UEE1303(1070) S12: Object-Oriented Programming

Pointer and Reference

What you will learn from Lab 3

In this laboratory, you will review the usage of pointer and reference, which are important in object-oriented programming.

TASK 3-1: POINTER TYPE

✓ 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>
#include <ctime>
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 n1, int n2, double v)
{
    obj->x = n1;
    obj->y = n2;
    obj->value = v;
}

void displayPoint2D(Point2D *obj)
{
    std::cout << "(" << obj->x << "," << obj->y << ") = ";
    std::cout << 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]);
    }
```

&ptArray[i]

```
    return 0;
}
```

- Please fix the compiler error.

TASK 3-2 : REFERENCE TYPE

✓ 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.

```
// lab3-2-2.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 &refa;  // compile error,  declared as a reference but not initialized
    for (int i = 0; i < 10; i++)                    int b, &refa=b;
    {
        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.

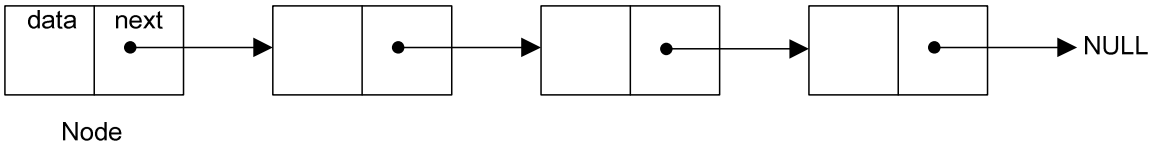
```
// lab3-2-3.cpp
#include <iostream>

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`.

TASK 3-4 EXERCISE

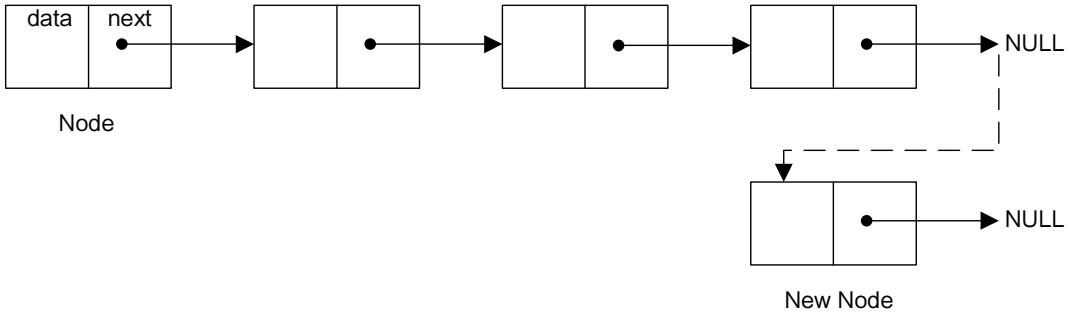
1. *LINKED LIST

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

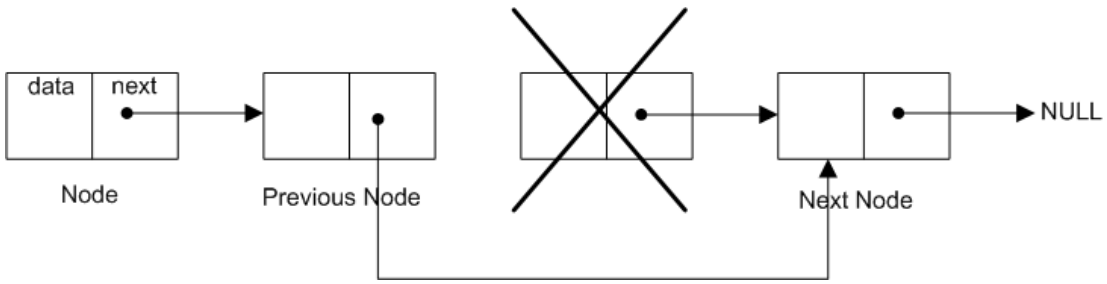


A node contains variables to hold the data information and has a pointer to link to next node. In this exercise, you need to write a simple version of linked list with three functions: ***insert***, ***delete*** and ***display***.

- **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.

```
>./ex3-1
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.End
```

```
3
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,

```
// ex3-1.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;
        }
    }
}
```

2. CARD FLIPPING

✓ Write a card flipping program. There are 36 cards and each card indicates one upper-case alphabet. The user has to gauss and memory the location of card until all cards are flipping. Notice that only two cards have the same alphabet and no daggling card is allowed here.

```
>./ex3-2
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
Please enter card index: 0
G * * * * *
* * * * * *
* * * * * *
* * * * * *
```

```
* * * * * *
Please enter card index: 15
G * * * * *
* * * * * *
* * * H * *
* * * * * *
* * * * * *
Try Again!
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
Please enter card index: 19
* * * * * *
* * * * * *
* * * * * *
* H * * * *
* * * * * *
Please enter card index: 15
* * * * * *
* * * * * *
* * * H * *
* H * * * *
* * * * * *
Good Job!
Please enter card index: 10
* * * * * *
* * * R * *
* * * H * *
* H * * * *
* * * * * *
…
…
…
…
…
Please enter card index: 5
G B C E T Z
R Q P R D E
Z X A H P V
T H S G D S
```

```
A B Q V C X
Congratulation!!
```

- You can design your own card flipping game. For example, decide the situation that the user enters the out-of-range number, design “exit” key to early end this game, or provide the hint to the user.