Arrays of Structures

Sometimes we can use Array of structures as memory database for storing data in memory.

Example -5 : Store Disk drive information in an array of structure (mainInventory.cpp)

We must write a data-entry program that receives each inventory item (disk drive) from the keyboard and saves it to an array of structure. We are going to write a menu-driven program with options : Add data, print data onscreen, or exit the program.

```
struct inventory      // Global structure definition.
{
    long  int storage;
    int   access_time;
    char  vendor_code;
    float cost;
    float price;
 }; // No structure variables defined globally.


#include <cstdlib>
#include "main.h"
#include <iostream>
#include <iomanip>

using namespace std;

 int mainInventory()
 {
    inventory disk[125];              // Local array of structures.
    int ans;  int num_items=0;      // Number of total items in the inventory.
    do
    {
      do
      {
         displayMenu();    // Display menu of user choices.
         cin >> ans;           // Get user's request.
      } while ((ans< 1) || (ans> 3));
      switch (ans)
      {
        case (1):  {
           disk[num_items] = enterInventory();    // Enter disk data.
```

```
            num_items++;                              // Increment number of items.
            break;
          }
          case (2):  {
            seeInventory(disk, num_items);        // Display disk data.
            break;
          }
          default :    { break; }
        }
    } while (ans!=3);      // Quit program when user is done.
    return 0;
}

//  Display Menu
void displayMenu(void)
  {
    cout << "\n\n*** Disk Drive Inventory System ***\n\n";
    cout << "Do you want to:\n\n";
    cout << "\t1. Enter new item in inventory\n\n";
    cout << "\t2. See inventory data\n\n";
    cout << "\t3. Exit the program\n\n";
    cout << "What is your choice? ";
    return;
}

//  Enter Inventory Data
inventory  enterInventory()
{
     char vndr;
     inventory disk_item;     // Local variable to fill with input.
     cout << "\n\nWhat is the next drive's storage in bytes? ";  cin >> disk_item.storage;
     cout << "What is the drive's access time in ms? ";              cin >> disk_item.access_time;
     cout << "What is the drive's vendor code (A, B, C, or D)? ";   cin >> vndr;
     disk_item.vendor_code = vndr;
     cout << "What is the drive's cost? ";         cin >> disk_item.cost;
     cout << "What is the drive's price? ";        cin >> disk_item.price;
     return (disk_item);
}
//  See Inventory Data
void seeInventory ( inventory disk[125],  int num_items)
{
     int ctr;
     cout << "\n\nHere is the inventory listing:\n\n";
     for (ctr=0;ctr<num_items;ctr++)
```

```
    {
       cout << "Storage: " << disk[ctr].storage << "\t";
       cout << "Access time: " << disk[ctr].access_time << "\n";
       cout << "Vendor code: " << disk[ctr].vendor_code << "\t";
       cout << fixed << setprecision(2);
       cout << "Cost: $" << disk[ctr].cost << "\t";
       cout << "Price: $" << disk[ctr].price << "\n";
    }
   return;
}
```

Example -6 : Swap content of two stacks using STL (Standard Template Library) and show user entered name in reverse order using stack. (mainSTL.cpp)

```
#include <iostream>
#include <stack>

using namespace std;

int mainSTL ()
{
   //  Swap two Stack defined in STL
    stack<int> stk1;
    stack<int> stk2;

    for (int i = 0; i < 5; ++i)
         stk1.push(i + 1);
    for (int i = 0; i < 3; ++i)
         stk2.push(i + 100);

    swap(stk1, stk2);

    cout << "Contents of stack s1 after swap operation." << endl;
    while (!stk1.empty()) {
     cout << stk1.top() << endl;            stk1.pop();
    }
    cout << endl;

    cout << "Contents of stack s2 after swap operation." << endl;
    while (!stk2.empty()) {
          cout << stk2.top() << endl;       stk2.pop();
    }
    //  Store a name in Stack and Show it in Reverse order
    const int MAX_NAME_LEN = 80;
```

Syntax of defining a stack in **stl** :
stack<object_type> stack_name;

The above statement will create a stack named stack_name of type object_type.

The C++ function std::stack::swap() exchanges the contents of two stacks and modifies size of the stack if necessary.

```
    char name[MAX_NAME_LEN];
    stack<char> stkName;
    cin.ignore();
    cout << "please enter your name..: ";
    //   cin.getline(name, MAX_NAME_LEN) reads a string until it encounters the new line character or
         maximum number of characters (80)
    cin.getline (name, MAX_NAME_LEN);

    //  It reads in a single line of text from standard input and puts it into s as a C-style string, which
        means that it is terminated by the null character. ... The newline character is not included in the
       string.
    for (int i = 0; name[i] != '\0' && !stkName.size() <= MAX_NAME_LEN; i++)
    {
      stkName.push(name[i]);
      //cout << stkName.size() << '\n';
    }
    cout << "your name backwards is..:\n";
    char letName;
    while (!stkName.empty())
    {
       letName = stkName.top();
       stkName.pop();
       cout << letName << '\n';
    }
    return 0;
}
```

Example -7: Exploring the string class and its various method functions (mainString.cpp)

```
#include <iostream>
#include <string>
using namespace std;

//  The standard C++ library provides a string class type that supports all the operations on string type.

int mainString ()
{
   //  Example : Different Type of Constructor
   char *line = "short line for testing";

   // with no arguments constructor
   string s1;     s1 = "Anatoliy";    cout << "s1  is: " << s1 << endl;

   // copy constructor
   string s2 (s1);      cout << "s2  is: " << s2 << endl;
```

```
// one argument in constructor
string s3 (line);     cout << "s3  is: " << s3 << endl;

// first argument of constructor C string second number of characters
string s4 (line,10);
cout << "s4  is: " << s4 << endl;

// 1 - C++ string,  2 - start position,   3 - number of characters
string s5 (s3,6,4);             // copy word 'line' from s3
cout << "s5  is: " << s5 << endl;

// 1 - number characters, 2 - character itself
string s6 (15,'*');
cout << "s6  is: " << s6 << endl;

// 1 - start iterator,  2 - end iterator
string s7 (s3.begin(),s3.end()-5);
cout << "s7  is: " << s7 << endl;

// you can instantiate string with assignment
string s8 = "Anatoliy";
cout << "s8  is: " << s8 << endl;

//  Example : Some common operation on string
string str1 = "Hello World - ";      string str2 = "Practical Session Started";
string str3;    int  len ;

// copy str1 into str3
str3 = str1;              cout << "str3 : " << str3 << endl;

// concatenates str1 and str2
str3 = str1 + str2;        cout << "str1 + str2 : " << str3 << endl;

// total lenghth of str3 after concatenation
len = str3.size();         cout << "Calling str3.size() :  " << len << endl;
cout << "Calling str3.length() :  " << str3.length() << endl;

/*  The find function will return an integer  representing the FIRST OCCURRENCE of a certain
        string, here the functions argument.  */
cout << "Calling find(\"Session\"):"    << str3.find("Session") << endl;

// The substr() function "substring" will return a string  which starts and ends at a determined
// index(es), the  arguments of the function.  If only 1 argument is given, it will begin at that index
```

```
   // and continue all the way to  the end.

   cout << "Calling substr(0,5): "       << str3.substr(0,5) << endl;
   cout << "Calling substr(6): "         << str3.substr(7) << endl;

   /*  The insert() function will look at a designated index  and insert the string, as the second
parameter,  in that place.  */

   cout << str3.insert(0,"WOULD YOU LIKE ") << endl;

   //  The append() function will add the string argument  to the END of the string.  */

   str3.append("----");      cout << str3 << endl;

   string str4 = "Nobody is perfect";
   string s   = ""; // empty string
   char *ch   = "abcdef";

   // Append string str at the end of s; return s

   // Appends at the end of s a copy of the n characters in str1, starting at position pos; if n is too large,
      characters are copied only until  the end of str is reached; returns s
   s.append(str4,0,6);     cout << "s is : " << s << endl;

   // Appends copies of the characters in the range [inpIt1, inpIt2] to s;
   string::iterator inpIt1 = str4.begin()+6;   //start from ' is'
   string::iterator inpIt2 = str4.end();

   s.append(inpIt1,inpIt2);   cout << "s is : " << s << endl;


   // Returns s[pos]
   for ( int pos = 0; pos < s.length(); ++pos )
          cout << s.at(pos) << " ";
   cout << endl;

   //  Swap

   str4 = "Robert";      string str5 = "Forest";
   cout << "str4 is: " << str4 << endl;
   cout << "str5 is: " << str5 << endl;

   cout << "str4.swap(str5)" << endl;
```

```
str4.swap(str5);
cout << "str4 is: " << str4 << endl;
cout << "str5 is: " << str5 << endl;


string strSTL = "STL is created from Dennis Ritchie";
s1  = "was";
s2  = "developed";
s3  = "Stepanov alexander";
cout << "strSTL is: " << strSTL << endl;

cout << "replace 'is' for 'was'" << endl;
strSTL.replace(4, // start position in str
      2,    // how many characters
      s1);  // source for replasment

cout << "strSTL is: " << strSTL << endl;

cout <<"replace 'created' for 'developed'" << endl;
int n = strSTL.find('c'); // pos of 'created'
int x = strSTL.find("from") -1;

strSTL.replace(strSTL.begin()+n, strSTL.begin()+x, s2);

cout << "str is: " << strSTL << endl;

cout << "replace 'Dennis' for 'alexander'" << endl;
int x1 = strSTL.find('D'); // search Dennis
int x2 = strSTL.find(' ',x1+1); // space after
int y1 = s3.find("alex"); // search 'alex'
string strTemp = "alexander";    int y2 = strTemp.length();

strSTL.replace(x1, // start position in str
      x2-x1,  // how characters to replace
      s3,    // source for replacement
      y1,    // start positio from source
      y2);   // how chracter start from y1

cout << "strSTL is: " << strSTL << endl;

cout << "replace 'from' for 'by'" << endl;
char ary[] = "bytes";
n = strSTL.find("from");
```

```
   // same variant possible with iterators
   // instead of number of position
   strSTL.replace(n, // start position in str
         4,    // how many characters
         ary,  // source
         2);   // first 2 characters from source

   cout << "strSTL is: " << strSTL << endl;

   cout << "replace 'a' for 'A' (alexander)" << endl;
   n = strSTL.find("alexander");

   strSTL.replace(n,  // start position in str
         1,     // how character(s)
         1,     // how many copies of character
         'A');  // character for replasment
   cout << "strSTL is: " << strSTL << endl;

   cout << "replace 'Ritchie' for 'Stepanov'" << endl;
   x1 = strSTL.find('R');
   y1 = s3.find(' ');

   strSTL.replace(strSTL.begin()+x1, // start pointer
         strSTL.end(), // to the end of str
         s3.begin(), // start pointer from source
         s3.begin()+y1 // end pointer from
         );          // source
   cout << "strSTL is: " << strSTL << endl;

   return 0;
}
```