

객체지향프로그래밍 LAB #13

<기초문제>

1. 아래의 프로그램을 작성하시오. (/*구현*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

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

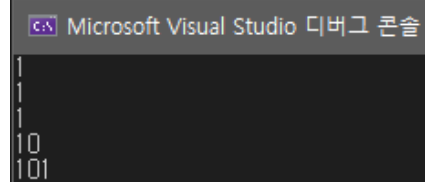
template</* 구현 */>
bool less_than(/* 구현 */) { return a < b; }

template<class T>
T sum(const vector<T>& v) { /* 구현 */ }

int main() {
    cout << less_than(2, 3) << endl;
    cout << less_than(2.1, 2.9) << endl;
    cout << less_than(2, 2.5) << endl;

    vector<int> v1{ 1, 2, 3, 4 };
    vector<double> v2{ 10.1, 20.2, 30.3, 40.4 };
    cout << sum(v1) << endl;
    cout << sum(v2) << endl;

    return 0;
}
```



Microsoft Visual Studio 디버그 콘솔

```
1
1
1
1
10
101
```

2. 아래의 프로그램을 작성하시오. (/*구현*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

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

template<class T>
class Point {
private:
    T x;
    T y;
public:
    Point(T _x, T _y);
    void setXY(T _x, T _y);
    T getX();
    T getY();
    void print();
};
```

```

template<class T>
Point<T>::Point(T _x, T _y) : x(_x), y(_y) {}

template<class T>
void Point<T>::setXY(T _x, T _y) {
    x = _x;
    y = _y;
}

// getX() 구현

// getY() 구현

// print() 구현

int main() {
    Point<int> pt(1, 2);
    Point<double> pt2(1.1, 2.2);
    pt.print();
    pt2.print();
}

```

Microsoft Visual Studio 디버그 콘솔

```

1, 2
1.1, 2.2

```

3. 아래의 프로그램을 작성하시오. (*구현* 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

```

#include <iostream>
#include <vector> // 빠른 search, 느린 pop/push
#include <list>   // 느린 search, 빠른 pop/push
using namespace std;

int main() {
    list<int> myList{ 1, 2, 3, 4 };
    char command;
    int inputVal;
    bool finished = false;
    while (!finished) {
        //command를 입력받음
        cout << "I)input, P)rint, L)ength, E)mpty, Q)uit>>";
        cin >> command;

        //command에 따라 기능 수행
        switch (command) {
            case 'I':
            case 'i':
                cin >> inputVal;
                // push_back 구현
                break;

```

```

        case 'P':
        case 'p':
            // simplified for로 list출력 구현
            break;
        case 'L':
        case 'l':
            cout << "Number of items: " << /* 구현 */ << endl;
            break;
        case 'E':
        case 'e':
            /* 구현 */
            break;
        case 'Q':
        case 'q':
            finished = true;
            cout << "Exit the program" << endl;
            break;
        default:
            cout << "Wrong command" << endl;
            break;
    }

    return 0;
}

```

Microsoft Visual Studio 디버그 콘솔

```

I)input, P)rint, L)ength, E)mpty, Q)uit>>p
1      2      3      4
I)input, P)rint, L)ength, E)mpty, Q)uit>>i
123
I)input, P)rint, L)ength, E)mpty, Q)uit>>p
1      2      3      4      123
I)input, P)rint, L)ength, E)mpty, Q)uit>>l
Number of items: 5
I)input, P)rint, L)ength, E)mpty, Q)uit>>e
I)input, P)rint, L)ength, E)mpty, Q)uit>>p
I)input, P)rint, L)ength, E)mpty, Q)uit>>q
Exit the program

```

4. 아래의 프로그램을 작성하시오. (/*구현*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

```

#include <iostream>
#include <vector>
using namespace std;

int main() {
    int ary[] = { 1, 2, 3, 4 };
    int* pBegin, *pEnd;
    pBegin = ary;
    pEnd = ary + 4;
    for (/* 구현 */)
        cout << *pIter << "Wt";
    cout << endl;
}

```

```

//auto, begin(), end()
vector<int> v{ 10, 20, 30, 40 };
auto iter_begin = begin(v);
auto iter_end = end(v);
for (/* 구현 */)
    cout << *iter << "Wt";
cout << endl;

return 0;
}

```

Microsoft Visual Studio 디버그 콘솔

1	2	3	4
10	20	30	40

5. 아래의 프로그램을 작성하시오. (/*구현*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

```

#include <iostream>
#include <vector> // 빠른 search, 느린 pop/push
#include <list>   // 느린 search, 빠른 pop/push
using namespace std;

template</* 구현 */>
void print(const Iter& iter_begin, const Iter& iter_end) {
    for (/* 구현 */)
        cout << *iter << "Wt";
    cout << endl;
}

template</* 구현 */>
void print_reverse(const Iter& iter_begin, const Iter& iter_end) {
    Iter iter = iter_end;
    /* 구현 */
    cout << endl;
}

int main() {
    vector<int> v{ 1, 2, 3, 4 };
    list<double> l{ 1.1, 2.2, 3.3 };
    int ary[] = { 100, 200, 300, 400 };

    print(begin(v), end(v));
    print(begin(l), end(l));
    print(begin(ary), end(ary));

    print_reverse(begin(v), end(v));
    print_reverse(begin(l), end(l));
    print_reverse(begin(ary), end(ary));

    return 0;
}

```

C# Microsoft Visual Studio 디버그 콘솔			
1	2	3	4
1.1	2.2	3.3	
100	200	300	400
4	3	2	1
3.3	2.2	1.1	
400	300	200	100

<응용문제>

1. 아래 코드를 기반으로 다양한 type(int, double, float)을 사용하여 추가, 삭제, 출력 기능을 하는 List class를 구현하고 이를 사용하는 프로그램을 작성하라. 이 때, list를 오름차순으로 정렬하여라.

조건 1. 오름차순으로 정렬을 하기 위해서는 중복된 데이터가 list 안에 있으면 안된다.
조건 2. list에 데이터를 추가하는 순간 정렬이 되어 있어야한다.
나와있는 조건 말고 다른 경우는 예외처리 하지 않아도 됨
Example)

입력 순서 → 3 - 4 - 5 - 1 - 2 list 안 데이터 순서 → 1 - 2 - 3 - 4 - 5

```
int command()
{
    int num;

    cout << "WnWt---- menu ----" << endl;
    cout << "Wt1. 리스트 추가" << endl;
    cout << "Wt2. 리스트 삭제" << endl;
    cout << "Wt3. 리스트 출력" << endl;
    cout << "Wt4. 프로그램 종료" << endl;
    cout << "WnWt입력 --> ";
    cin >> num;
    return num;
}

int main()
{
    CList<type> list; // type형으로 list 선언
    type input; // list에 입력 할 데이터
    int com; // 선택한 기능
    while (1)
    {
        com = command(); // 기능을 선택
        switch (com)
        {
            case 1: // 추가
                cout << "Wn추가할 데이터 : ";
                cin >> input;
                list.Add(input);
                break;
            case 2: // 삭제
                cout << "Wn삭제 할 데이터 : ";
                cin >> input;
                list.Delete(input);
                break;
            case 3: // 출력
                list.Print();
        }
    }
}
```

```

        break;
    case 4: // 프로그램 종료
        cout << "WnWt 프로그램을 종료합니다Wn";
        return 0;
        break;
    default:
        break;
    }
}
return 0;}

```

[참조 1]

```

template <typename T>
class CList
{
public:
    CList();
    ~CList();

    bool IsEmpty(); // list가 비어 있으면 1, 아니면 0
    bool IsFull(); // list가 꽉 차 있으면 1, 아니면 0

    void Add(T data); // list에 데이터 추가
    void Delete(T data); // list에 데이터 삭제
    void Print(); // list에 데이터 출력

private:
    T m_Array[5]; // 데이터를 저장할 공간
    int m_Length; // list에 있는 데이터 수
};

```

1 - 출력화면 :

<기본 화면>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 -->

```

<추가>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 1
추가할 데이터 : 7

```

<추가 할 때 list가 차 있을 때>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 1
추가할 데이터 : 9
List is full.

```

<중복 된 데이터가 있을 시>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 1
추가할 데이터 : 7
중복된 데이터가 존재합니다.

```

<삭제>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 2
삭제할 데이터 : 7

```

<삭제 할 때 list가 비어 있을 때>

```
----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 2

삭제할 데이터 : 1

List is empty.
```

<출력>

```
----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 3

※ Current List
1 2 3 4 5
```

<종료>

```
----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 4

프로그램을 종료합니다
```

2. 아래의 조건을 만족하는 프로그램을 작성하라.

1. 크기가 10인 vector1과 vector2를 만든다.
2. vector1의 범위는 0~10이고 vector2의 범위는 0~20이며 난수로 채워진다.
3. vector1에 있는 어떠한 수와 vector2의 있는 어떠한 수를 곱 했을 때 가장 큰 경우(곱의 최댓값)과 최솟값을 찾는다.
4. 이 때 vector의 데이터에 접근하기 위해서 iterator만을 사용한다.

2 – 출력화면 :

```
<vetor 1>
8 9 7 5 1 7 5 6 3 9
<vetor 2>
8 19 16 0 20 5 6 8 14 18

최댓값 = 180
최솟값 = 0
```

3. 람다 함수를 활용하여 회문을 판별하는 프로그램을 작성하세요.

- 1) 단어를 뒤집어도 똑같은 단어를 회문이라고 정의합니다.(ex. level)
- 2) 람다 함수를 활용하여 회문을 판별하는 프로그램을 작성하세요.
- 3) 'Q' 혹은 'q' 입력 시 반복을 종료합니다.

3 – 출력 예시

```
문자열 하나를 입력해주세요 : LEVEL
입력하신 문자열의 역순 : LEVEL
이 문자는 회문입니다.

문자열 하나를 입력해주세요 : HELLO
입력하신 문자열의 역순 : OLLEH
이 문자는 회문이 아닙니다.

문자열 하나를 입력해주세요 : COMPUTER
입력하신 문자열의 역순 : RETUPMOC
이 문자는 회문이 아닙니다.

문자열 하나를 입력해주세요 : ABCDCBA
입력하신 문자열의 역순 : ABCDCBA
이 문자는 회문입니다.
```


4. 홀수 숫자 n 을 하나 입력받고, $n \times n$ 크기의 마방진을 출력하는 프로그래밍을 작성하세요.

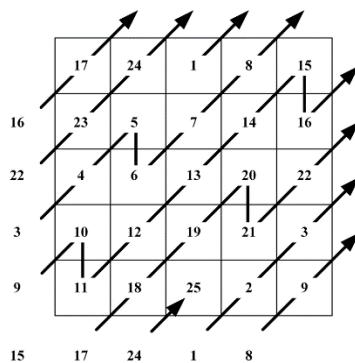
1) 마방진이란, $n \times n$ 행렬에서 가로, 세로, 대각선 방향의 숫자를 더하면 모두 같은 값이 나오는 배열입니다.

- 예시

4	9	2
3	5	7
8	1	6

2) 마방진을 만드는 원리는 1에서부터(보통 1은 첫 번째 줄 가운데에 두고 시작합니다.)

오른쪽 위 대각선 방향으로 숫자를 하나씩 늘려가는 방식을 사용합니다.



4 - 출력예시

```
홀수 숫자를 하나 입력해 주세요 : 3
8   1   6
3   5   7
4   9   2
계속하려면 아무 키나 누르십시오 . . .
```

```
홀수 숫자를 하나 입력해 주세요 : 5
17  24   1   8  15
23   5   7  14  16
 4   6  13  20  22
10  12  19  21   3
11  18  25   2   9
계속하려면 아무 키나 누르십시오 . . .
```

5. 다양한 Type을 사용하는 Queue Class를 구현하세요

```
변수(private) :
    T * p_list; // 정수형 변수들을 가지는 배열
    int size; // 현재 저장된 변수들의 개수
    int MAX_SIZE; // 최대 저장할 수 있는 p_list의 크기

함수(public);
    Queue(int _MAX_SIZE = 1000)
        // 생성자: p_list의 크기를 MAX_SIZE만큼 동적 할당.
    ~Queue()
        // 소멸자: p_list의 동적 할당을 해제
    int find_index(T _item)
        // p_list에서 _item과 동일한 값이 있는지 검색 후 발견시 index를 반환한다 만약
        // 발견하지 못하면 -1을 반환한다
    void Enqueue(T _item)
        // 입력 item을 p_list의 끝에 저장한다. 만약 _item과 동일한 값이 p_list에 존재할 경우
        // p_list에 _item을 추가하지 않는다. (힌트: find_index 함수를 사용해서 중복된 값이
        // p_list에 있는지 조사 후 없는 경우에 입력 item을 p_list에 추가). size가 MAX_SIZE보다 크면
        // item을 추가하지 않는다. ("Error: out of memory" 출력)
    T Dequeue()
        // p_list에 있는 첫번째 item을 제거한 다음 그 아이템을 return한다 (힌트: size 값을
        // 줄이면 p_list의 아이템을 제거한 것과 동일한 효과) size가 0일 때는 item을 제거하지 않는다.
        // ("Error: No item exists in the list" 출력)
    void print() const
        // Queue 객체의 item들을 출력한다
    int get_size()
        // Queue 객체의 크기를 출력한다
    T get_item(int _index)
        // p_list의 해당 index에 있는 item 값을 리턴한다.
```

<시작코드-변경금지>

```
int main()
{
    Queue<int> int_queue;
    Queue<float> float_queue;
```

```

Queue<char> char_queue;

int_queue.Enqueue(1);
int_queue.Enqueue(2);
int_queue.Enqueue(2);
int_queue.Enqueue(5);

float_queue.Enqueue(4.3);
float_queue.Enqueue(2.5);
float_queue.Enqueue(3.7);
float_queue.Enqueue(3.7);

char_queue.Enqueue('b');
char_queue.Enqueue('b');
char_queue.Enqueue('c');
char_queue.Enqueue('a');

cout << "<Before Dequeue>" << endl;
int_queue.print();
float_queue.print();
char_queue.print();

int_queue.Dequeue();
float_queue.Dequeue();
float_queue.Dequeue();
char_queue.Dequeue();
char_queue.Dequeue();
char_queue.Dequeue();
char_queue.Dequeue();

cout << "<After Dequeue>" << endl;
int_queue.print();
float_queue.print();
char_queue.print();

return 0;
}

```

5 - 출력예시

```

<Before Dequeue>
Items in the list : 1, 2, 5,
Items in the list : 4.3, 2.5, 3.7,
Items in the list : b, c, a,
Error: No item exist in the list
<After Dequeue>
Items in the list : 2, 5,
Items in the list : 3.7,
Items in the list :

```

