Main.cpp

```
// This program uses subscript notation with a pointer variable
and
// pointer notation with an array name.
#include <iostream>
#include "pointers.h"
using namespace std;

int main()
{
   const int NUM_COINS = 5;
   int coins[NUM_COINS] = {1, 2, 3, 4, 5};
   int *intPtr; // Pointer to a double
   int count;   // Array index

   Pointers p;

   // Display the contents of the coins array. Use subscripts
   // with the pointer!
   p.printValues(coins, intPtr, NUM_COINS);

   // Display the contents of the array again, but this time
   // use pointer notation with the array name!
   p.printAgain(coins, intPtr, NUM_COINS);

   cout << sizeof(coins) / sizeof(coins[0]) << endl;
   cout << endl;

   return 0;
}
```

Pointers.h

```
class Pointers
```

```
{
private:
public:
    void printValues(int coins[], int *intPtr, int NUM_COINS);
    void printAgain(int coins[], int *intPtr, int NUM_COINS);
};
```

Pointers.cpp

```
#include <iostream>
#include "pointers.h"

using namespace std;

void Pointers::printValues(int coins[], int *intPtr, int
NUM_COINS)
{
    // Assign the address of the coins array to intPtr.
    intPtr = coins;
    cout << "Here are the values in the coins array:\n";
    for (int i = 0; i < NUM_COINS; i++)
    {
        cout << intPtr[i] << " ";
    }
    cout << endl;
    for (int i = 0; i < NUM_COINS; i++)
    {
        cout << coins[i] << " ";
    }
}

void Pointers::printAgain(int coins[], int *intPtr, int
NUM_COINS)
{
```

```
    intPtr = coins;
    cout << "\nAnd here they are again:\n";
    for (int i = 0; i < NUM_COINS; i++)
    {
        cout << "Contents at coins[" << i << "]: ";
        cout << "At memory address " << (intPtr + i) << ": ";
        cout << "content is: " << *(intPtr + i) << " ";
        cout << endl;
        cout << "At memory address " << &(coins[i]) << ": ";
        cout << "At memory address " << (intPtr + i) << " ";
        cout << "Content is: " << *(intPtr + i) << " ";
        cout << endl;
    }
    cout << "The number of elements stored in contigous memory
is: ";
}
```

Output:

```
1 2 3 4 5
1 2 3 4 5
And here they are again:
Contents at coins[0]: At memory address 0x61fef4: content is: 1
At memory address 0x61fef4: At memory address 0x61fef4 Content is: 1
Contents at coins[1]: At memory address 0x61fef8: content is: 2
At memory address 0x61fef8: At memory address 0x61fef8 Content is: 2
Contents at coins[2]: At memory address 0x61fefc: content is: 3
At memory address 0x61fefc: At memory address 0x61fefc Content is: 3
Contents at coins[3]: At memory address 0x61ff00: content is: 4
At memory address 0x61ff00: At memory address 0x61ff00 Content is: 4
Contents at coins[4]: At memory address 0x61ff04: content is: 5
At memory address 0x61ff04: At memory address 0x61ff04 Content is: 5
The number of elements stored in contigous memory is: 5
```