

Nội dung tuần 04

Luyện tập cách xây dựng các lớp đối tượng nâng cao.

Hướng dẫn

Hiểu rõ về Copy Constructor và Operator = (hàm khởi tạo sao chép và toán tử gán)

Xem ví dụ sau:

```
class HS
{
private:
    char *hoTen;

public:
    HS(void);
    HS(const char *ht);
    HS(const HS& hs);
    ~HS(void);
    const HS& operator=(const HS& hs);
};

HS::HS(void)
{
    hoTen = NULL;
}

HS::HS(const char *ht)
{
    int len = strlen(ht);
    hoTen = new char[len + 1];
    strcpy_s(hoTen, len + 1, ht);
}

HS::HS(const HS& hs)
{
    cout << "Copy Constructor..." << endl;
    int len = strlen(hs.hoTen);
    hoTen = new char[len + 1];
    strcpy_s(hoTen, len + 1, hs.hoTen);
}

HS::~~HS(void)
{
    if (hoTen != NULL)
    {
        delete[] hoTen;
    }
}

const HS& HS::operator=(const HS& hs)
```

```
{
    cout << "Operator = ..." << endl;
    if (hoTen != NULL)
    {
        delete[] hoTen;
    }
    int len = strlen(hs.hoTen);
    hoTen = new char[len + 1];
    strcpy_s(hoTen, len + 1, hs.hoTen);
    return *this;
}

void main()
{
    HS hs1("sdfdgfg");
    cout << "hs2(hs1)" << endl;
    HS hs2(hs1);
    cout << endl << "hs3 = hs1" << endl;
    HS hs3 = hs1;
    cout << endl << "hs2 = hs3" << endl;
    hs2 = hs3;
    cout << endl;
}
```

```
hs2(hs1)
Copy Constructor...
hs3 = hs1
Copy Constructor...
hs2 = hs3
Operator = ...
Press any key to continue . . .
```

Chú ý 2 dòng lệnh được highlight đều là phép gán nhưng chỉ 1 lần hàm Operator= được gọi. Như vậy hàm toán tử gán chỉ được gọi khi đối tượng đã được khởi tạo trước đó (hs2 đã được khởi tạo trước). ***Do vậy nguyên tắc khi có sử dụng cấp phát động thì phải thu hồi trong hàm toán tử gán.***

Sử dụng hàm random

Trước khi sử dụng được hàm phát sinh số ngẫu nhiên **rand()** thì trước hết phải thực hiện gieo hạt giống cho nó thông qua hàm **srand(unsigned int)**, và chỉ cần gieo 1 lần đầu cho suốt chương trình là được.

Vấn đề cần có giá trị trong khoảng [a,b] mong muốn thì sử dụng phép đồng dư (%).

```
//cần có dòng lệnh này trước khi gọi hàm rand() lần đầu tiên
srand((unsigned)time(NULL));
//giá trị lấy được sẽ nằm trong khoảng [a, b]
int x = rand() % (b - a + 1) + a;
```

Sửa bài Số nguyên lớn

Phân khai báo

```
#define MAXLEN 100

class SoNguyenLon
{
private:
    int mangSo[MAXLEN];
    int soCS;
    static SoNguyenLon snlMax;

public:
    SoNguyenLon(void);
    SoNguyenLon(const int& cs, const int& scs);
    SoNguyenLon(const unsigned int& n);
    SoNguyenLon(const SoNguyenLon& snl);
    ~SoNguyenLon(void);

    SoNguyenLon operator+(const SoNguyenLon& snl);
    SoNguyenLon operator-(const SoNguyenLon& snl);
    bool operator>(const SoNguyenLon& snl);
    const SoNguyenLon& operator=(const SoNguyenLon& snl);
    friend SoNguyenLon operator+(const unsigned int& n, const SoNguyenLon& snl);
    friend SoNguyenLon operator-(const unsigned int& n, const SoNguyenLon& snl);
    friend ostream& operator<<(ostream& os, const SoNguyenLon& snl);

    static SoNguyenLon SNLMax();
};
```

Phân cài đặt

```
SoNguyenLon::SoNguyenLon(void)
{
    soCS = 1;
    mangSo[soCS - 1] = 0;
    if (*this > snlMax)
    {
        snlMax = *this;
    }
}

SoNguyenLon::SoNguyenLon(const int& cs, const int& scs)
{
    int csR = cs;
    if (csR < 1)
    {
        csR = 1;
    }
    if (csR > 9)
    {
        csR = 9;
    }
    soCS = abs(scs);
    if (soCS < 1)
    {
        soCS = 1;
    }
}
```

```
}
if (soCS > MAXLEN)
{
    soCS = MAXLEN;
}
for (int i=0; i<soCS; ++i)
{
    mangSo[i] = cs;
}
if (*this > snlMax)
{
    snlMax = *this;
}
}

SoNguyenLon::SoNguyenLon(const unsigned int& n)
{
    unsigned int temp = n;
    soCS = 0;
    while (temp > 9)
    {
        mangSo[soCS++] = temp % 10;
        temp /= 10;
    }
    mangSo[soCS++] = temp;
    if (*this > snlMax)
    {
        snlMax = *this;
    }
}

SoNguyenLon::SoNguyenLon(const SoNguyenLon& snl)
{
    soCS = snl.soCS;
    for (int i=0; i<soCS; ++i)
    {
        mangSo[i] = snl.mangSo[i];
    }
}

SoNguyenLon::~SoNguyenLon(void)
{
}

bool SoNguyenLon::operator>(const SoNguyenLon& snl)
{
    if (soCS > snl.soCS)
    {
        return true;
    }
    if (soCS < snl.soCS)
    {
        return false;
    }
    for (int i=soCS-1; i>=0; --i)
    {
        if (mangSo[i] == snl.mangSo[i])

```

```
        {
            continue;
        }
        if (mangSo[i] > snl.mangSo[i])
        {
            return true;
        }
        return false;
    }
    return false;
}

const SoNguyenLon& SoNguyenLon::operator=(const SoNguyenLon& snl)
{
    soCS = snl.soCS;
    for (int i=0; i<soCS; ++i)
    {
        mangSo[i] = snl.mangSo[i];
    }
    return *this;
}

SoNguyenLon SoNguyenLon::operator+(const SoNguyenLon& snl)
{
    SoNguyenLon snlKQ;
    const SoNguyenLon *snlSCSMax = (soCS > snl.soCS) ? this : &snl;
    const SoNguyenLon *snlSCSMin = (soCS < snl.soCS) ? this : &snl;
    int soCSMin = (soCS > snl.soCS) ? snl.soCS : soCS;
    int nho = 0;
    for (int i=0; i<snlSCSMin->soCS; ++i)
    {
        snlKQ.mangSo[i] = mangSo[i] + snl.mangSo[i] + nho;
        nho = snlKQ.mangSo[i] / 10;
        snlKQ.mangSo[i] %= 10;
    }
    for (int i=snlSCSMin->soCS; i<snlSCSMax->soCS; ++i)
    {
        snlKQ.mangSo[i] = snlSCSMax->mangSo[i] + nho;
        nho = snlKQ.mangSo[i] / 10;
        snlKQ.mangSo[i] %= 10;
    }
    snlKQ.soCS = snlSCSMax->soCS;
    if (nho > 0)
    {
        snlKQ.mangSo[snlKQ.soCS++] = 1;
    }
    if (snlKQ > snlMax)
    {
        snlMax = snlKQ;
    }
    return snlKQ;
}

SoNguyenLon SoNguyenLon::operator-(const SoNguyenLon& snl)
{
    SoNguyenLon snlKQ;
    int nho = 0, i;
```

```
if (soCS >= snl.soCS)
{
    for (i=0; i<snl.soCS; ++i)
    {
        snlKQ.mangSo[i] = mangSo[i] - snl.mangSo[i] - nho;
        nho = 0;
        if (snlKQ.mangSo[i] < 0)
        {
            snlKQ.mangSo[i] += 10;
            nho = 1;
        }
    }
    for (; i<soCS; ++i)
    {
        snlKQ.mangSo[i] = mangSo[i] - nho;
        nho = 0;
        if (snlKQ.mangSo[i] < 0)
        {
            snlKQ.mangSo[i] += 10;
            nho = 1;
        }
    }
    snlKQ.soCS = soCS;
    while(snlKQ.mangSo[snlKQ.soCS-1] == 0)
    {
        snlKQ.soCS--;
    }
}
return snlKQ;
}

SoNguyenLon operator+(const unsigned int& n, const SoNguyenLon& snl)
{
    SoNguyenLon snlTemp(n);
    SoNguyenLon snlKQ = snlTemp + snl;
    if (snlKQ > SoNguyenLon::snlMax)
    {
        SoNguyenLon::snlMax = snlKQ;
    }
    return snlKQ;
}

SoNguyenLon operator-(const unsigned int& n, const SoNguyenLon& snl)
{
    SoNguyenLon snlTemp(n);
    return snlTemp - snl;
}

ostream& operator<<(ostream& os, const SoNguyenLon& snl)
{
    for (int i=snl.soCS-1; i>=0; --i)
    {
        os << snl.mangSo[i];
    }
    return os;
}
```

```
SoNguyenLon SoNguyenLon::SNLMax()
{
    return snlMax;
}
```

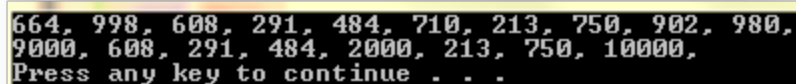
Bài tập

Bài 1

Khai báo và cài đặt lớp đối tượng **LinkedList** sao cho hàm main sau chạy đúng

```
void main()
{
    srand(1234);
    LinkedList l;
    for (int i=0; i<10; ++i)
    {
        if (rand()%2 == 0)
        {
            l.AddHead(rand() % 1001);
        }
        else
        {
            l.AddTail(rand() % 1001);
        }
    }
    cout << l << endl;
    l.RemoveHead();
    l.RemoveTail();
    l[-1] = 9000;
    l[4] = 2000;
    l[100] = 10000;
    cout << l << endl;
}
```

Với kết quả:



```
664, 998, 608, 291, 484, 710, 213, 750, 902, 980,
9000, 608, 291, 484, 2000, 213, 750, 10000,
Press any key to continue . . .
```

Gợi ý phần khai báo:

```
struct Node
{
    int info;
    Node *pNext;
};

class LinkedList
{
private:
    Node *pHead, *pTail;
```

```
int curN;

public:
    LinkedList(void);
    ~LinkedList(void);

    static Node* CreateNode(const int& n);
    Node* AddHead(const int& n);
    Node* AddTail(const int& n);
    Node* RemoveHead();
    Node* RemoveTail();
    friend ostream& operator<<(ostream& os, const LinkedList& ll);
    int& operator[](const int& i);
};
```

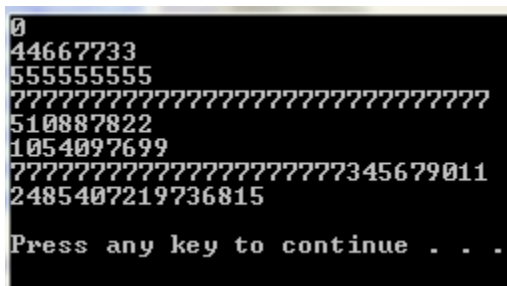
Bài 2

Khai báo và cài đặt lớp đối tượng **SoNguyenLon** với số lượng chữ số không giới hạn (sử dụng **LinkedList**) sao cho hàm main sau chạy đúng

```
void main()
{
    SoNguyenLon snl1;
    SoNguyenLon snl2(44667733);
    SoNguyenLon snl3(5, 9);
    SoNguyenLon snl4(7, 30);
    SoNguyenLon snl5 = snl3 - snl2;
    SoNguyenLon snl6 = 1098765432 - snl2;
    SoNguyenLon snl7 = snl4 - snl3 + 123456789;
    SoNguyenLon snl8 = snl2 * snl3;

    cout << snl1 << endl << snl2 << endl << snl3 << endl;
    cout << snl4 << endl << snl5 << endl << snl6 << endl;
    cout << snl7 << endl << snl8 << endl << endl;
}
```

Với kết quả:



```
0
44667733
555555555
????????????????????????????????????
510887822
1054097699
????????????????????????????????????7345679011
2485407219736815
Press any key to continue . . .
```