**Test1**

#include <iostream>

#define arrMaxSize 1024

using namespace std;

/\*

\* 마지막 항 안나옴;

\*/

int Fibo\_repeat(int num) {

int\* fn = new int[arrMaxSize];

fn[0] = 1;

fn[1] = 1;

for (int i = 2; i < num; i++) {

fn[i] = fn[i-1] + fn[i-2];

}

return fn[num-1];

}

int main() {

int n;

cout << " 몇 번째 수열까지 출력할까요? : ";

cin >> n;

for (int i = 1; i < n+1; i++) {

cout << Fibo\_repeat(i);

cout << " ";

if (i % 5 == 0) cout << '\n';

}

}

**Test2**

#include <iostream>

#define arrMaxsize 10

using namespace std;

int\* binarySearch(int\* pArr, int\* pFirst, int\* pLast, int key);

void OUTPUT(int\* pArr, int num);

int main() {

int arr[arrMaxsize] = { 5,9,13,17,21,28,37,46,55,88 };

cout << "원시 데이터 : ";

for (int i = 0; i < arrMaxsize; i++) {

cout.width(3);

cout << arr[i];

}

int x;

while (true) {

cout << '\n';

cout << "검색 데이터 입력(검색종료 : 0) ";

cin >> x;

if (x == 0) break;

int\* result =binarySearch(arr, arr, arr + arrMaxsize - 1, x);

if (result == NULL) {

cout << "없다고!!!" << endl;

continue;

}

else {

cout << "검색 데이터 위치 : ";

cout << result - arr + 1 << "번째 위치 " << \*result;

}

}

return 0;

}

/\*

1. 중간값 찾기

2. pmid == key 면 return

3. key < pmid면 왼쪽에서 다시 찾기

4. key > pmid면 오른쪽에서 다시 찾기

\*/

int\* binarySearch(int\* pArr, int\* pFirst, int\* pLast, int key) {

while (pFirst-pArr <= pLast-pArr) {

int\* pmid = pFirst + (pLast - pFirst) / 2;

if (\*pmid == key) return pmid;

else if (key < \*pmid) {

pLast = pmid - 1;

continue;

}

else if (key > \*pmid) {

pFirst = pmid + 1;

continue;

}

}

return NULL;

}

**Test3**

#include <iostream>

using namespace std;

class SNode {

int data;

SNode\* link;

friend class SLinkedList;

};

class SLinkedList {

SNode\* head;

SNode\* tail;

int count;

public:

SLinkedList();

~SLinkedList();

bool isEmpty() const;

int countSNode() const;

SNode\* frontSNode() const; //첫 노드 탐색

SNode\* makeSNode(const int& e);

void addRear(const int& e);

void removeFront();

void printSLinkedList();

};

SLinkedList::SLinkedList() {

head = NULL;

tail = NULL;

}

SLinkedList::~SLinkedList() {

while (!isEmpty()) removeFront();

}

bool SLinkedList::isEmpty() const {

return head == NULL;

}

int SLinkedList::countSNode() const{

SNode\* temp = head;

int count = 0;

while (temp != NULL) {

temp = temp->link;

count++;

}//while의 결과 : temp = NULL;

return count;

}

SNode\* SLinkedList::frontSNode() const {

if (isEmpty()) {

cout << " 빈 리스트입니다." << endl;

return NULL;

}

return head;

}

SNode\* SLinkedList::makeSNode(const int& e) {

SNode\* nNode = new SNode; //SNode를 저장할 메모리 공간 부여

nNode->data = e;

nNode->link = NULL;

return nNode;

}

void SLinkedList::addRear(const int& e) {

SNode\* NewNode = makeSNode(e);

if (isEmpty()) {

head = NewNode;

return;

}

//탐색

SNode\* temp = head;

while (temp->link != NULL) {

temp = temp->link;

}//while 결과 : 마지막 노드

temp->link = NewNode;

}

void SLinkedList::removeFront() {

SNode\* old = head;

head = old->link;

delete old;

}

void SLinkedList::printSLinkedList() {

SNode\* temp = head;

while (temp!= NULL) {

cout << temp->data << " ->> ";

temp = temp->link;

}//temp = NULL

cout << "NULL" << endl;

}

int main() {

int x;

SLinkedList s;

while (true) {

cout << "임의의 정수 입력(종료 : 0): ";

cin >> x;

if (x == 0) break;

else {

s.addRear(x);

}

}

cout << "### 입력된 데이터(총 : " << s.countSNode() << " ) ###" << endl;

s.printSLinkedList();

return 0;

}