1. POINTER
2. Explain the result of following program

#include <iostream>

using namespace std;

typedef int \*IntPtrType;

void main()

{

IntPtrType ptr\_a, ptr\_b, \*ptr\_c;

ptr\_a = new int;

\*ptr\_a = 3;

ptr\_b = ptr\_a;

cout << \*ptr\_a << " " << \*ptr\_b << "\n";

//Result 3 3 (\*ptr\_a = 3 and ptr\_b is assign by ptr\_a so they have same value)

ptr\_b = new int;

\*ptr\_b = 9;

cout << \*ptr\_a << " " << \*ptr\_b << "\n";

//Result 3 9 (now \*ptr\_b is assign with 9, ptr\_a have currrent value)

\*ptr\_b = \*ptr\_a;

cout << \*ptr\_a << " " << \*ptr\_b << "\n";

//Result 3 3 (assgin (value of) prt\_b = (value of) ptr\_a = 3)

delete ptr\_a;

ptr\_a = ptr\_b;

cout << \*ptr\_a << " " << \*ptr\_b << "\n";

//Result 3 3 (value of ptr\_b = 3 and ptr\_a = ptr\_b now)

ptr\_c = &ptr\_a;

cout << \*ptr\_c << " " << \*\*ptr\_c << "\n";

//Result is address of memory block that ptr\_a point to and value of it.

//00B60E50 3

delete ptr\_a;

ptr\_a = NULL;

system("pause");

}

1. Detect and solve problems of following program:

#include <iostream>

using namespace std;

void main()

{

int a[4] = { 1, 2, 3, 4 };

int \*p = a;

int \*p2 = new int;

delete p;

delete a;

delete p2;

}

Problems: delete operator can't be use to delete pointer to a local stack allocated variable so “delete p; delete a; ” will throw errors. Only dynamically allocated variables can be destroyed by using delete operator.

Solve: don’t use “delete p; delete a;”

1. Using pointer:
   * Why should we use delete ?

Because when we use new to create a single object, It will register a memory area with the address stored in the pointer. If we not delete it, operating system is noted that memory area still using and not provider for other needs, it will make waste memory.

* + When we use delete ?

As soon as we create a dynamic object (like calling the new operator), we need to call delete somewhere. (like when we don’t use that object anymore).

* + Difference between delete and delete[]

Delete: use to deallocated a non-array object that allocate and create with new operator.

Delete[]: use to deallocated an array that allocate and create with new operator

Examples:

int \*p = new int;

int \*parr[2] = new int[3][2];

delete p;

delete [] parr;

1. Code:

int iVar = 10; //0x100

int \*p1 = &iVar; //0x200

int \*\*p2 = &p1; //0x300

Result:

1. \*p1 == 10;
2. \*p2 == 0x100;
3. \*(\*p2) == 10;