**s객체지향프로그래밍 LAB #13**

**<기초문제>\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

|  |
| --- |
| #include<iostream>  #include<vector>  using namespace std;  template<typename T, typename V>  bool less\_than(T a, V b) { return a < b; }  template<class T> // template <class T>로 구현하는 이유??  T sum(const vector<T>& v) {  T result = 0;  for (auto elem : v) {  result += elem;  }  return result;  }  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;  } |
|  |

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 -> template <class T> 와 template <typename> 의 차이점. 명확하게 분리가 가능?  template <class T> // T type 의 Point T type으로 추가 설정이 필요. -> T Point<T>::getX()  T Point<T>::getX() {  return this->x;  }  // getY() 구현  template<class T>  T Point<T>::getY() {  return this->y;  }  // print() 구현  template<class T>  void Point<T>::print() {  cout << this->getX() << ", " << this->getY() << endl;  }  int main() {  Point<int> pt(1, 2);  Point<double> pt2(1.1, 2.2);  pt.print();  pt2.print();  } |
|  |

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)nput, P)rint, L)ength, E)mpty, Q)uit>>";  cin >> command;  //command에따라 기능 수행  switch (command) {  case 'I':  case 'i':  cin >> inputVal;  myList.push\_back(inputVal);  break;  case 'P':  case 'p':  // simplified for로 list출력 구현  for (auto elem : myList) {  cout << elem << '\t';  }  cout << endl;  break;  case 'L':  case 'l':  cout << "Number of items: " << myList.size() << endl;  break;  case 'E':  case 'e':  myList.clear();  break;  case 'Q':  case 'q':  finished = true;  cout << "Exit the program" << endl;  break;  default:  cout << "Wrong command" << endl;  break;  }  }  return 0;  } |
|  |

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 (auto pIter = pBegin; pIter < pEnd; pIter++)  cout << \*pIter << "\t";  cout << endl;  //auto, begin(), end()  vector<int> v{ 10, 20, 30, 40 };  auto iter\_begin = begin(v);  auto iter\_end = end(v);  for (auto iter = iter\_begin; iter<iter\_end; iter++)  cout << \*iter << "\t";  cout << endl;  return 0;  } |
|  |

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<class Iter>  void print(const Iter& iter\_begin, const Iter& iter\_end) {  for (auto iter = iter\_begin; iter!=iter\_end; iter++)  cout << \*iter << "\t";  cout << endl;  }  template<class Iter>  void print\_reverse(const Iter& iter\_begin, const Iter& iter\_end) {  Iter iter = iter\_end;  while (iter != iter\_begin) {  iter--;  cout << \*iter << '\t';  }  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;  } |
|  |

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

|  |
| --- |
| #include <iostream>  using namespace std;  int sum(int x, int y) { return x + y; }  int mult(int x, int y) { return x \* y; }  int evaluate(int(\*f)(int, int), int x, int y) {  return f(x, y);  }  int main() {  cout << evaluate(&sum, 2, 3) << endl;  cout << evaluate(&mult, 2, 3) << endl;  // lambda 함수: [](입력변수)->리턴타입 {본문}  // sum(): [](int x, int y)->int { return x + y; }  cout << evaluate(sum, 20, 30) << endl;  // simplified lambda함수 표헌: [](입력변수) {본문}  // mult(): [](int x, int y) { return x \* y; }  cout << evaluate(mult, 20, 30) << endl;  //생성과 호출을 동시에: 람다함수(입력값) // 변수는 x,y 로 받을 거고 int 형으로 설정해주면서 출력을 동시에 해줘~라고 임의로 설정해놓음.  [](int x, int y) {cout << x << ", " << y << endl; }(20, 30);  auto f = [](int x, int y) { return x - y; };  cout << f(1000, 2000) << endl;  return 0;  } |
|  |

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

|  |
| --- |
| #include <iostream>  #include <functional> // function object  using namespace std;  //[closure]: 외부 변수를 lambda함수 내부로 전달  //[a]: 변수 a를 call by value로 lambda함수에 전달  //[&a]: 변수 a를 call by reference로 전달  //[=]: 모든 외부 변수를 call by value로 전달  //[&]: 모든 외부 변수를 call by ref.로 전달  // - 사용시 주의할 점: closure를 사용할 경우 function객체로 assign 받을 것  int evaluate2(function<int(int, int)> f, int x, int y) {  return f(x, y);  }  int main() {  int a = 10, b = 20;  //[a]: 변수 a를 call by value로 lambda함수에 전달  // lambda function 정의할 때, [여기에 설정을 할 수 있음. 설정에 따른 변화는 위에 제시되어 있음.]  cout << evaluate2([a](int x, int y) -> int {return 10 + x + y; }, 2, 3) << endl;  //[&]: 모든 외부 변수를 call by ref.로 전달  cout << evaluate2([&](int x, int y)->int {return x \* y; }, a, b) << endl;  /\* a = 20; a \* x 람다 함수 구현 \*/  a = evaluate2([](int x, int y)-> int {return x \* y; }, a, 2);  cout << "a: " << a << endl;  return 0;  } |
|  |

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

|  |
| --- |
| #include <iostream>  #include <functional>  #include <algorithm> // for\_each, copy, transform  #include <vector>  using namespace std;  int main() {  vector<int> v1 = { 1,2,3,4 };  for (int& elem : v1) {  cout << elem << '\t';  }  cout << endl;  // for\_each(시작위치(iter), 끝위치(iter), 람다함수)  for\_each(begin(v1), end(v1), [](int x) {cout << x << '\t'; }); // v1의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)  cout << endl;  for\_each(begin(v1), end(v1), [](int& x) {x++; }); // v1의 시작부터 끝까지 모든 elem++  for\_each(begin(v1), end(v1), [](int x) {cout << x << '\t'; }); // v1의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)  cout << endl;  int a = 10;  for\_each(begin(v1), end(v1), [a](int& x) {x += a; }); // v1의 시작부터 끝까지 모든 elem+=a  for\_each(begin(v1), end(v1), [](int x) {cout << x << '\t'; }); // v1의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)  cout << endl;  vector<int> v2(v1.size());  // copy: container1 (source)의 element를 container 2(destinstion)로 복사  // copy(src시작위치, src끝위치, dst시작위치)  // v1: {1, 2, 3, 4}  // v2: {0, 0, 2, 3}  copy(begin(v1)+1, end(v1)-1, begin(v2)+2); // v1의 (시작+1) ~ (끝-1)을 v2의 (시작+2)위치부터 하나씩 복사  for\_each(begin(v2), end(v2), [](int& x) {cout << x << '\t'; }); // v2의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)  cout << endl;  // transform: cont1의 element를 변형한다음(람다함수) cont2에 복사  // transform(src시작위치, src끝위치, dst시작위치, 람다함수)  transform(begin(v1), end(v1), begin(v2), [](int x) {return x \* x; }); // v1의 시작부터 끝까지 제곱해서 v2에 복사  for\_each(begin(v2), end(v2), [](int x) {cout << x << '\t'; }); // v2의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)  cout << endl;  return 0;  } |
|  |

**<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 à 3 – 4 – 5 – 1 – 2 list 안 데이터 순서 à 1 – 2 – 3 – 4 - 5 |

이 문제에서 주의해야할 점은 모든 데이터를 m\_Length로만 다뤄야 한다는 점이었다. 물론 다른 방법도 있겠지만, 그렇게 되면 변수를 알 수 없는 m\_Array를 초기화 해줘야 한다는 에러가 계속 발생했기 때문. 그리고 추가적으로 여러 조건을 걸어놓고 bool 변수로 상태를 알 수 있도록 해주는 것보다 특정 case가 되면 동작을 멈추도록 하는 return을 쓰는 것이 훨씬 편함을 기억하자.

|  |
| --- |
| #include <iostream>  using namespace std;  template<typename T>  class CList {  public:  CList() {  m\_Length = 0;  }  ~CList() {}  bool IsEmpty() { // list가 비어 있으면 1, 아니면 0  if (m\_Length == 0) return 1;  else return 0;  }  bool IsFull() { // list가 꽉 차 있으면 1, 아니면 0  if (m\_Length == 5) return 1;  else return 0;  }  void Add(T data) { // list에 데이터 추가  if (IsFull() == true) cout << '\t' << '\t' << "List is full" << endl;  else if (IsFull() == false) {  for (int i = 0; i < m\_Length - 1; i++) {  if (data == m\_Array[i]) {  cout << "중복된 데이터가 존재합니다." << endl;  return;  }  }  m\_Array[m\_Length] = data;  m\_Length++;  }  for (int i = 0; i < m\_Length; i++) { // Sort the List when add the list  for (int j = i; j < m\_Length; j++) {  if (m\_Array[i] > m\_Array[j]) {  T temp = m\_Array[i];  m\_Array[i] = m\_Array[j];  m\_Array[j] = temp;  }  }  }  }  void Delete(T data) { // list에 데이터 삭제  int index = -1;  if (IsEmpty() == true) {  cout << "List is empty" << endl;  }  else {  for (int i = 0; i < m\_Length; i++) {  if (data == m\_Array[i]) {  m\_Length = m\_Length - 1;  for (int j = i; j < m\_Length; j++) {  m\_Array[j] = m\_Array[j + 1];  }  break;  }  }  }  }  void Print() { // list에 데이터 출력  cout << "※ Current List" << endl;  for (int i = 0; i < m\_Length; i++) {  cout << m\_Array[i] << " ";  }  cout << endl;  }  private:  T m\_Array[5]; // 데이터를 저장할 공간  int m\_Length; // list에 있는 데이터 수  };  int command()  {  int num;  cout << "\n\t---- menu ----" << endl;  cout << "\t1. 리스트 추가" << endl;  cout << "\t2. 리스트 삭제" << endl;  cout << "\t3. 리스트 출력" << endl;  cout << "\t4. 프로그램 종료" << endl;  cout << "\n\t입력 --> ";  cin >> num;  return num;  }  int main()  {  CList<int> list; // type형으로 list 선언  int input; // list에 입력 할 데이터  int com; // 선택한 기능  while (1)  {  com = command(); // 기능을 선택  switch (com)  {  case 1: // 추가  cout << "\n추가할 데이터 : ";  cin >> input;  list.Add(input);  break;  case 2: // 삭제  cout << "\n삭제할 데이터 : ";  cin >> input;  list.Delete(input);  break;  case 3: // 출력  list.Print();  break;  case 4: // 프로그램 종료  cout << "\n\t프로그램을 종료합니다\n";  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:

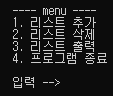
T m\_Array[5]; // 데이터를 저장할 공간

int m\_Length; // list에 있는 데이터 수

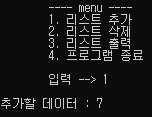
};

1 – 출력화면 :

<기본 화면>



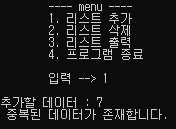
<추가>



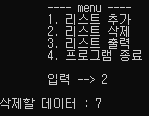
<추가 할 때 list가 차 있을 때>



<중복 된 데이터가 있을 시>



<삭제>



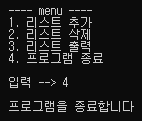
<삭제 할 때 list가 비어 있을 때>



<출력>



<종료>



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. |

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> vector1(10);

vector<int> vector2(10);

for (int i = 0; i < vector1.size(); i++) {

int num1 = rand() % 11;

vector1.at(i) = num1;

}

for (int j = 0; j < vector2.size(); j++) {

int num2 = rand() % 21;

vector2.at(j) = num2;

}

// for문 -> Iterator 사용.

vector<int>::iterator iter\_v1, iter\_v2;

for (iter\_v1 = vector1.begin(); iter\_v1 != vector1.end(); iter\_v1++) {

cout << \*iter\_v1 << " ";

}

cout << endl;

for (iter\_v2 = vector2.begin(); iter\_v2 != vector2.end(); iter\_v2++) {

cout << \*iter\_v2 << ' ';

}

cout << endl;

int max\_val(0), min\_val(200);

for (iter\_v1 = vector1.begin(); iter\_v1 != vector1.end(); iter\_v1++) {

for (iter\_v2 = vector2.begin(); iter\_v2 != vector2.end(); iter\_v2++) {

max\_val = max(max\_val, (\*iter\_v1) \* (\*iter\_v2));

min\_val = min(min\_val, (\*iter\_v1) \* (\*iter\_v2));

}

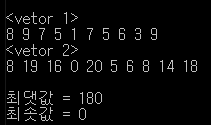
}

cout << "최댓값 = " << max\_val << endl;

cout << "최솟값 = " << min\_val << endl;

}

2 – 출력화면 :



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. |

#include <iostream>

#include <string>

using namespace std;

string reverse\_string(string word) {

auto word\_begin = word.begin();

auto word\_end = word.end();

string result = "";

for (auto iter = word\_end - 1; iter != word\_begin; iter--) {

result += \*iter;

}

result += word[0];

return result;

}

int main() {

string word = "";

while (1) {

cout << "문자열 하나를 입력해주세요 : ";

cin >> word;

if (word == "q" || word=="Q") {

cout << "종료합니다. " << endl;

return 0;

}

cout << "입력하신 문자열의 역순 : ";

string rev\_str = reverse\_string(word);

cout << rev\_str << endl;

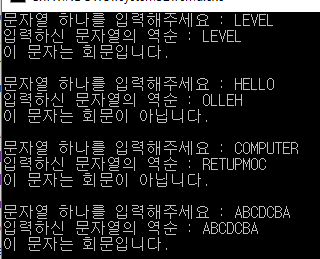
//if (word == rev\_str) cout << "이 문자는 회문입니다." << '\n' << endl;

//else cout << "이 문자는 회문이 아닙니다. " << '\n' << endl;

[](string word1, string word2) {if (word1 == word2) cout << "이 문장은 회문입니다." << '\n' << endl; else cout << "이 문자는 회문이 아닙니다. " << '\n' << endl; }(word, rev\_str);

}

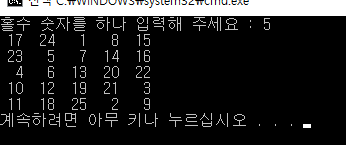
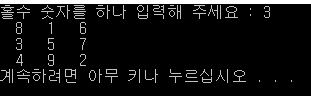
}

3 – 출력 예시

4. 홀수 숫자 n을 하나 입력받고, n\*n 크기의 마방진을 출력하는 프로그래밍을 작성하세요.

|  |
| --- |
| 1) 마방진이란, n\*n 행렬에서 가로, 세로, 대각선 방향의 숫자를 더하면 모두 같은 값이 나오는 배열입니다.  - 예시  EMB00002c180357  2) 마방진을 만드는 원리는 1에서부터(보통 1은 첫 번째 줄 가운데에 두고 시작합니다.) 오른쪽 위 대각선 방향으로 숫자를 하나씩 늘려가는 방식을 사용합니다.  EMB00002c180358 |

4 - 출력예시



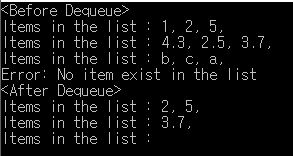
**5. 다양한 Type을 사용하는 Queue Class를 구현하세요**

|  |
| --- |
| 변수(private) :  T \* p\_list; // 정수형 변수들을 가지는 배열  int size; //현재 저장된 변수들의 개수  int MAX\_SIZE; // 최대로 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)  // 입력item을 p\_list의 끝에 저장한다. 만약 \_item과 동일한 값이 p\_list에 존재할 경우 p\_list에 \_입력 item을 추가하지 않는다. (힌트: find\_index 함수를 사용해서 중복된 값이 p\_list에 있는지 조사후 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 제거한다음 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



6. 아래의 코드를 기반으로 정수를 입력 받았을 때, 그 정수만큼 list를 출력하는 프로그램을 작성하시오. 이때, 만약 list의 크기보다 입력 받은 정수가 크다면 try/catch문을 이용하여 예외처리한다. 또한, 0 이하의 숫자를 입력 받으면 프로그램을 종료하도록 한다.

|  |
| --- |
| int main() {  vector<int> list{ 10, 20, 30, 40, 50 };  int num; // 출력할 list의 수  while (1) { /\* 구현 \*/ }  cout << "Program exit..." << endl;  return 0;  } |

6-출력화면:

