
#include <conio.h>
int findMin(int numList[], int n) {
    static int min = numList[0];
    if (n == -1)
       return min;
    else if ( numList[n-1] < min ) {
       min =  numList[n-1];
       return ( findMin(numList, n - 1));
    }
    else
```

```
        return(findMin(numList, n - 1));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   int *a = new int[n];
   cout << "Enter " << n << " numbers:";
   for(int i = 0; i < n; i++)
   cin >> a[i];
   cout << findMin(a, n);
   getch();
}
```



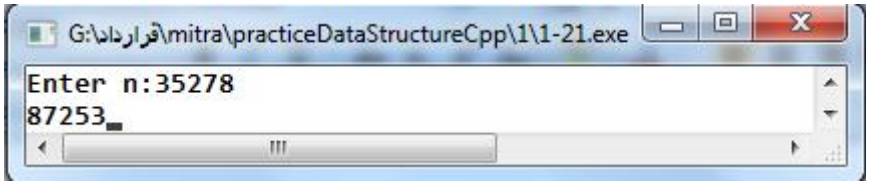
مثال ۲۰–۱. تابع بازگشتی که دو آرایه را با هم مقایسه کرده، نتیجه را برمی‌گرداند.

```
#include <iostream.h>
#include <conio.h>
bool equals(int list1[], int list2[], int n) {
     if (n == -1)
       return true;
    else if ( list1[n-1] !=  list2[n-1] )
       return false;
    else
       return(equals(list1, list2, n - 1));
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   int *a = new int[n];
   int *b = new int[n];
   cout << "Enter " << n << " numbers:";
   for(int i = 0; i < n; i++)
      cin >> a[i];
   cout << "Enter " << n << " numbers:";
   for(int i = 0; i < n; i++)
      cin >> b[i];
   if(equals(a, b, n))
     cout << "Yes";
   else
     cout << "No";
   getch();
}
```



مثال ۲۱–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، معکوس آن را برمی‌گرداند.

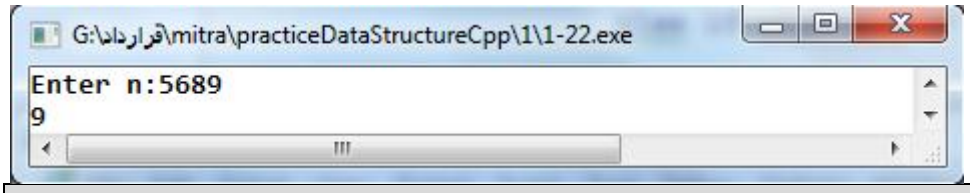
```
#include <iostream.h>
#include <conio.h>
long reverse(long n) {
  static long r = 0;
  if (n == 0)
     return 0;
  r = r * 10;
  r = r + n % 10;
  reverse(n / 10);
  return r;
}
void main() {
   int n;
   cout << "Enter n:";
   cin >> n;
   cout << reverse(n);
   getch();
}
```



مثال ۲۲–۱. تابع بازگشتی که n را به‌عنوان پارامتر دریافت کرده، بزرگ‌ترین رقم آن را برمی‌گرداند.

```
#include <iostream.h>
#include <conio.h>
long maxDigit(int n) {
  static int  max = -1;
  if (n == 0)
     return max;
  else if ( n % 10 > max ) {
     max = n % 10;
     maxDigit(n / 10);
  }
  else
    maxDigit(n / 10);
}
void main() {
   int n;
```

```
    cout << "Enter n:";
    cin >> n;
    cout << maxDigit(n);
    getch();
}
```

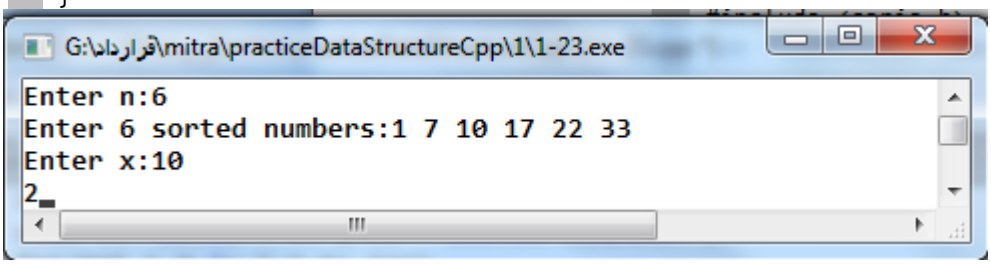


مثال ۲۳–۱. تابع بازگشتی که جست‌وجوی دودویی را در آرایه مرتب‌شده انجام می‌دهد.

```
#include <iostream.h>
#include <conio.h>
int BinarySearch(int nums[],int low , int high,int searchedNumber)
{
	if(low == high)
		if(nums[low] == searchedNumber)
			return low;
		else
			return -1;
	else
	{
		int leftAns = BinarySearch(nums,low,
		  (high + low) / 2, searchedNumber);
		int rightAns = BinarySearch(nums,
		  (high + low) / 2 + 1, high, searchedNumber);
		if(leftAns != -1)
			return leftAns;
		else if(rightAns != -1)
			return rightAns;
		else
			return -1;
	}
}

void main() {
```

```
    int n;
    cout << "Enter n:";
    cin >> n;
    int *a = new int[n];
    cout << "Enter " << n << " sorted numbers:";
    for(int i = 0; i < n; i++)
      cin >> a[i];
    int x;
    cout << "Enter x:";
    cin >> x;
    cout << BinarySearch(a, 0, n - 1, x);
    getch();
}
```



فصل ۲

ماتریس‌ها

مثال ۱ – ۲. برنامه‌ای که یک ماتریس را دریافت کرده، در جهت عقربه ساعت چرخش می‌دهد. اگر ماتریس به‌صورت زیر باشد:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

خروجی به‌صورت ماتریس زیر خواهد شد:

5	1	2	3
9	10	6	4
13	11	7	8
14	15	16	12

مراحل طراحی و اجرا:

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

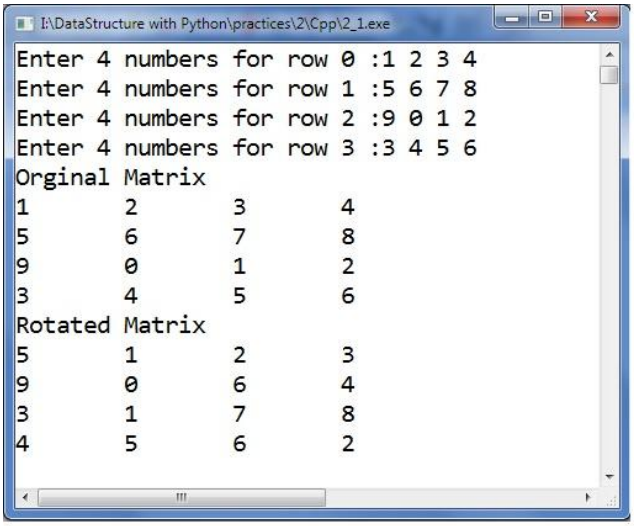
```
#include <iostream.h>
#include <conio.h>
#define  ROW  4
#define  COL  4
// A function to rotate a matrix mat[][] of size R x C.
// Initially, m = ROW and n = COL
void rotatematrix(int mat[][COL], int m, int n)
{
       int row = 0, col = 0;
       int prev, curr;
       /*     row - Staring row index
             m - ending row index
            col - starting column index
             n - ending column index
             i - iterator
       */
       while (row < m && col < n)
       {

              if (row + 1 == m || col + 1 == n)
                     break;
```

```
                // Store the first element of next row, this
                // element will replace first element of current row
                prev = mat[row + 1][col];
          /* Move elements of first row from the remaining rows */
                for (int i = col; i < n; i++)
                {
                        curr = mat[row][i];
                        mat[row][i] = prev;
                        prev = curr;
                }
                row++;
      /* Move elements of last column from the remaining columns */
                for (int i = row; i < m; i++)
                {
                        curr = mat[i][n-1];
                        mat[i][n-1] = prev;
                        prev = curr;
                }
                n--;
       /* Move elements of last row from the remaining rows */
                if (row < m)
                {
                        for (int i = n-1; i >= col; i--)
                        {
                                curr = mat[m-1][i];
                                mat[m-1][i] = prev;
                                prev = curr;
                        }
                }
                m--;
         /* Move elements of first column from the remaining rows */
                if (col < n)
                {
                        for (int i = m-1; i >= row; i--)
                        {
                                curr = mat[i][col];
                                mat[i][col] = prev;
                                prev = curr;
                        }
                }
                col++;
        }
}
void printMatrix(int arr[][COL], int row, int col)
{
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) cout<<arr[i][j]<<"\t";
            cout << '\n';
        }
}
void readMatrix(int a[][COL], int row, int col)
{
    for (int i = 0; i < row; i++) {
```

```
      cout << "Enter "<< col << " numbers for row " << i <<  " :";
              for (int j = 0; j < col; j++) cin >> a[i][j];
   }
}
int main()
{
   int a[ROW][COL];
   readMatrix(a, ROW, COL);
   cout << "Orginal Matrix\n";
   printMatrix(a, ROW, COL);
   rotatematrix(a, ROW, COL);
   cout << "Rotated Matrix\n";
   printMatrix(a, ROW, COL);
   getch();
   return 0;
}
```

۲. پروژه را ذخیره و اجرا کرده تا نمونه خروجی را به‌صورت زیر ببینید:



مثال ۲– ۲. برنامه‌ای که یک ماتریس را بدون استفاده از حافظه اضافی ۹۰ درجه چرخش می‌دهد. به‌عنوان‌مثال، اگر آرایه ورودی به‌صورت زیر باشد:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

خروجی به‌صورت ماتریس زیر خواهد شد:

```
4  8  12  16
3  7  11  15
2  6  10  14
1  5  9  13
```

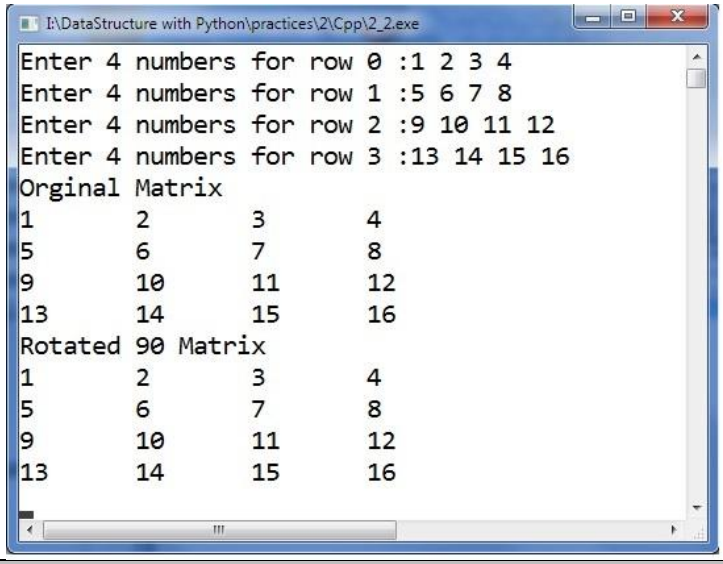
مراحل طراحی و اجرا:

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
#include <iostream.h>
#include <conio.h>
#define R 4
#define C 4
// After transpose we swap elements of column
// one by one for finding left rotation of matrix by 90 degree
void swap(int a, int b)
{
   int temp = a;
   a = b;
   b = temp;
}
void reverseColumns(int arr[R][C])
{
for (int i = 0; i < C; i++)
       for (int j = 0, k = C - 1; j < k; j++, k--)
              swap(arr[j][i], arr[k][i]);
}
// Function for do transpose of matrix
void transpose(int arr[R][C])
{
  for (int i = 0; i < R; i++)
       for (int j = i; j < C; j++)
              swap(arr[i][j], arr[j][i]);
}
// Function for print matrix
void printMatrix(int arr[R][C])
{
    for (int i = 0; i < R; i++) {
       for (int j = 0; j < C; j++) cout << arr[i][j] << "\t";
       cout << '\n';
    }
}
// Function to anticlockwise rotate matrix by 90 degree
void rotate90(int arr[R][C])
{
   transpose(arr);
   reverseColumns(arr);
}
void readMatrix(int a[][C], int row, int col)
{
    for (int i = 0; i < row; i++) {
       cout <<"Enter " << col <<" numbers for row " << i <<  " :";
       for (int j = 0; j < col; j++) cin >> a[i][j];
```

```
    }
}
int main()
{
    int arr[R][C];
    readMatrix(arr, R, C);
    cout << "Orginal Matrix\n";
    printMatrix(arr);
    rotate90(arr);
    cout << "Rotated 90 Matrix\n";
    printMatrix(arr);
    getch();
    return 0;
}
```

۲. پروژه را ذخیره و اجرا کرده تا نمونه خروجی را به‌صورت زیر ببینید:



مثال ۳ – ۲. برنامه‌ای که یک ماتریس را دریافت کرده، ۱۸۰ درجه چرخش می‌دهد. به‌عنوان‌مثال، اگر ماتریس ورودی به‌صورت زیر باشد:

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

ماتریس چرخش یافته به‌صورت زیر خواهد شد:

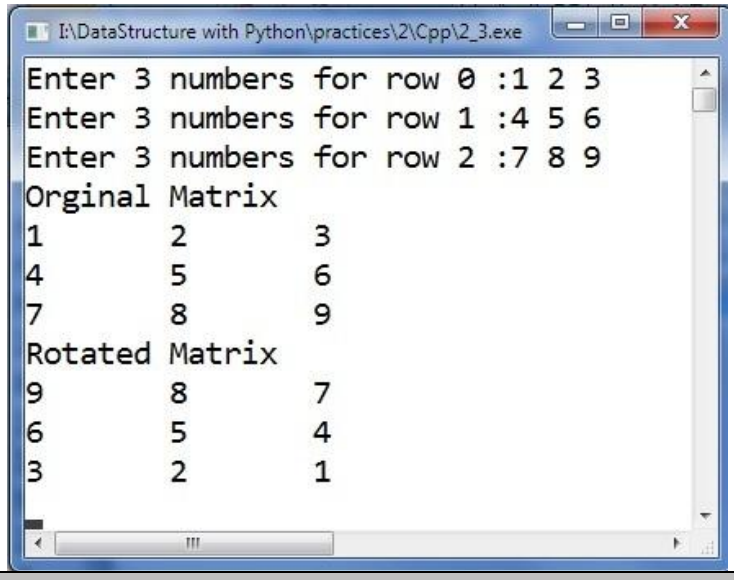
```
6  5  4  3
2  1  0  9
8  7  6  5
4  3  2  1
```

مراحل طراحی و اجرا:

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
#include <iostream.h>
#include <conio.h>
#define N 3
void rotateMatrix(int mat[][N])
{
       // Simply print from last cell to first cell.
       for (int i = N - 1; i >= 0; i--) {
           for (int j = N - 1; j >= 0; j--)
              cout <<  mat[i][j] << "\t";
           cout << "\n";
       }
}
void printMatrix(int arr[N][N])
{
       for (int i = 0; i < N; i++) {
          for (int j = 0; j < N; j++) cout << arr[i][j] << "\t";
          cout << '\n';
       }
}
void readMatrix(int a[][N], int row, int col)
{
    for (int i = 0; i < row; i++) {
      cout << "Enter " << col <<" numbers for row " << i <<  " :";
      for (int j = 0; j < col; j++) cin >> a[i][j];
   }
}
int main()
{
   int mat[N][N];
   readMatrix(mat, N, N);
   cout << "Orginal Matrix\n";
   printMatrix(mat);
   cout << "Rotated Matrix\n";
   rotateMatrix(mat);
   getch();
   return 0;
}
```

۲. پروژه را ذخیره و اجرا کرده تا نمونه خروجی را به‌صورت زیر ببینید:



مثال ۴ – ۲. برنامه‌ای که یک آرایه را دریافت کرده، هر عنصر آن را با حاصل‌ضرب عنصر قبلی و بعدی جایگزین می‌کند. اگر عنصر جایگزینی، اولین عنصر باشد، آن را با حاصل‌ضرب عنصر اول و دوم جایگزین می‌کند. اما، اگر عنصر آخر آرایه باشد، آن را با حاصل‌ضرب عنصر آخر و یکی مانده به عنصر آخر جایگزین می‌نماید. به‌عنوان‌مثال، اگر آرایه به‌صورت زیر باشد:

۲ ۳ ٤ ٥ ٦

آرایه حاصل به شکل زیر خواهد شد:

٦ ٨ ١٥ ٢٤ ٣٠

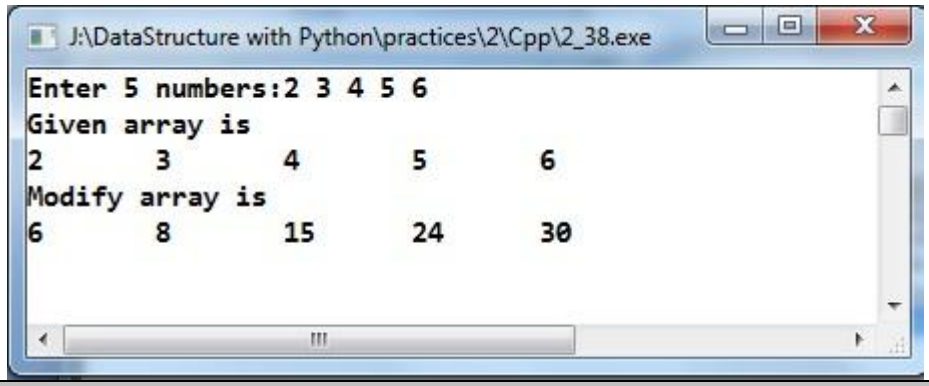
مراحل طراحی و اجرا:

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
#include <iostream.h>
#include <conio.h>
void modify(int arr[], int n)
{
	// Nothing to do when array size is 1
	if (n <= 1) return;
	// store current value of arr[0] and update it
	int prev = arr[0];
	arr[0] = arr[0] * arr[1];
	// Update rest of the array elements
	for (int i=1; i<n-1; i++)
	{
		// Store current value of next interation
		int curr = arr[i];
		// Update current value using previos value
		arr[i] = prev * arr[i+1];
		// Update previous value
		prev = curr;
	}
```

```
        // Update last array element
        arr[n-1] = prev * arr[n-1];
}
void readArray(int arr[], int n)
{
  cout << "Enter " << n << " numbers:";
  for(int i = 0; i < n; i++)
     cin >> arr[i];
}
void printArray(int arr[], int n)
{
   for(int i = 0; i < n; i++)
     cout << arr[i] << "\t";
   cout << endl;
}
int main()
{
   int arr[] = {2, 3, 4, 5, 6};
   int n = sizeof(arr)/sizeof(arr[0]);
   readArray(arr, n);
   cout << "Given array is \n";
   printArray(arr, n);
   modify(arr, n);
   cout << "Modify array is \n";
   printArray(arr, n);
   getch();
   return 0;
}
```

۲. پروژه را ذخیره و اجرا کرده تا نمونه خروجی را به‌صورت زیر ببینید:



مثال ۵ – ۲. برنامه‌ای که یک آرایه را دریافت کرده، بیش‌ترین حاصل‌ضرب هر آرایه فرعی را پیدا می‌کند و نمـایش می‌دهد. به‌عنوان‌مثال، اگر آرایه ورودی به‌صورت زیر باشد:

۶ -۳ -۱۰ ۰ ۲

خروجی ۱۸۰ خواهد شد که آرایه فرعی برابر است با:

۶ -۳ -۱۰

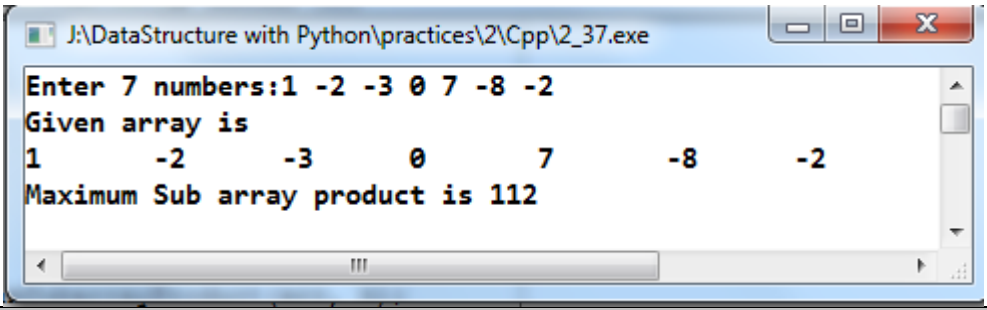
مراحل طراحی و اجرا:

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
#include <iostream.h>
#include <conio.h>
// Utility functions to get minimum of two integers
int min (int x, int y) {return x < y? x : y; }
// Utility functions to get maximum of two integers
int max (int x, int y) {return x > y? x : y; }
/* Returns the product of max product subarray.
Assumes that the given array always has a subarray
with product more than 1 */
int maxSubarrayProduct(int arr[], int n)
{
       // max positive product ending at the current position
       int max_ending_here = 1;
       // min negative product ending at the current position
       int min_ending_here = 1;
       // Initialize overall max product
       int max_so_far = 1;
       /* Traverse through the array. Following values are
       maintained after the i'th iteration:
       max_ending_here is always 1 or some positive product
                                    ending with arr[i]
       min_ending_here is always 1 or some negative product
                                    ending with arr[i] */
       for (int i = 0; i < n; i++)
       {
       /* If this element is positive, update max_ending_here.
              Update min_ending_here only if min_ending_here is
              negative */
              if (arr[i] > 0)
              {
                 max_ending_here = max_ending_here*arr[i];
                 min_ending_here=min(min_ending_here * arr[i], 1);
              }
              /* If this element is 0, then the maximum product
              cannot end here, make both max_ending_here and
              min_ending_here 0
              Assumption: Output is alway greater than or equal
                                    to 1. */
              else if (arr[i] == 0)
              {
                     max_ending_here = 1;
                     min_ending_here = 1;
              }
              /* If element is negative. This is tricky
              max_ending_here can either be 1 or positive.
              min_ending_here can either be 1 or negative.
              next min_ending_here will always be prev.
              max_ending_here * arr[i] next max_ending_here
              will be 1 if prev min_ending_here is 1, otherwise
              next max_ending_here will be prev min_ending_here *
              arr[i] */
```

```
            else
            {
              int temp = max_ending_here;
              max_ending_here=max(min_ending_here * arr[i], 1);
              min_ending_here = temp * arr[i];
            }
            // update max_so_far, if needed
            if (max_so_far < max_ending_here)
            max_so_far = max_ending_here;
        }
        return max_so_far;
}
void readArray(int arr[], int n)
{
  cout << "Enter " << n << " numbers:";
  for(int i = 0; i < n; i++)
     cin >> arr[i];
}
void printArray(int arr[], int n)
{
   for(int i = 0; i < n; i++)
     cout << arr[i] << "\t";
   cout << endl;
}
int main()
{
   int arr[] = {1, -2, -3, 0, 7, -8, -2};
   int n = sizeof(arr)/sizeof(arr[0]);
   readArray(arr, n);
   cout << "Given array is \n";
   printArray(arr, n);
   cout << "Maximum Sub array product is ";
   cout <<    maxSubarrayProduct(arr, n);
   getch();
   return 0;
}
```

۲. پروژه را ذخیره و اجرا کرده تا نمونه خروجی را به‌صورت زیر ببینید:



مثال ۶ – ۲. برنامه‌ای که ضرب دو ماتریس را به‌صورت بازگشتی انجام داده و نمایش می‌دهد. به‌عنوان‌مثال، اگر ماتریس‌های ورودی به‌صورت زیر باشند:

A = 12 56 B = 2 6
45 78 5 8

خروجی به‌صورت زیر خواهد شد:

۳۰٤ ٥۲۰
٤۸۰ ۸۹٤

مراحل طراحی و اجرا:

۱. پروژه جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تغییر دهید:

```
#include <iostream.h>
#include <conio.h>
const int MAX = 100;
#define R 3
#define COL 3
void multiplyMatrixRec(int row1, int col1, int A[][MAX], int row2,
     int col2, int B[][MAX], int C[][MAX])
{
       // Note that below variables are static
       // i and j are used to know current cell of
       // result matrix C[][]. k is used to know
       // current column number of A[][] and row
       // number of B[][] to be multiplied
       static int i = 0, j = 0, k = 0;
       // If all rows traversed.
       if (i >= row1)
              return;
       // If i < row1
       if (j < col2)
       {
       if (k < col1)
       {
              C[i][j] += A[i][k] * B[k][j];
              k++;
              multiplyMatrixRec(row1,col1,A,row2,col2, B, C);
       }
       k = 0;
       j++;
       multiplyMatrixRec(row1, col1, A, row2, col2, B, C);
       }
       j = 0;
       i++;
       multiplyMatrixRec(row1, col1, A, row2, col2, B, C);
}
// Function to multiply two matrices A[][] and B[][]
void multiplyMatrix(int row1, int col1, int A[][MAX],
       int row2, int col2, int B[][MAX])
{
       if (row2 != col1)
       {
              cout << "Not Possible\n";
              return;
       }
```