UEE 1303(1069): Object-Oriented Programming

Lab #3: Pointer and Reference

In this laboratory session you will:

- Review the usage of pointer and reference

Lab 3-1: Pointer

✓ Program `lab3-1` below shows some examples of using for pointer manipulation including pointer declarations and assignments.

```
// lab3-1-1.cpp
#include <iostream>
using namespace std;
int main()
{
    double a = 1.34;
    double *pa = &a;
    cout << "a = " << a << endl;
    cout << "&a = " << &a << endl;
    cout << "*a = " << *a << endl;
    cout << "pa = " << pa << endl;
    cout << "&pa = " << &pa << endl;
    cout << "*pa = " << *pa << endl;
    *pa = 6.5;
    cout << "a = " << a << endl;
    cout << "*pa = " << *pa << endl;
    return 0;
}
```

– Please try to explain the execution results by yourself. Notice that there is a compiler error in this example.

✓ The following is an example to use pointer arithmetic to dereference the array elements.

```
// lab3-1-2.cpp
#include <iostream>
#include <cstdlib>
```

```
using std::cout;
using std::endl;

int main()
{
    int a[10];
    srand(time(NULL));

    for (int i = 0;i < 10; i++)
        a[i] = rand()%20 + 10;

    int *pa = a;
    for (int i = 0; i < 10; i++)
        cout << *(pa++) << " ";
    cout << endl;

    return 0;
}
```

✓ The program demonstrates that a pointer is used to point the structure object.

```
// lab3-1-3.cpp
#include <iostream>
typedef struct
{
    int x;
    int y;
    double value;
}Point2D;

void assignPoint2D(Point2D *obj, int x, int y, double
value)
{
    obj->x = x;
    obj->y = y;
    obj->value = value;
}
```

```
void displayPoint2D(Point2D *obj)
{
    std::cout << "(" << obj->x << "," << obj->y << ") = "
              << obj->value << std::endl;
}
int main()
{
    Point2D ptArray[10];
    for (int i = 0; i < 10; i++)
    {
        assignPoint2D(ptArray[i], i, i+2, i*10);
        displayPoint2D(ptArray[i]);
    }

    return 0;
}
```

- Please fix the compiler error.

Lab 3-2: Reference

✓ A reference is an implicit pointer that is automatically dereferenced.

```
// lab3-2-1.cpp
#include <iostream>
using namespace std;
int main()
{
    int a = 1024;
    int &refa = a;
    cout << "a = " << a << endl;
    cout << "&a = " << &a << endl;
    cout << "*a = " << *a << endl;
    cout << "refa = " << refa << endl;
    cout << "&refa = " << &refa << endl;
    cout << "*refa = " << *refa << endl;

    return 0;
}
```

- Please try to explain the execution results by yourself. Notice that there is a compiler error in this example.
- Note that the addresses of `pa` and `a` are the same in program `lab3-1-1`, but the addresses of `refa` and `a` are different in this example.

✓ The following is an example to use reference arithmetic to reference the array elements.

```
// lab3-2-1.cpp
#include <iostream>
#include <cstdlib>
using std::cout;
using std::endl;

int main()
{
    int a[10];
    srand(time(NULL));
    for (int i = 0;i < 10; i++)
        a[i] = rand()%20 + 10;

    int &refa;
    for (int i = 0; i < 10; i++){
        refa = a[i];
        cout << refa << " ";
    }
    cout << endl;

    return 0;
}
```

- Since a reference must be initialized to an object, there is a compiler error in this example.

✓ The program demonstrates that a reference type is used to reference the structure object.

```
// lab3-2-3.cpp
#include <iostream>
```

```
typedef struct
{
    int x;
    int y;
    double value;
}Point2D;

void assignPoint2D(Point2D &obj, int x, int y, double
value)
{
    obj.x = x;
    obj.y = y;
    obj.value = value;
}

void displayPoint2D(Point2D &obj)
{
    std::cout << "(" << obj.x << "," << obj.y << ") = "
              << obj.value << std::endl;
}
int main()
{
    Point2D ptArray[10];
    for (int i = 0; i < 10; i++)
    {
        assignPoint2D(ptArray[i],i,i+2,i*10);
        displayPoint2D(ptArray[i]);
    }

    return 0;
}
```

- Try to identify the difference between program `lab3-1-3` and `lab3-2-3`.

Exercise 3-1

✓ Write two versions of the string-comparison function `strcmp`. The first version should use ***array subscripting***, and the second should use ***pointers and pointer***

arithmetic.

✓ The sample output of the program is shown as follows.

- If two strings are the same, return 0

```
>./ex3-1
Enter first string: good
Enter second string: good
The value returned from stringCompare1( "good", "good" ) is 0
The value returned from stringCompare2( "good", "good" ) is 0
```

- If two strings are different and the lengths of two strings are the same, return 1.

```
>./ex3-1
Enter first string: goodbye
Enter second string: goodish
The value returned from stringCompare1( "goodbye", "goodish" ) is 1
The value returned from stringCompare2( "goodbye", "goodish" ) is 1
```

- If two strings are different and the lengths of two strings are different, return -1.

```
>./ex3-1
Enter first string: goodbye
Enter second string: goodness
The value returned from stringCompare1( "goodbye", "goodness" ) is
-1
The value returned from stringCompare2( "goodbye", "goodness" ) is
-1
```

✓ The main structure of the program is like as

```
// ex3-1.cpp
#include <iostream>
using namespace std;

/* Write a function prototype for stringCompare1 */
/* Write a function prototype for stringCompare2 */

int main()
{
    char string1[100], string2[100];
```

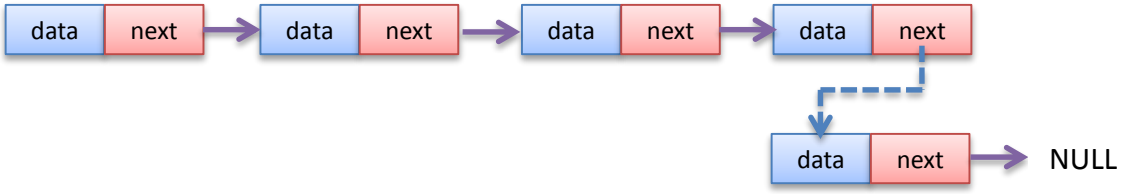
```
    cout << "Enter first string: ";
    cin >> string1;
    cout << "Enter second string: ";
    cin >> string2;

    cout << "The value returned from stringCompare1( \""
         << string1
         << "\", \"" << string2 << "\" ) is "
         << /* Write a function call for stringCompare1 */
         << "\nThe value returned from stringCompare2( \""
         << string1
         << "\", \"" << string2 << "\" ) is "
         << /* Write a function call for stringCompare2 */
         << '\n';

    return 0;
}
```

Exercise 3-2 (LINKED LIST)

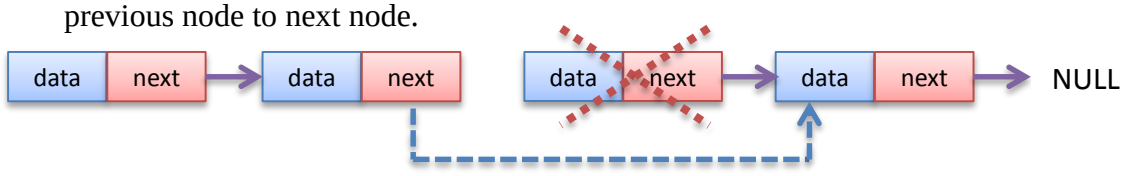
✓ Linked list is a kind of data structure consisting of a sequence of nodes as shown below:

✓ A node contains variables to hold the data information and has a pointer to link to next node. In this exercise, you need to write a simple version of linked list with three functions: *insert*, *delete* and *display*.

- ***Insert***: Insert the data to the end of linked list.



- ***Delete***: Delete the data from linked list. Note that you need to relink previous node to next node.



✓ The sample output of the program is shown as follows.

```
>./ex3-2
Please select an option:
1.Insert a node
2.Delete a node
3.Display the list
4.End
1
Please enter the number:
1
Please select an option:
1.Insert a node
2.Delete a node
3.Display the list
4.End
1
Please enter the number:
2
Please select an option:
1.Insert a node
2.Delete a node
3.Display the list
4.End
1
Please enter the number:
3
Please select an option:
1.Insert a node
2.Delete a node
3.Display the list
4.End
3
1->2->3->
Please select an option:
1.Insert a node
2.Delete a node
3.Display the list
```

```
4.End
2
Please enter the number:
2
Please select an option:
1.Insert a node
2.Delete a node
3.Display the list
4.End3
1->3->
Please select an option:
1.Insert a node
2.Delete a node
3.Display the list
4.End
4
```

✓ The main structure of the program is like as

```
// ex3-2.cpp
#include <iostream>
using namespace std;
struct Node
{
    int data;
    Node *next;
};
//global variable root is used to record the head of link
list
Node* root = NULL;
int main()
{
    size_t i = 0;
    while (1)
    {
        cout << "Please select an option:" << endl
             << "1.Insert a node" << endl
             << "2.Delete a node" << endl
```

```
            << "3.Display the list" << endl
            << "4.End" << endl;
        cin >> i;
        int data;
        switch(i)
        {
        case 1:
            cout << "Please enter the number:" << endl;
            cin >> data;
            InsertNode(data);
            break;
        case 2:
            cout << "Please enter the number:" << endl;
            cin >> data;
            if ( !DeleteNode(data) )
            cout << "Failed to delete node " << data << endl;
            break;
        case 3:
            Display();
            break;
        case 4:
            return 0;
        default:
            break;
        }
    }
}
```