**//Deactivation and restoration of scope**

#include <stdio.h>

int main()

{

int x = 5;

printf("before block:\t x=%d\n", x);

{

int x = 10; // deactivation of previous scope for x

printf("from block:\t x=%d\n", x);

}

// restoration of the original scope for x

printf("after block:\t x=%d\n", x);

}

**//demo for static function variables in C**

#include <stdio.h>

int main()

{

int a = func(5); // a,b are local variables of main()

int b = func(6); // we perform 2 invocations of func

printf("a=%d b=%d\n", a, b);

}

int func(int x)

{

static int c = 10; // this is done only at first invocation of func

// int c = 10; // this will be done at every invocation of func

c += x;

return(c);

}

**//Allocating storage in Heap**

#include <stdio.h>

#include <stdlib.h>

int \*getArray(int len)

{

int \*arr = malloc(len\*sizeof(int));

int i;

for (i=0; i<len; i++)

arr[i] = i;

return(arr);

}

int main()

{

int len = 5; // stack allocation

int \*a = getArray(len); // heap allocation

int i = 0;

for ( ; i<5; i++)

printf("%d ", a[i]);

printf("\n");

}

**// demo for using the mutable keyword. If a class member is declared as**

**// mutable, you can modify it even if the class instance is declared as**

**// const.**

#include <iostream>

using namespace std;

class Point

{

public:

int x;

mutable int y;

static const int myNumber = 12345; // static const initialization

Point(int xx, int yy): x(xx), y(yy)

{ }

void print() const // preventing to modify the class members

{

cout << x << ' ' << y << endl;

}

void print2() const // however, mutable members CAN be modified

{

y = 15;

cout << x << ' ' << y << endl;

}

};

int main()

{

const Point q(3,4); // constant object

// q.x = 5; // ERROR!

q.y = 10; // this is OK for a mutable member

q.print(); // prints 3 10

q.print2(); // prints 3 15

}

**//Operator Overloading**

Time operator =(const Time &t) // overloaded = operator

{

cout << "greetings from the = operator" << endl;

minutes = t.minutes;

hours = t.hours;

return \*this;

}