자료구조 실습02

Data Structures Lab02

Lab 02: Sorted List 설계 및 구현

◎ 내용:

- ☞ Lab 01에서 구현한 Unsorted List를 수정하여 Sorted List를 구현
- ☞ Sorted List의 응용
- ☞ Run Time 오류 수정을 위한 Debugger 사용법

◎ 방법

- ☞ Lab01에 구현한 Unsorted List의 멤버 함수를 Unsorted List용으로 수정
- ☞ 순차적인 자료 검색이 요구되는 멤버 함수를 Iteration을 이용하여 구현
- ☞ 검색을 위한 멤버 함수를 Binary Search를 이용하여 구현

◎ 제출물

- ☞ “Lab 02 과제”에 정의된 응용 프로그램 소스코드
- ☞ 실행 결과

Lab02 : 예제

◎ 내용:

☞ UnSorted List를 다음 멤버함수를 수정하여 Sorted List로 변환

- Add(ItemType data)
- Delete(ItemType data)
- Retrieve_SeqS(ItemType& data): sorted list의 장점을 살릴 수 있도록 수정

☞ Application class에 다음을 추가

- SearchByID_SequenS(); **// primary key인 id로 검색**

◎ **방법:**

☞ 예제 **solution**은 주어진다**.** 하지만 **Lab01**과 마찬가지로 스스로 프로그램을 작성하고 그 결과를 확인하는 용도로 사용해야 한다**.**

예제: SortedList ADT

```
class SortedList
{
public:
        SortedListType();                       // default constructor
        ~SortedListType();                      // default destructor

        void MakeEmpty();                       // Make list empty(Initialize list)
        int GetLength();                        // Return the number of records in the list
        bool IsFull();                          // Check the list upper limit
        bool IsEmpty();                         // Check the list is empty
        void ResetList();                       // Initialize the iterator pointer
        int GetNextItem(ItemType& data);        // Update pointer to point to next record and return this new record.
        int Add(ItemType data);                 // Add a new data to list
        int Delete(ItemType data);              // Delete data record from list
        int Replace(ItemType data);             // Find to same record using primary key and replace it
        int Retrieve(ItemType& target);         // Find the item whose primary key matches with the primary key of target
                                                // and return the copy of the item in target.
        int RetrieveByBS(ItemType& data);  // Retrieve by using binary search
private:
        ItemType m_Array[MAXSIZE];              // array for record
        int m_Length;                           // number of record
        int m_CurPointer;                       // current pointer
};
```

Lab02: Add (1/3)

◎ Add new item: Empty array

MAXSIZE: 10 Length: 0

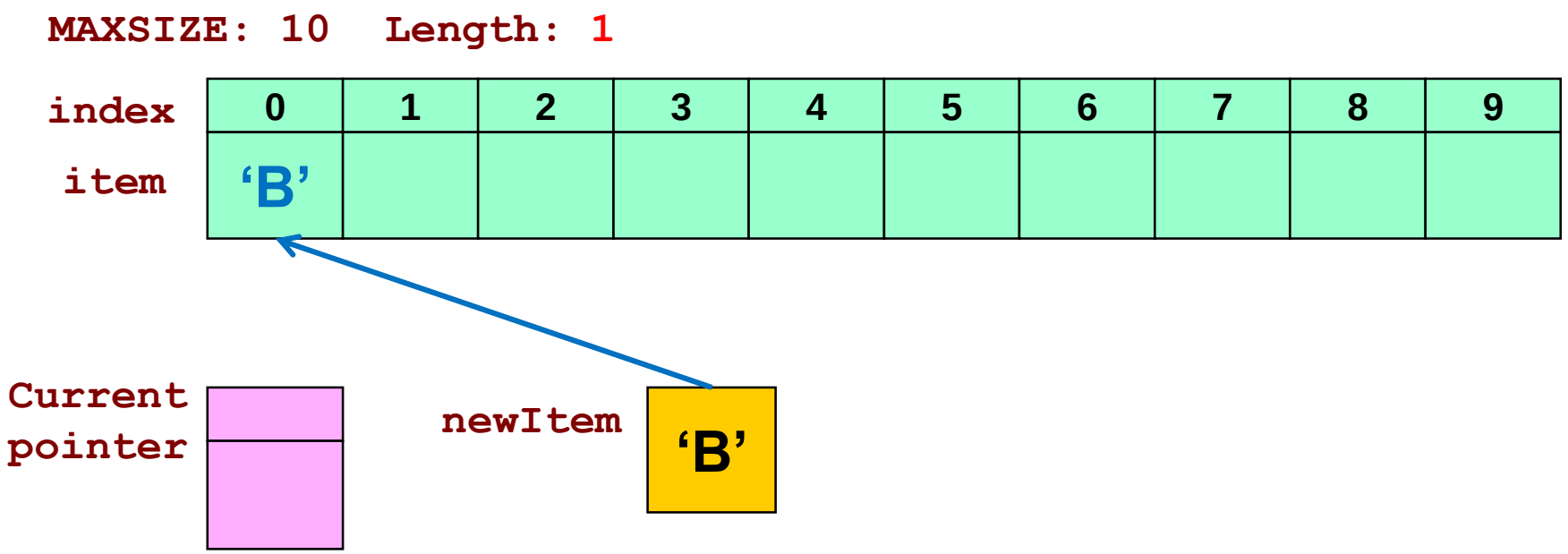
index	0	1	2	3	4	5	6	7	8	9
item										

Current pointer

newItem **'B'**

Lab02: Add (1/3)

◎ Add new item: Empty array



Lab02: Add (2/3)

◎ Add new item: Add to last

MAXSIZE: 10 Length: 1

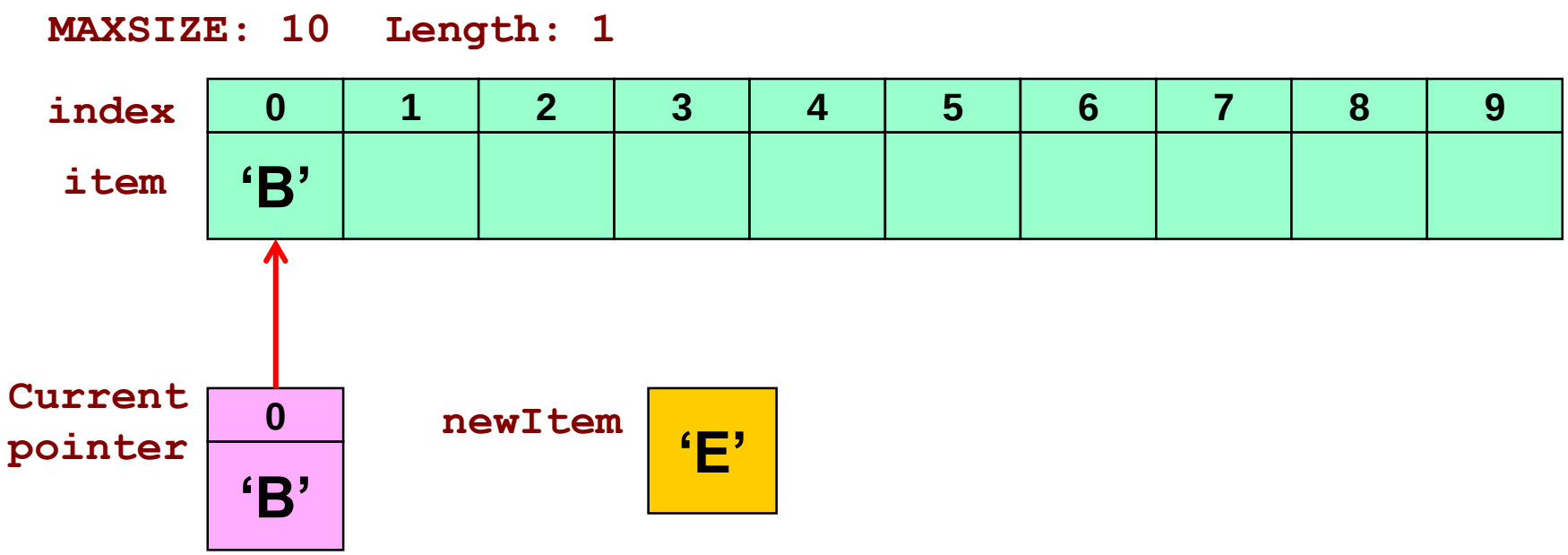
index	0	1	2	3	4	5	6	7	8	9
item	'B'									

Current pointer

newItem 'E'

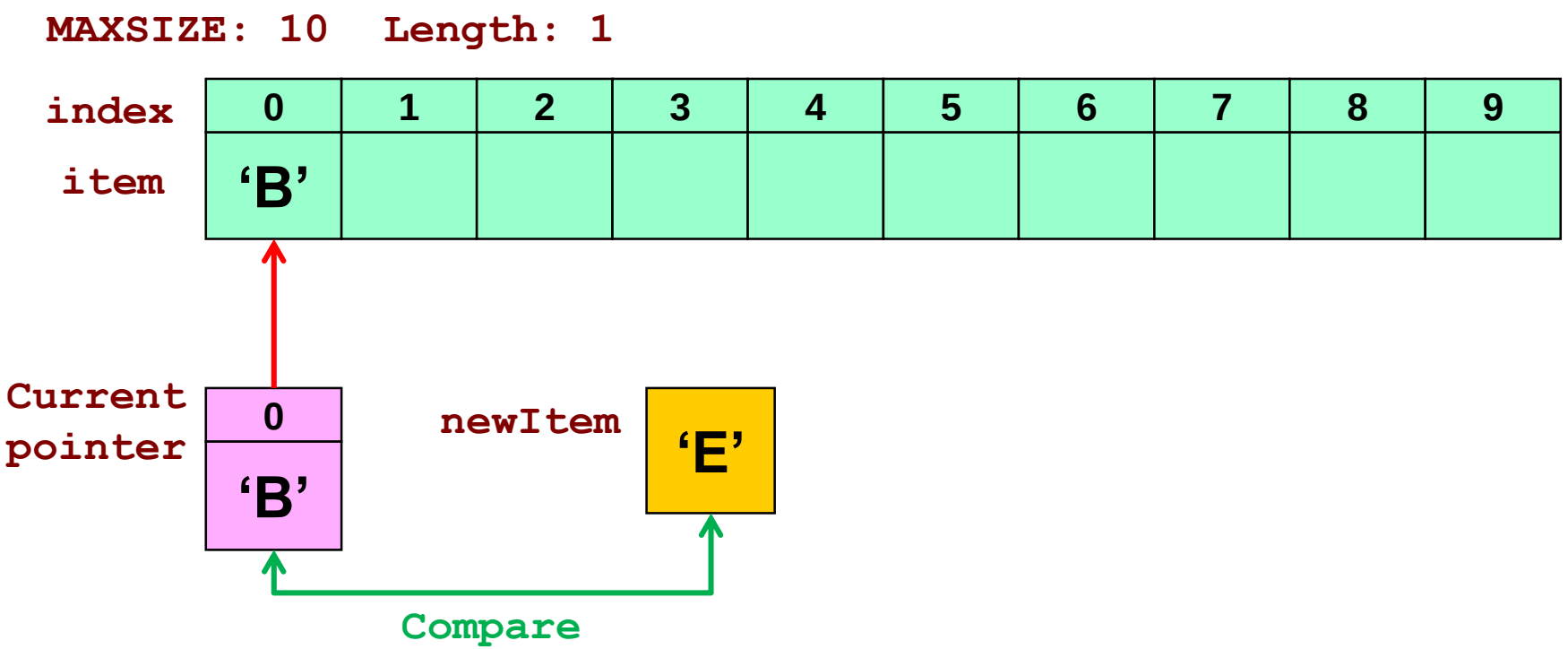
Lab02: Add (2/3)

◎ Add new item: Add to last



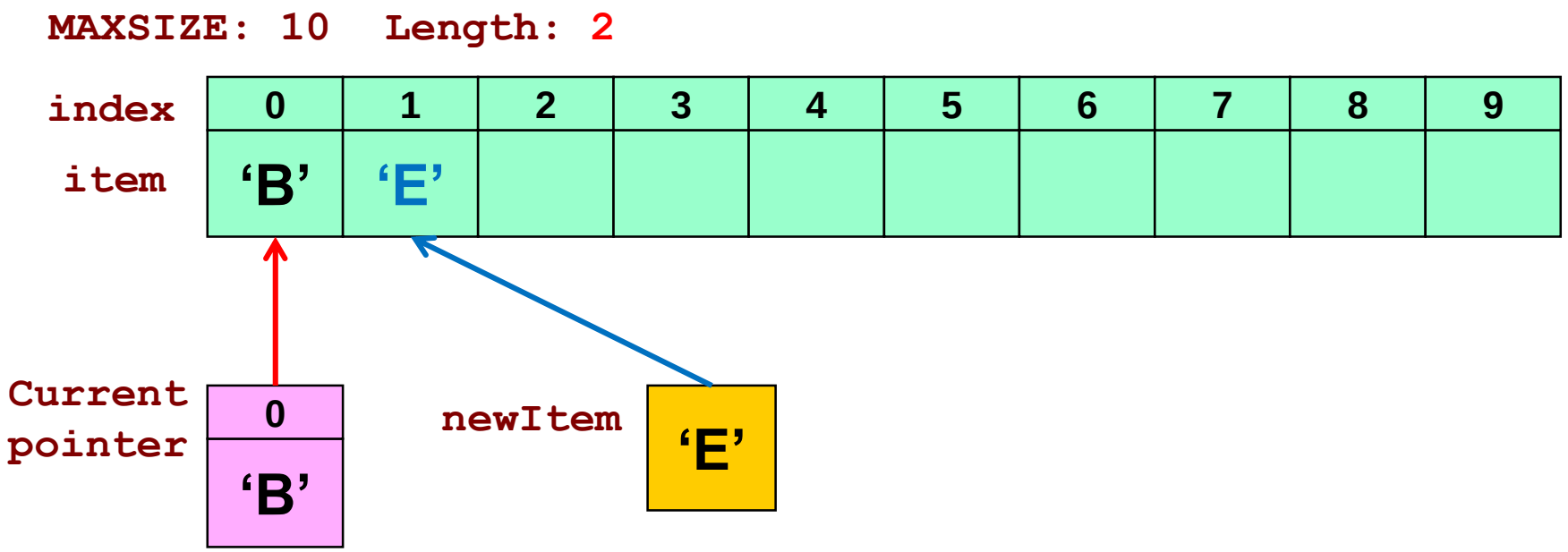
Lab02: Add (2/3)

◎ Add new item: Add to last



Lab02: Add (2/3)

◎ Add new item: Add to last



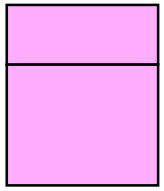
Lab02: Add (2/3)

◎ Add new item: Add to last

MAXSIZE: 10 Length: 2

index	0	1	2	3	4	5	6	7	8	9
item	'B'	'E'								

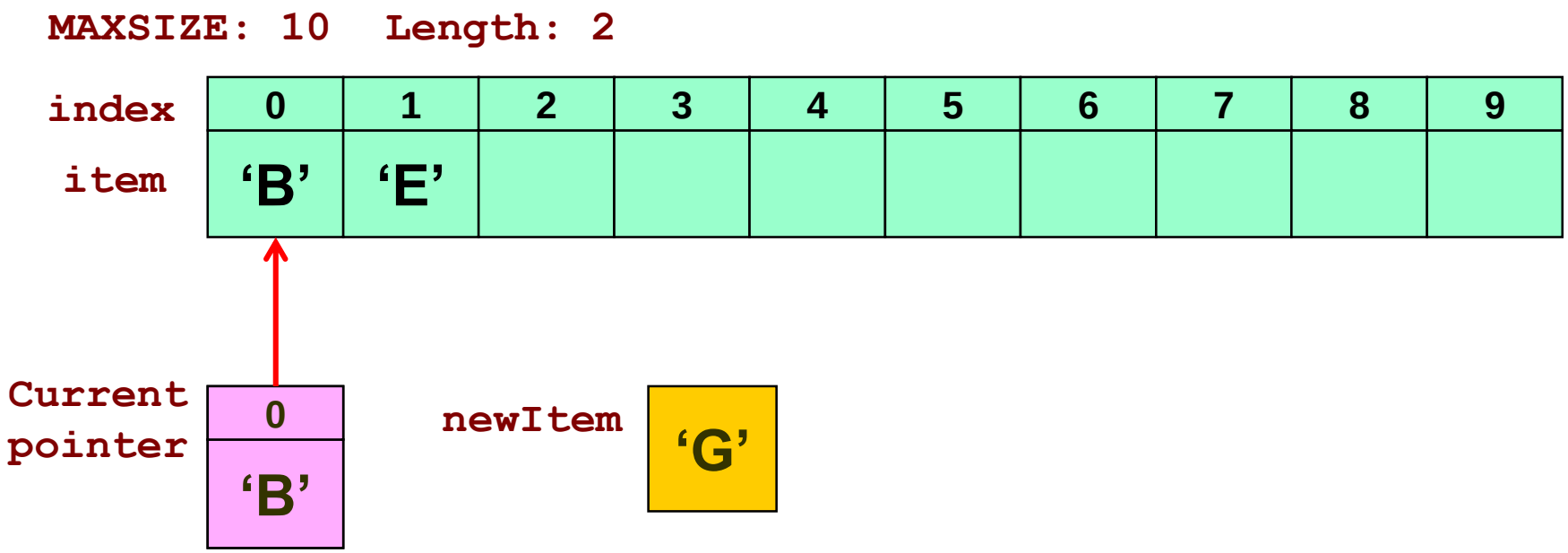
Current pointer



newItem

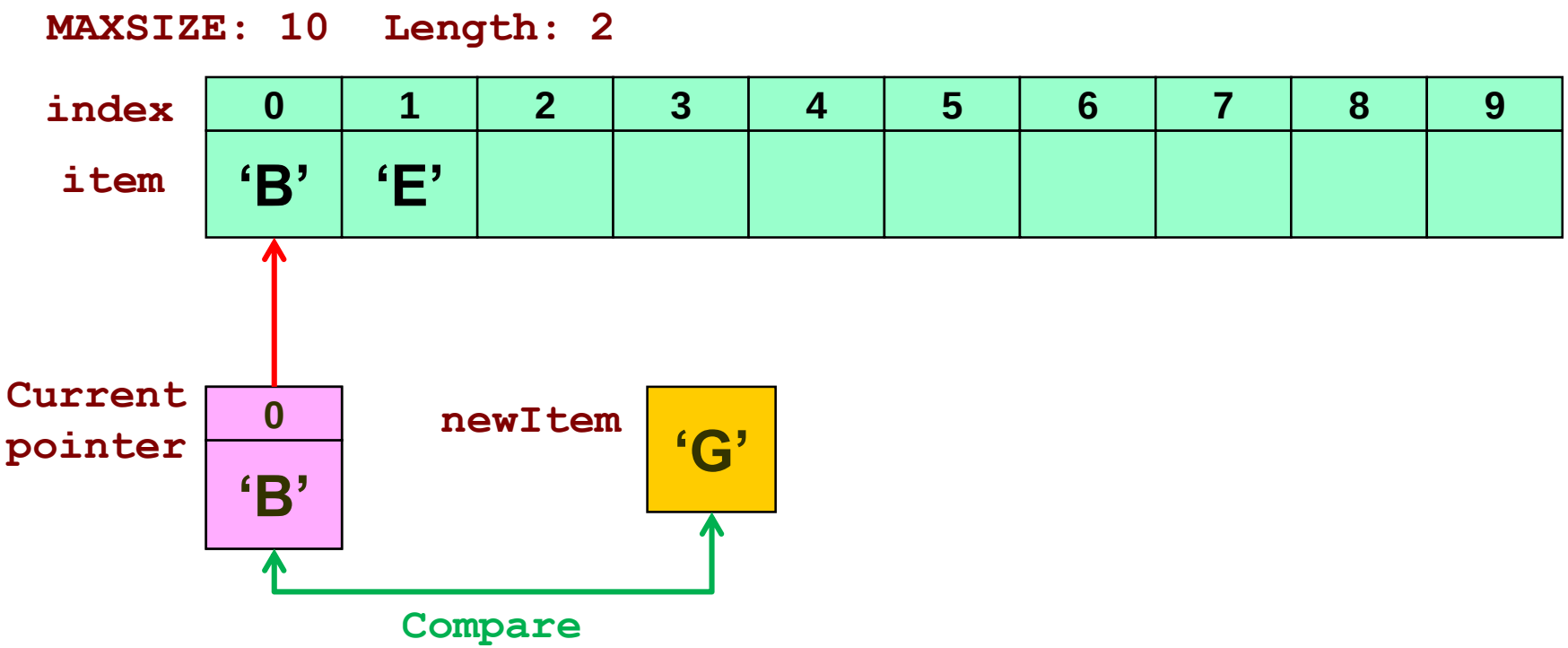
Lab02: Add (2/3)

◎ Add new item: Add to last



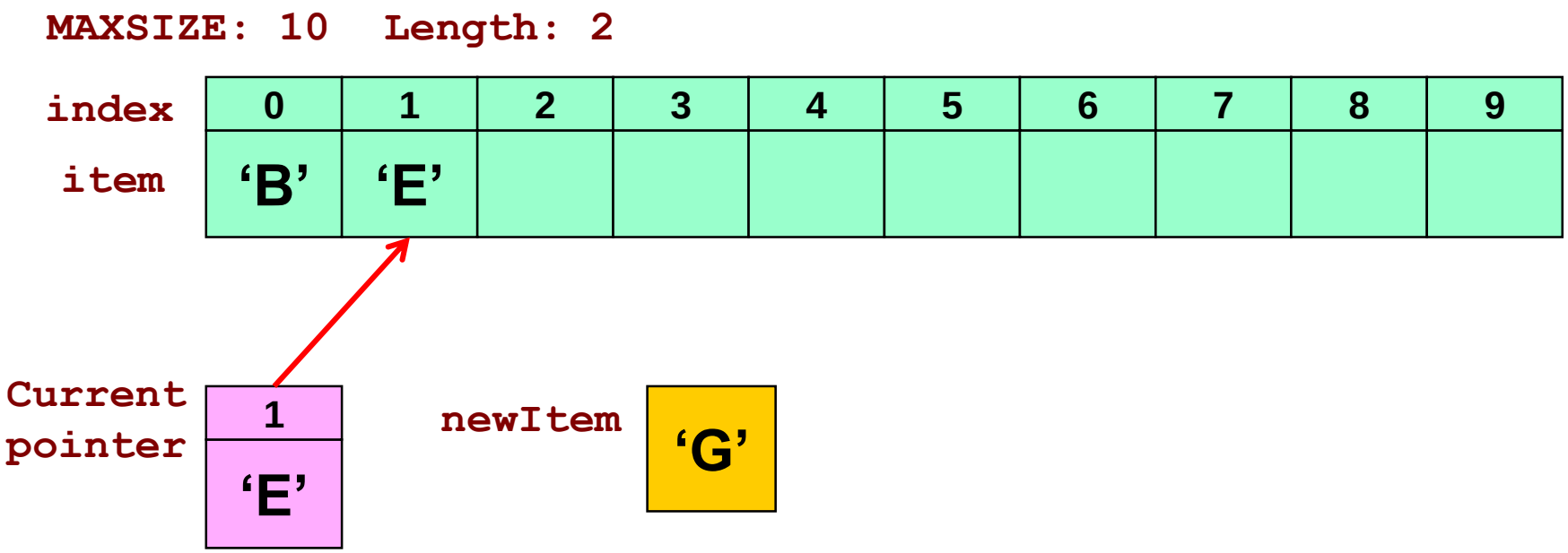
Lab02: Add (2/3)

◎ Add new item: Add to last



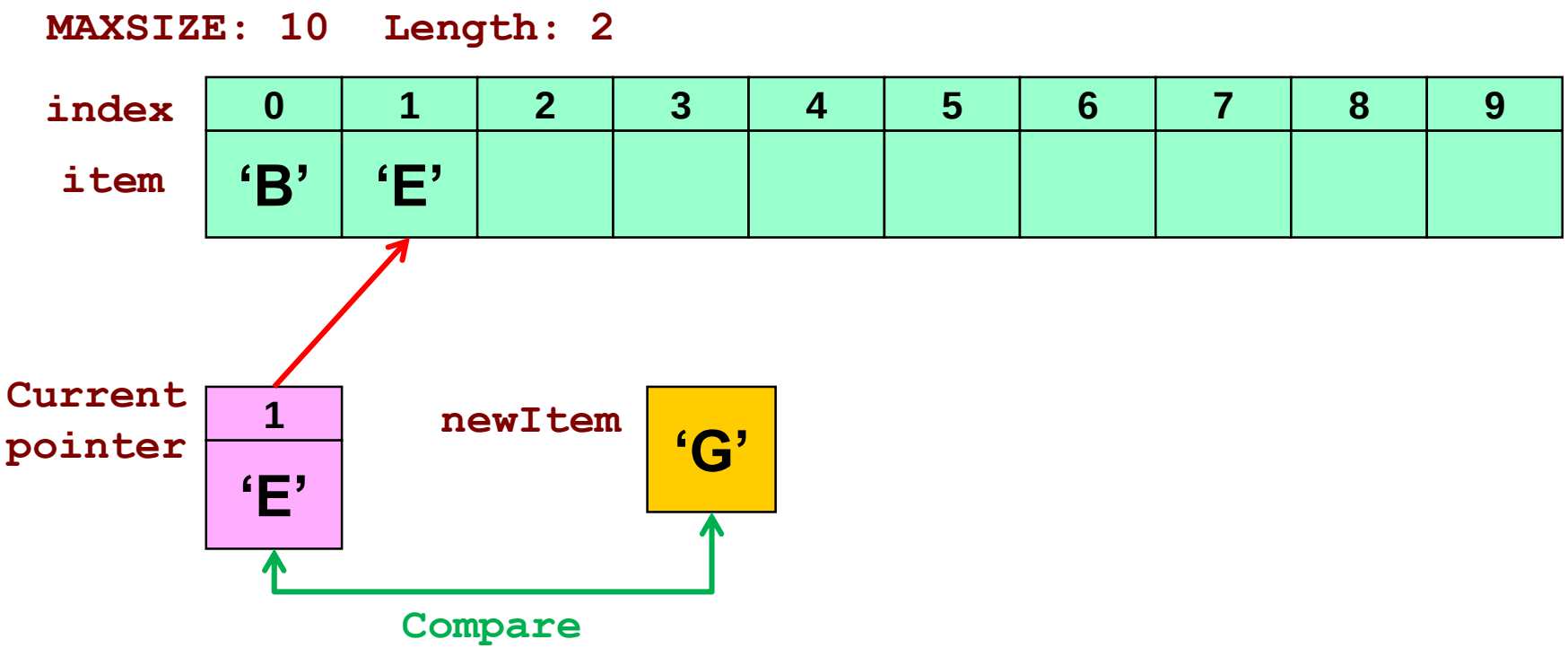
Lab02: Add (2/3)

◎ Add new item: Add to last



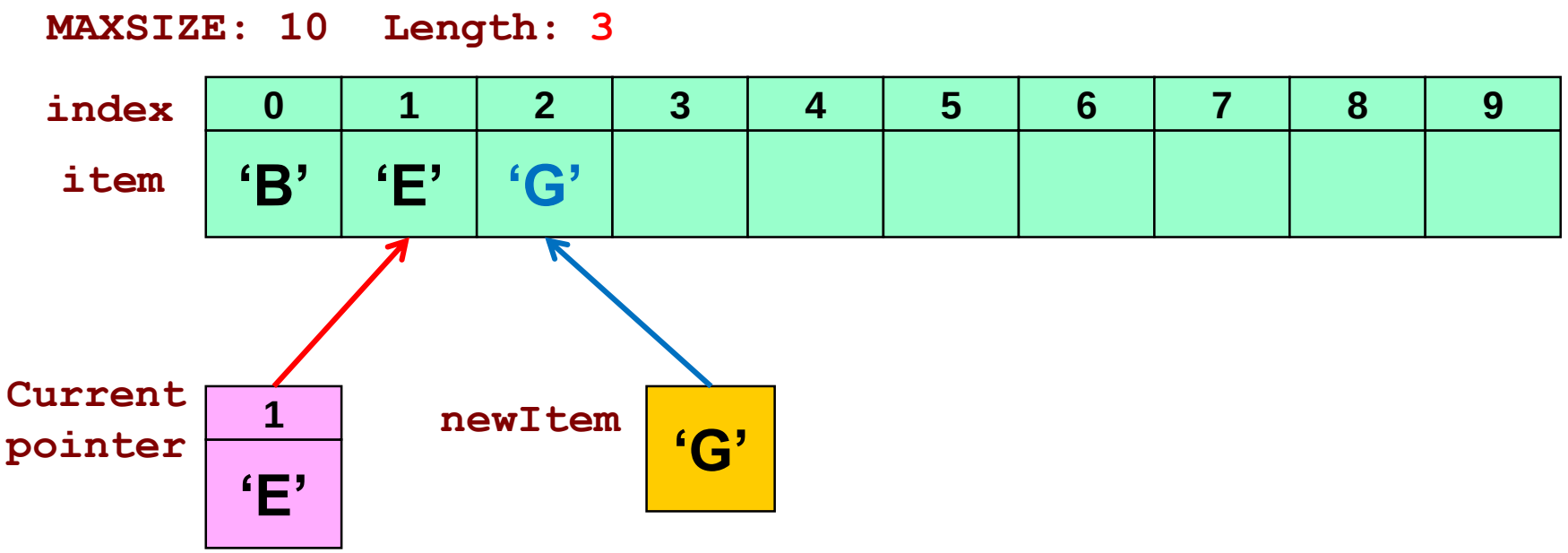
Lab02: Add (2/3)

◎ Add new item: Add to last



Lab02: Add (2/3)

◎ Add new item: Add to last



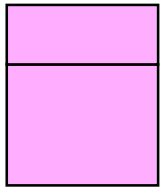
Lab02: Add (3/3)

◎ Add new item: Add to middle

MAXSIZE: 10 Length: 3

index	0	1	2	3	4	5	6	7	8	9
item	'B'	'E'	'G'							

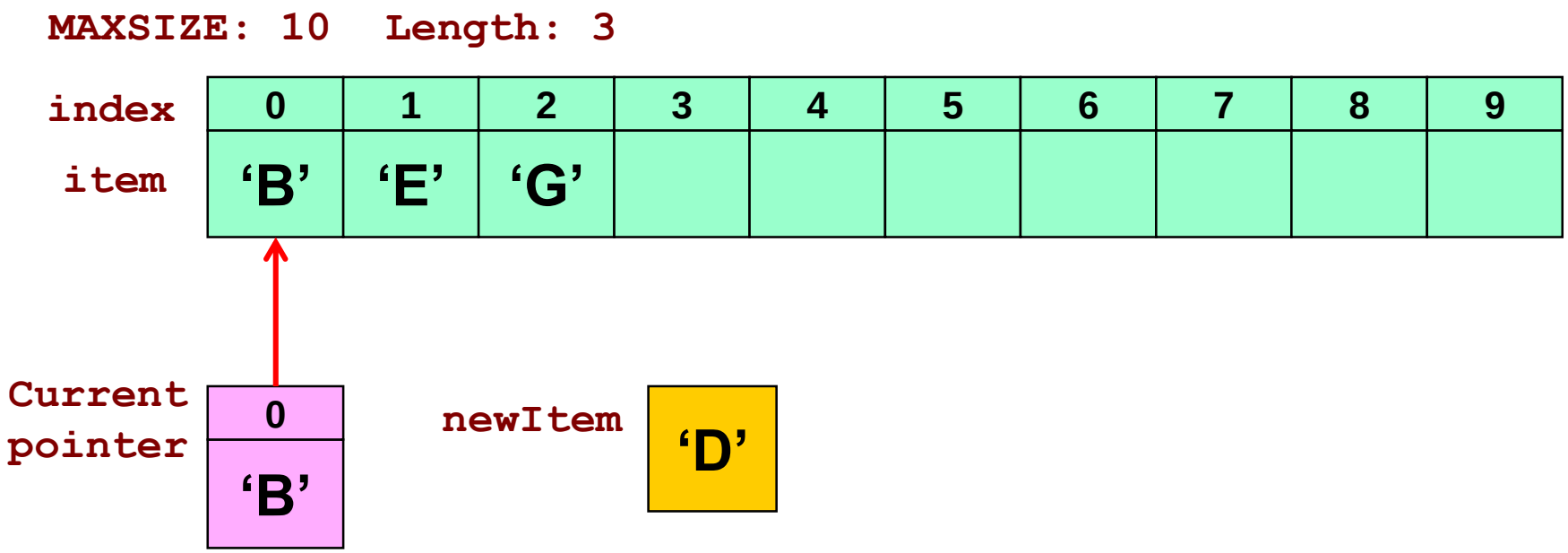
Current pointer



newItem

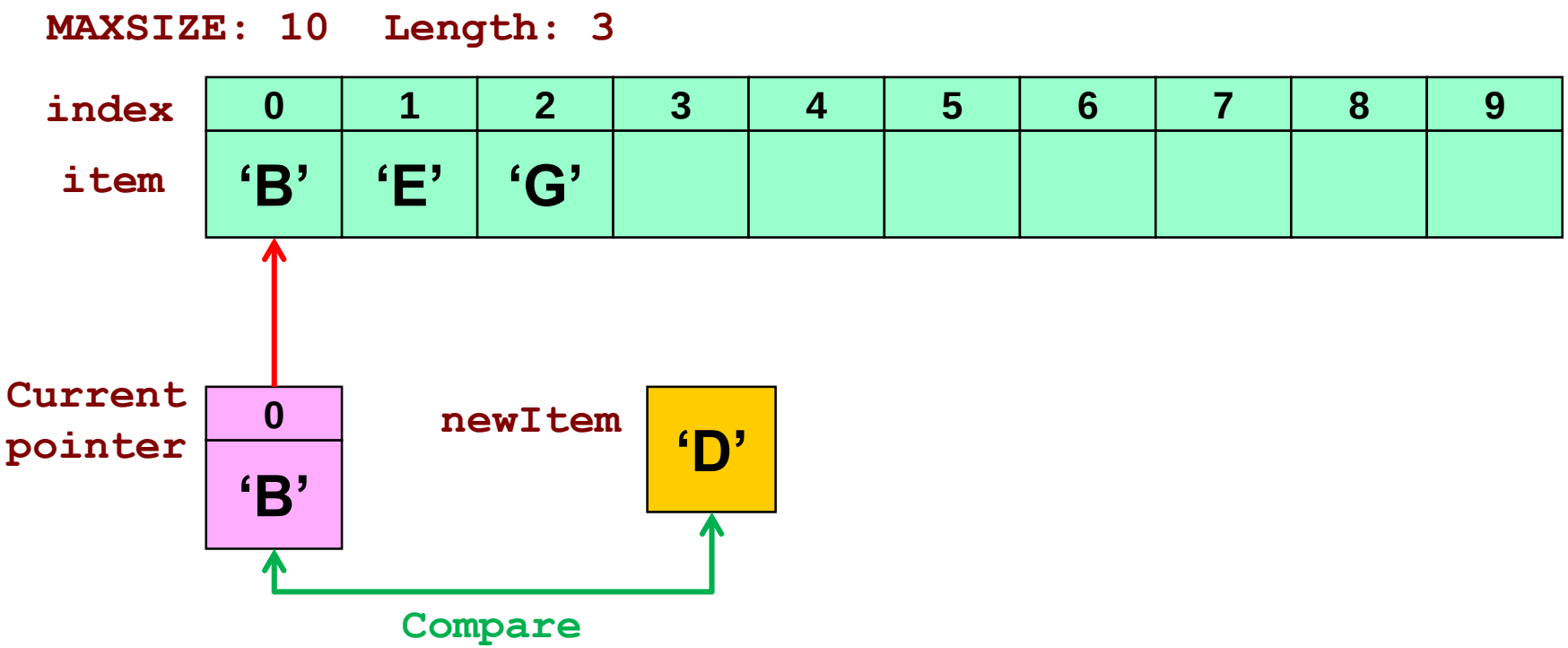
Lab02: Add (3/3)

◎ Add new item: Add to middle



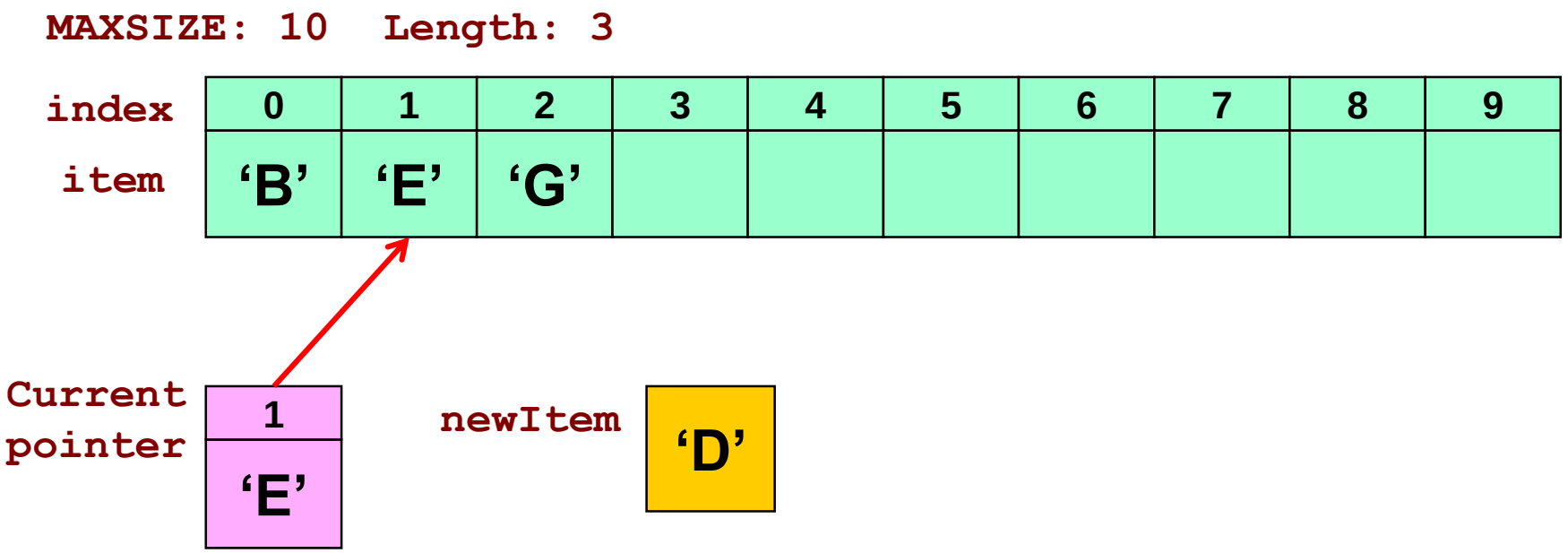
Lab02: Add (3/3)

◎ Add new item: Add to middle



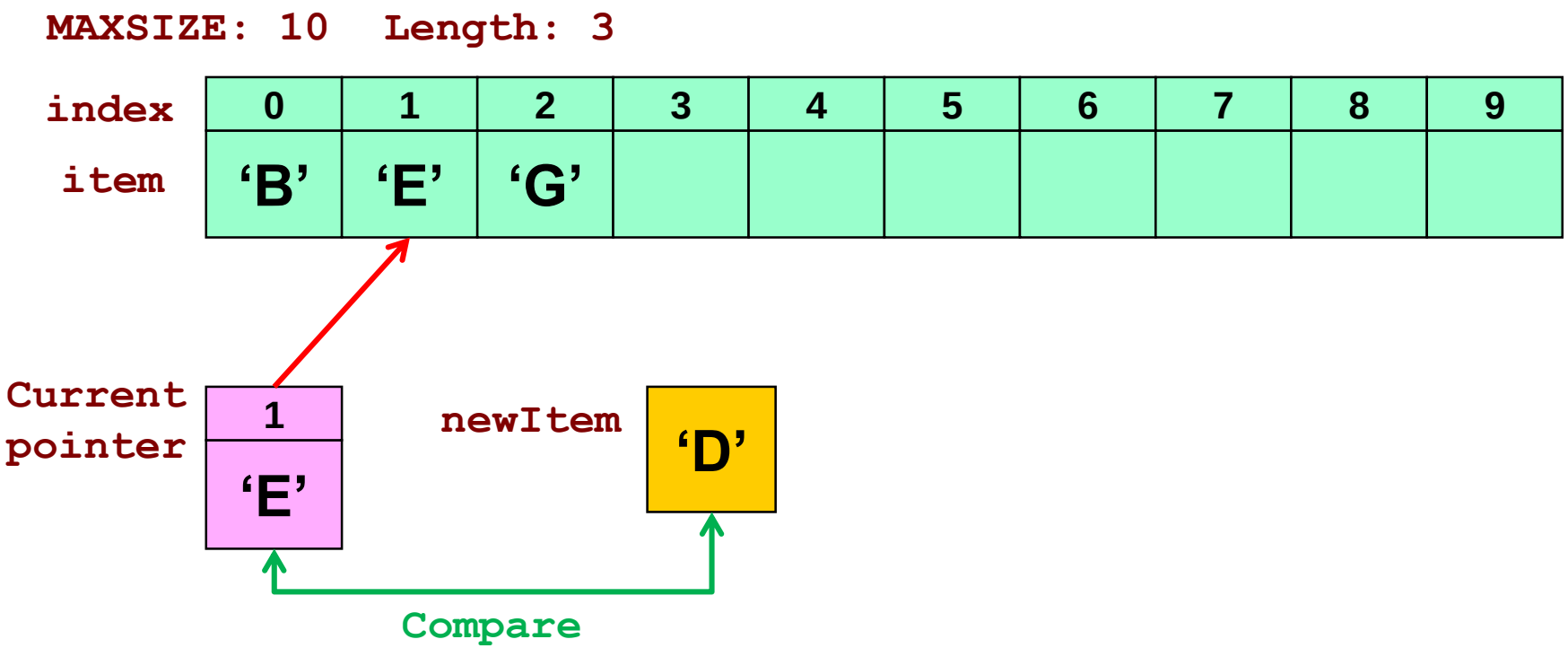
Lab02: Add (3/3)

◎ Add new item: Add to middle



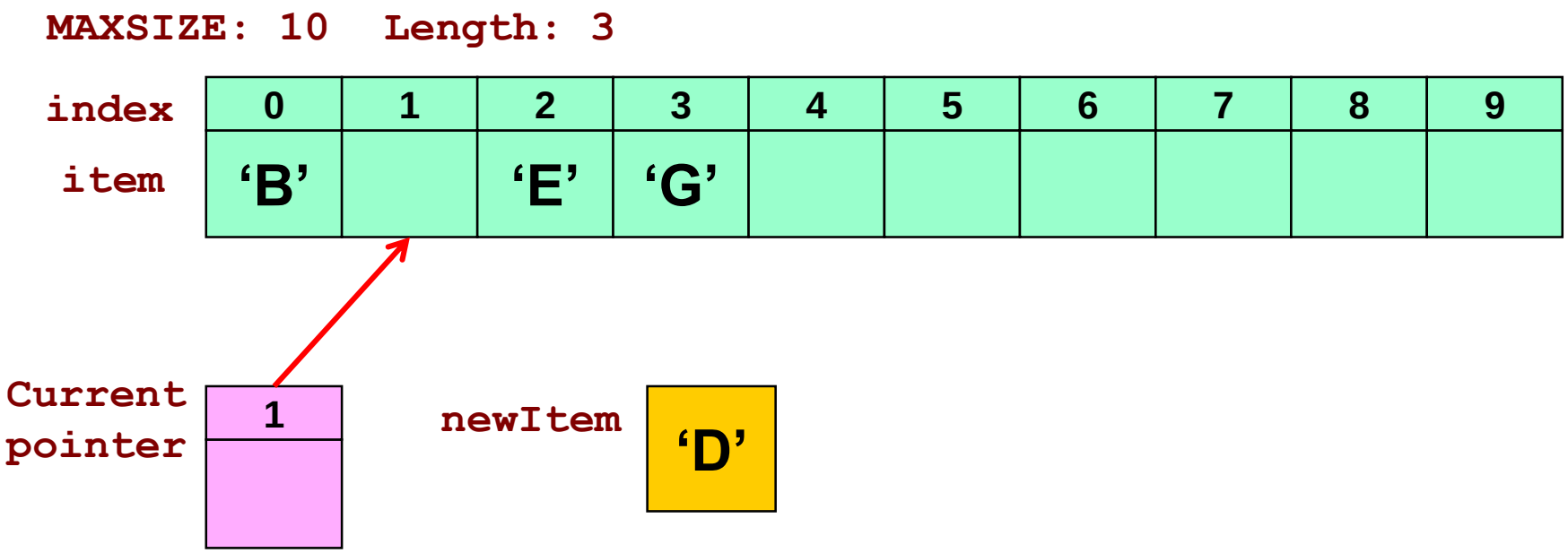
Lab02: Add (3/3)

◎ Add new item: Add to middle



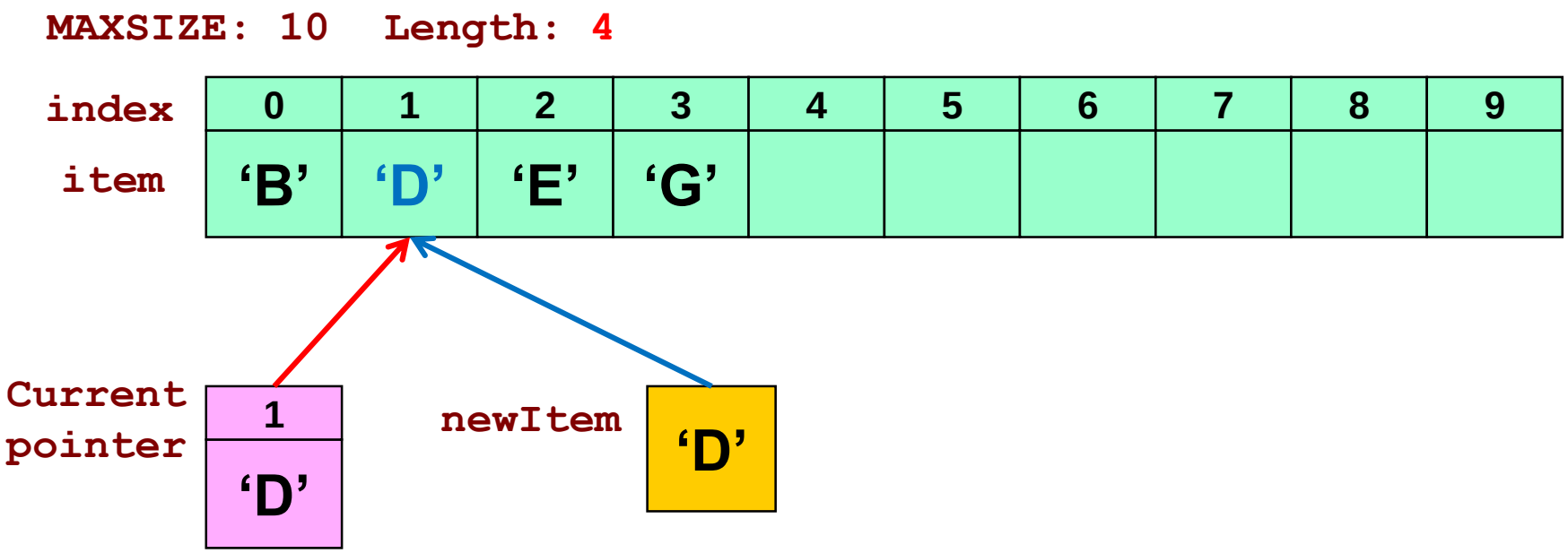
Lab02: Add (3/3)

◎ Add new item: Add to middle



Lab02: Add (3/3)

◎ Add new item: Add to middle



Add

- 새로운 항목의 다음 3 방법으로 이루어 진다.
  - List가 비어 있는 경우
    - 첫번째에 추가
  - 새 항목이 마지막 항목(index (length-1)에 있는 항목)보다 클 때
    - 맨 뒤(index (length)에 추가
  - 그 밖의 경우
    - 현재 index 이후에 있는 모든 항목을 한 칸씩 뒤로 이동
    - 현재 index에 새 항목 추가
- 구현을 위해서는 새 항목을 list에 있는 항목들과 비교하여 새 항목의 들어가 위치를 찾는다.

Algorithm

- ◎ List가 꽉 차있는 경우 0을 리턴
- ◎ List가 비어 있으면
  - ☞ 맨 앞에 삽입, length 증가, 1을 리턴
- ◎ 새 항목의 삽입 위치 iPos를 검색
- ◎ iPos==length이면
  - ☞ 인덱스 length에 추가, length++, 1리턴
- ◎ iPost<length이면
  - ☞ iPos부터 length-1 사이의 모든 항목을 뒤로 한 칸씩 이동
  - ☞ iPos에 추가,
  - ☞ Length++, 1리턴

Algorithm

```
// List가 꽉 차있는 경우 0을 리턴
// List가 비어 있으면
      //맨 앞에 삽입, length 증가, 1을 리턴
// 새 항목의 삽입 위치 iPos를 검색
      //iPos==length이면
       // 인덱스 length에 추가, length++, 1리턴
     // iPost<length이면
       // iPos부터 length-1 사이의 모든 항목을 뒤로 한 칸씩 이동
       // iPos에 추가,
       // Length++, 1리턴
```

Coding

```
Int SortedList:: Add(ItemType inData)
{
    // List가 꽉 차있는 경우 0을 리턴
    if (IsFull()) return 0;
    // List가 비어 있으면
    if (length==0) {
     //맨 앞에 삽입, length 증가, 1을 리턴
        m_Array[length++]=inData;
     }
    // 새 항목의 삽입 위치 iPos를 검색
    // 끝에 도달하거나 현재 data가 클 때까지
    bool found=false;  Int iPos=0;
    while (!found && iPos<length){
        switch(m_Array[iPos].Compare(inData))  {
        case EQUAL:
                cout<<“ the same item exist in the list\n”;
                return 0;
        case GREATER:
                found=true; break
         case LESS:
                iPos++; break;
          }
    }
//iPos==length이면
    if (iPos==length) {
          // 인덱스 length에 추가, length++, 1리턴
          m_Array[iPos]=inData;
          length++; return 1;
        }
    // iPost<length이면
          // iPos부터 length-1 사이의 모든 항목을 뒤로 한 칸씩 이동
          for (int i=length; i>iPos; i--)
                m_Array[i] = m_Array[i-1];
          // iPos에 추가,
        m_Array[iPos]=inData;
        // Length++, 1리턴
        Legnth++;
        return 1;
 }
```

Retrieve 함수 작성

- inData에 찾고자 하는 항목의 primary key가 저장되었고 가정
- List를 차례대로 방문하면서 inData의 primary key와 일치하는 항목을 찾는다.
  - curPos를 첫번째 항목을 가리키도록 세팅
  - curPos를 이동하면서 inData의 key와 비교한다. 만약 다음의 경우에는 이동을 중단하고 조치를 취한다.
    - curPos의 항목이 inData의 키와 일치 하는 경우
      - ✓ 찾은 항목을 inData에 복사하고 1을 리턴
    - curPos가 맨 끝에 도달하거나 curPos의 항목의 Key가 inData의 key보다 클 경우
      - ✓ 존재하지 않음으로 0을 리턴

** while loop을 사용할 경우 loop 종료조건은?

- M_Array[curPos]==inData의 Key
- M_Array[curPos] > inData의 Key
- curPos>=m_Length

Retrieve 함수 작성

```
// inData에 찾고자 하는 항목의 primary key가 저장되었고 가정
// List를 차례대로 방문하면서 inData의 Key와 일치하는 항목을 찾는다.
// curPos를 첫번째 항목을 가리키도록 세팅
curPos = 0;
// curPos를 이동하면서 현재 항목의 key와 inData의 key를  비교한다. 만약 다음의 경우에는
이동을 중단하고 조치를 취한다.
While (m_Array[curPos].Compare(inData)==LESS && curPos<m_Length)
{
   curPos++;
}
// curPos의 항목의 key가 inData의 Key와 일치 하는 경우
If (m_Array[curPos].Compare(inData)==EQUAL)  {
    // 찾은 항목을  inData에 복사하고 1을 리턴
    inData=m_Array[curPos]; return 1;
}
Else  //현재 항목의 key가 inData의 Key보다 크거나 curPos가 맨 끝네 도달하는 경우,  존재하지
않음으로  0을 리턴
       Return 0;
}
```

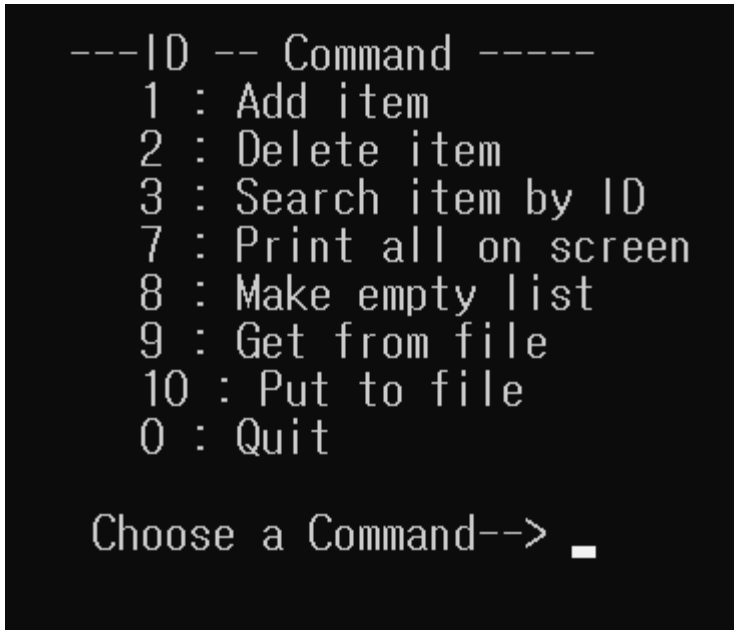
Retrieve 함수 작성-for loop 사용

```
// inData에 찾고자 하는 항목의 primary key가 저장되었고 가정
SortedList::Retrieve(ItemType& inData)
{
// List를 차례대로 방문하면서 inData의 Key와 일치하는 항목을 찾는다.
// curPos를 첫번째 항목을 가리키도록 세팅
Int curPos = 0;
// curPos를 이동하면서 현재 항목의 key와 inData의 key를  비교한다.
For(; m_Array[curPos].Compare(inData)==LESS && curPos<m_Length; curPos++);

// curPos의 항목의 key가 inData의 Key와 일치 하는 경우
If (m_Array[curPos].Compare(inData)==EQUAL)  {
    // 찾은 항목을  inData에 복사하고 1을 리턴
    inData=m_Array[curPos]; return 1;
}
Else  //현재 항목의 key가 inData의 Key보다 크거나 curPos가 맨 끝네 도달하는 경우,  존재하지 않음으로  0을 리턴
        Return 0;
}
```

예제: Application class 정의

- Lab 01과 동일하게 아래와 같은 명령을 수행할 수 있는 application class를 작성



과제 : 응용 프로그램 작성

◎ 내용:

☞ SortedList Class에 다음 함수 추가

- Retrieve_Binary(ItemType& inData): Binary Search를 이용하여 Retrieve_Seq()와 동일한 기능을 수행
- Replace(ItemType inData): 리스트에서 inData의 ID와 일치하는 항목을 찾아서 inData의 자료로 치환. 치환하면 1을 그렇지 않으면 0을 리턴.

☞ Application Class에 다음 함수 추가

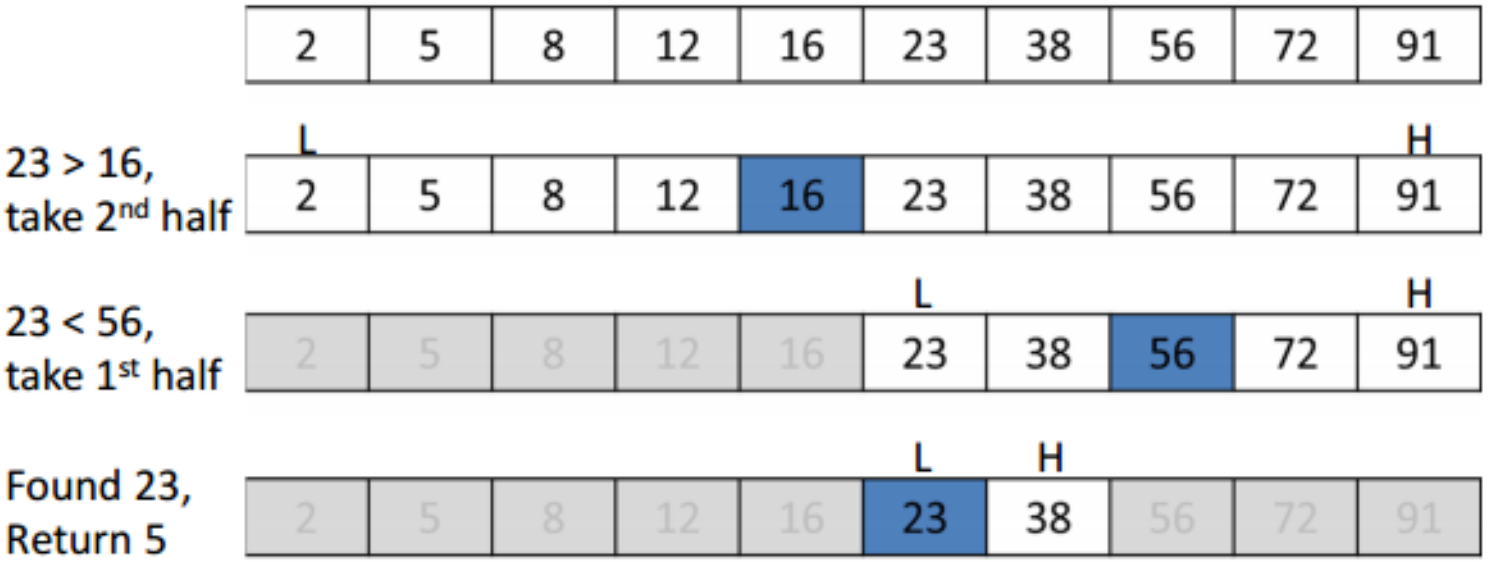
- SearchById_BinaryS():

SearchByName(): 키보드에서 이름을 읽고 list에서 이 이름을 가진 모든 레코드를 찾아서 화면에 출력. 예) 김이 입력되면 이름에 김이 포함되는 모든 항목(“김”, “김파래“, “김연아“)을 화면에 출력

- ReplaceItem(): 키보드에서 새 항목을 받아서 Replace()함수를 호출하여 기존 항목을 치환
- Run(): 새 명령을 지원하도록 수정

과제 참조: Binary Search

If searching for 23 in the 10-element array:



SarchByName

◎ 검색할 이름(inName)을 입력

1. Iterator를 초기화
2. GetNextItem()을 이용하여 List의 항목을 가져와 inName과 비교
   ☞ 일치하면 화면에 출력
3. 마지막 항목이 아니면 2를 반복

참고자료: Debugger

◎ Debugger

☞ A debugging tool that is used to test and debug other programs.

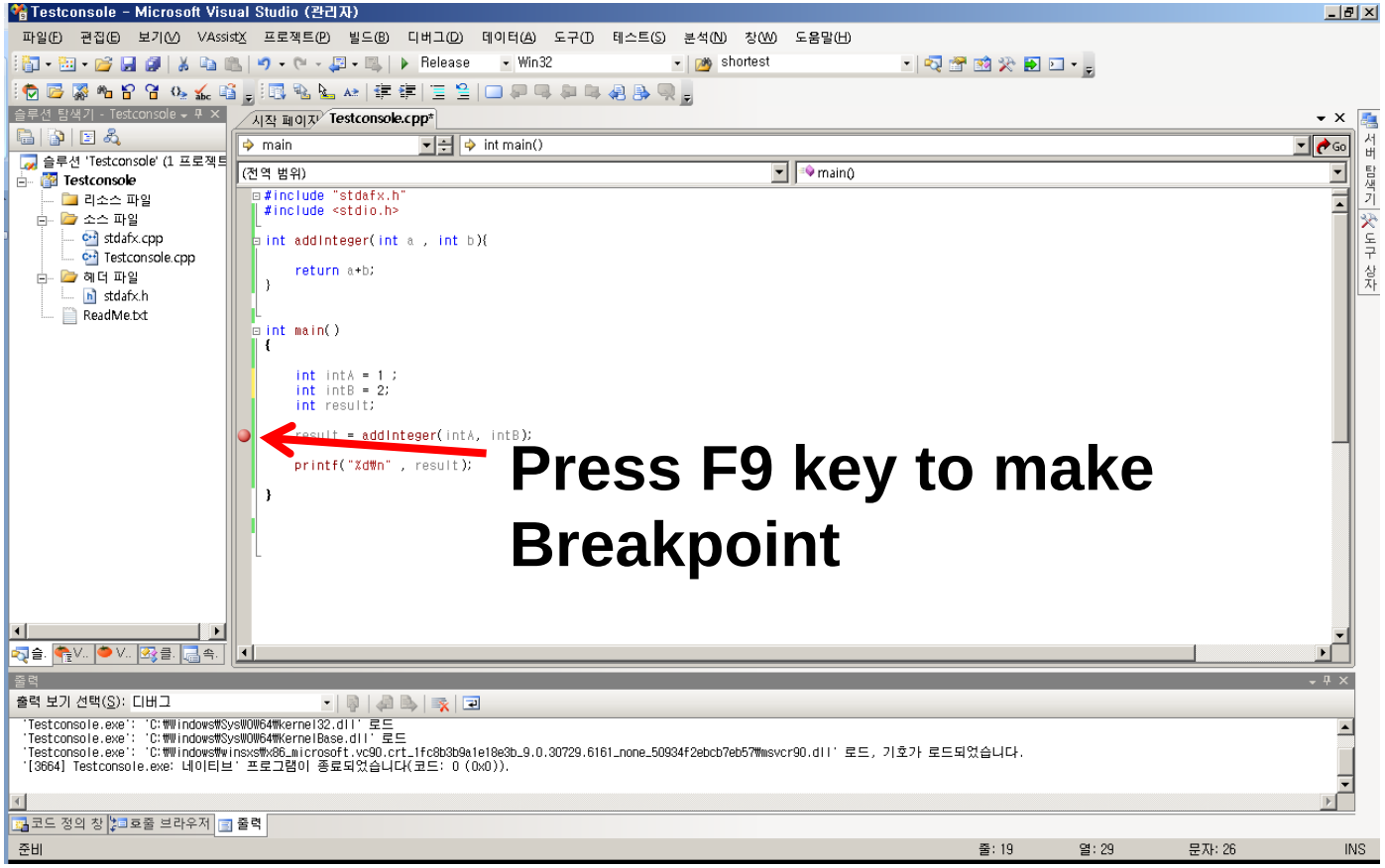
◎ Debuggee

☞ A process or application upon which a debugger acts. The process that is being debugged.

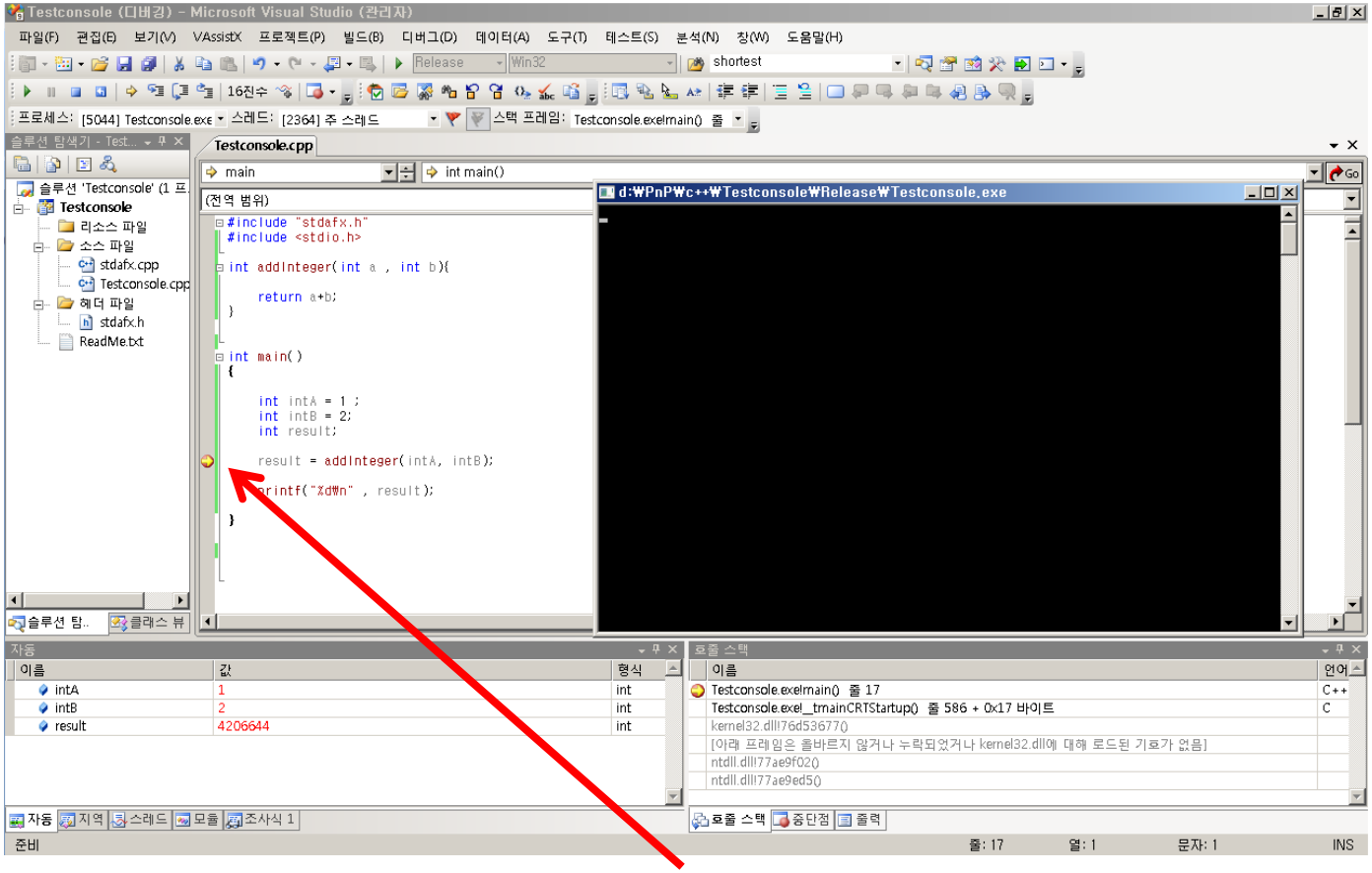
Sorts of Debug Shortcut (Visual Studio)

- Start Debugging(F5)
  - Start Debugging. Application will stop at the first Breakpoint.
- Stop Debugging(Shift + F5)
  - Stop Debugging. Debugger will terminate the application.
- Make Break Point(F9)
  - When you press “F9” key, red point will appear the headline of source code. Debugger will break the application when meet the break point. Breakpoint can be made while coding time and debugging time.
- Step Into(F10)
  - Run into the function(method). Can’t enter library function(method) as like API, Standard library.
- Step Over(F11)
  - Run over the function(method).

Set ‘Break point’

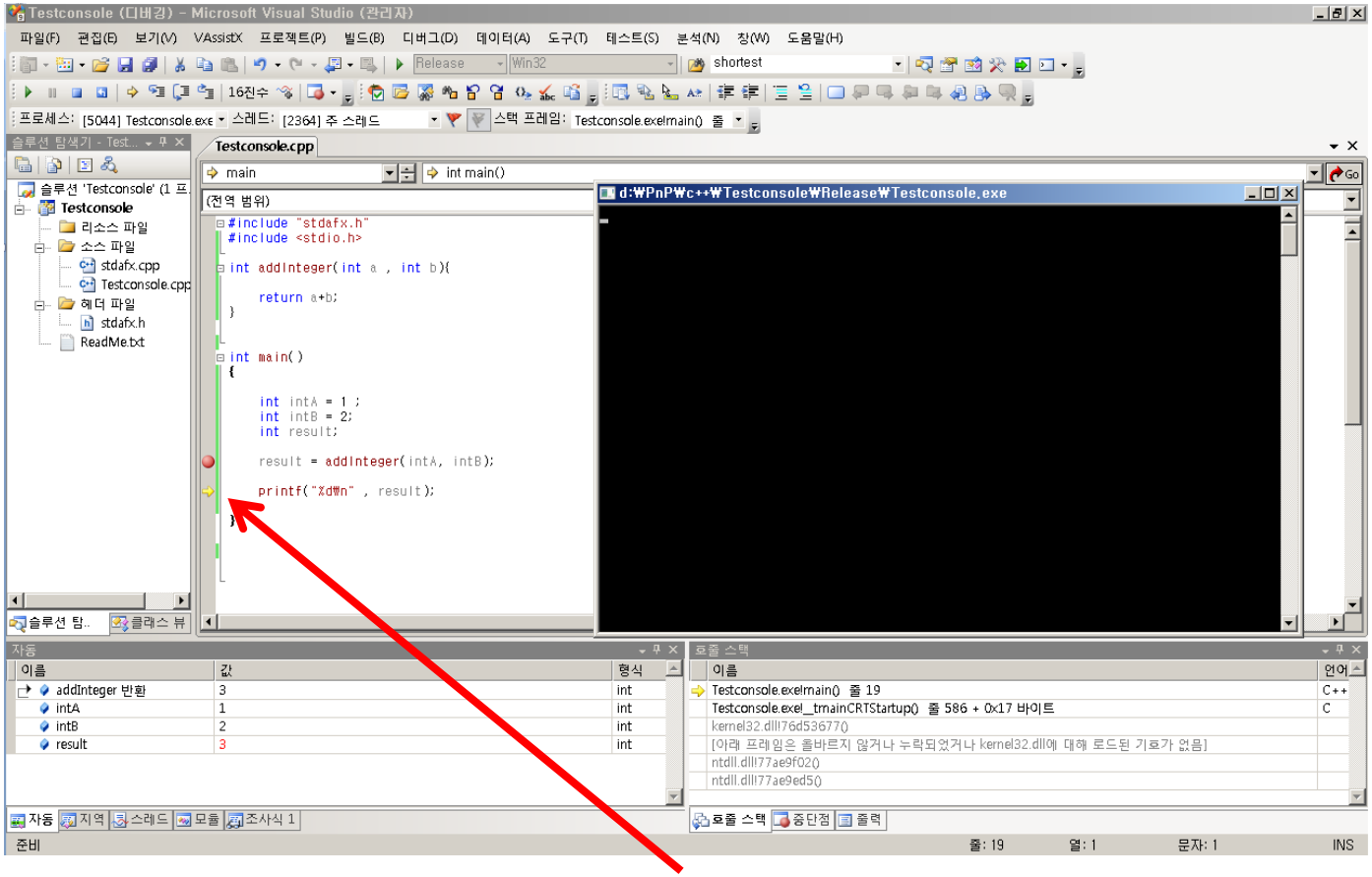


Run debug mode



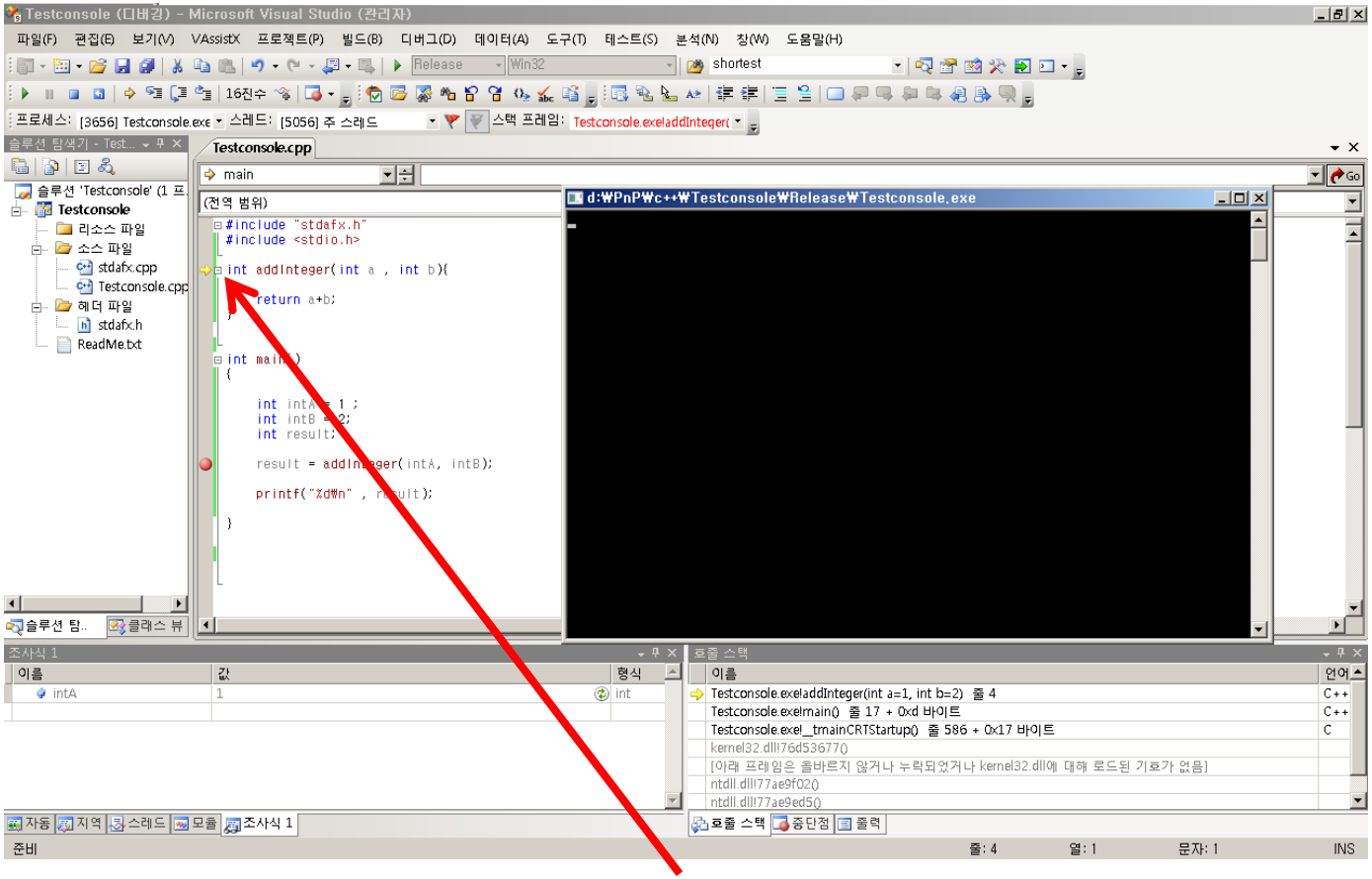
- Press F5 key to enter debug mode. Application will stop at first breakpoint.
- Yellow arrow indicates the source code line that will be implemented very next time.

Step over (F10)



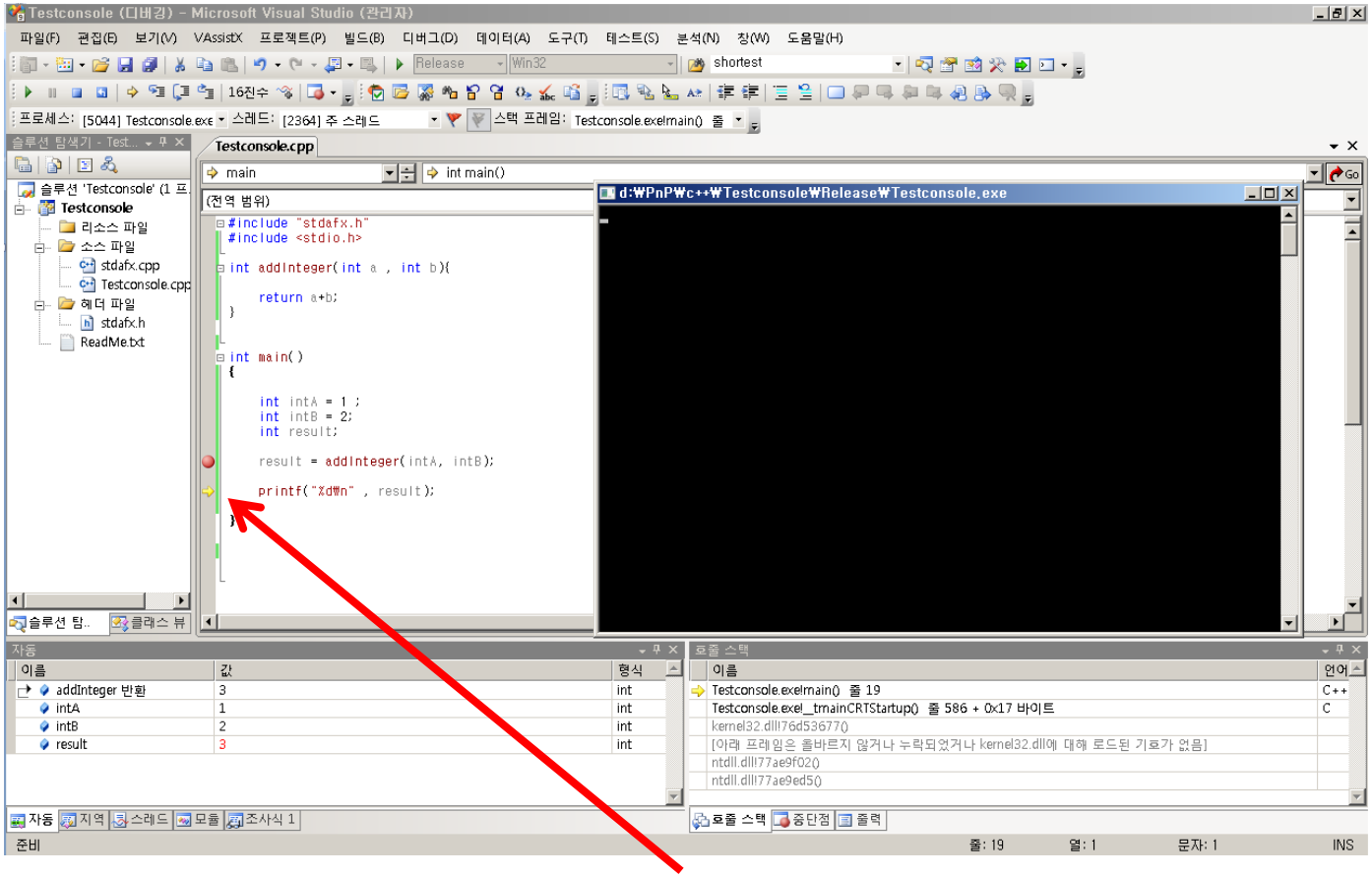
- Press F10 key to step over the function “addinteger()”.
- Yellow arrow will indicate very next line.

Step into (F11)



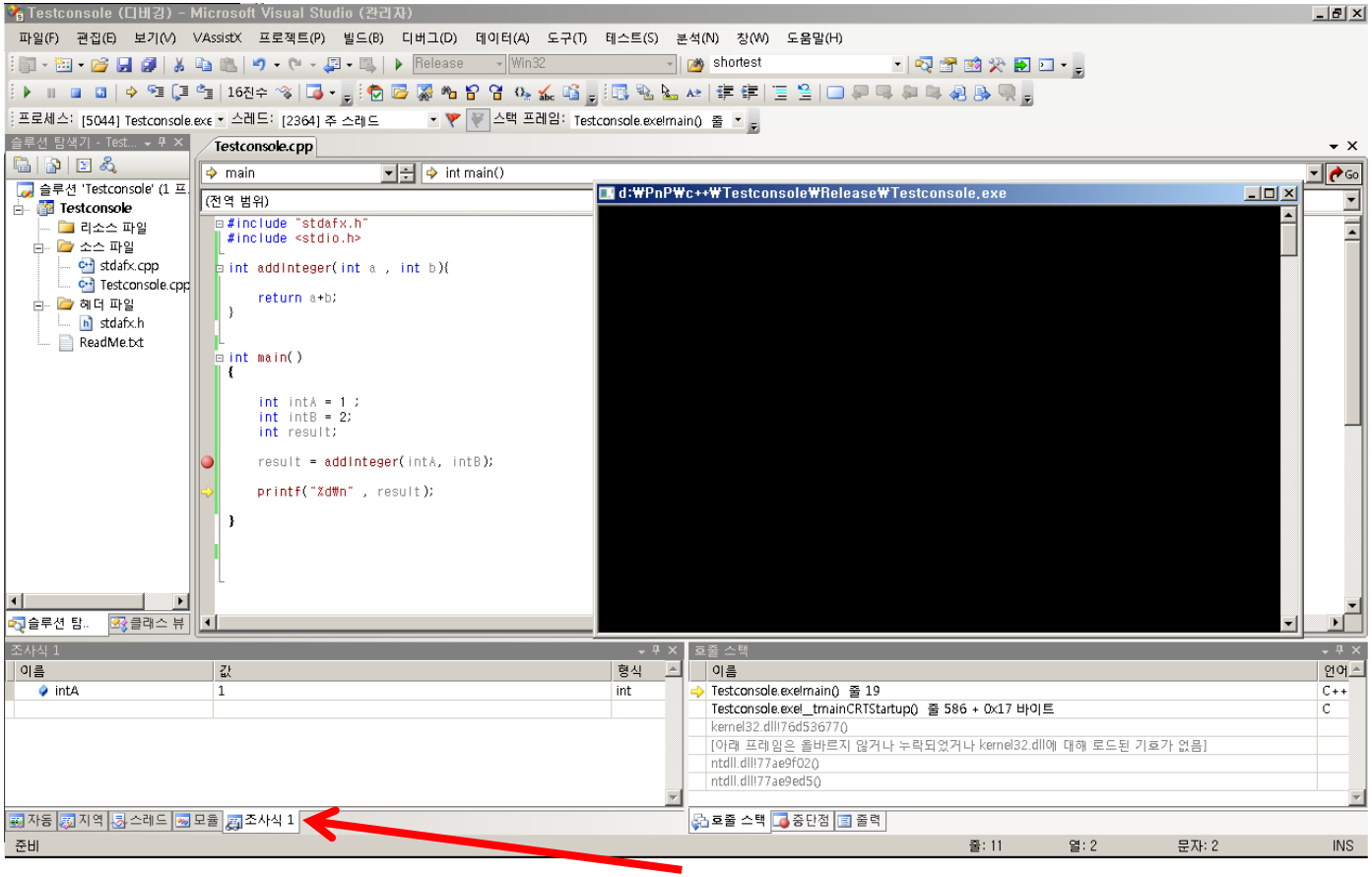
- Press F11 key to step into the function “addinteger()”.
- Yellow arrow will indicate the insight of function.

Check value of variables



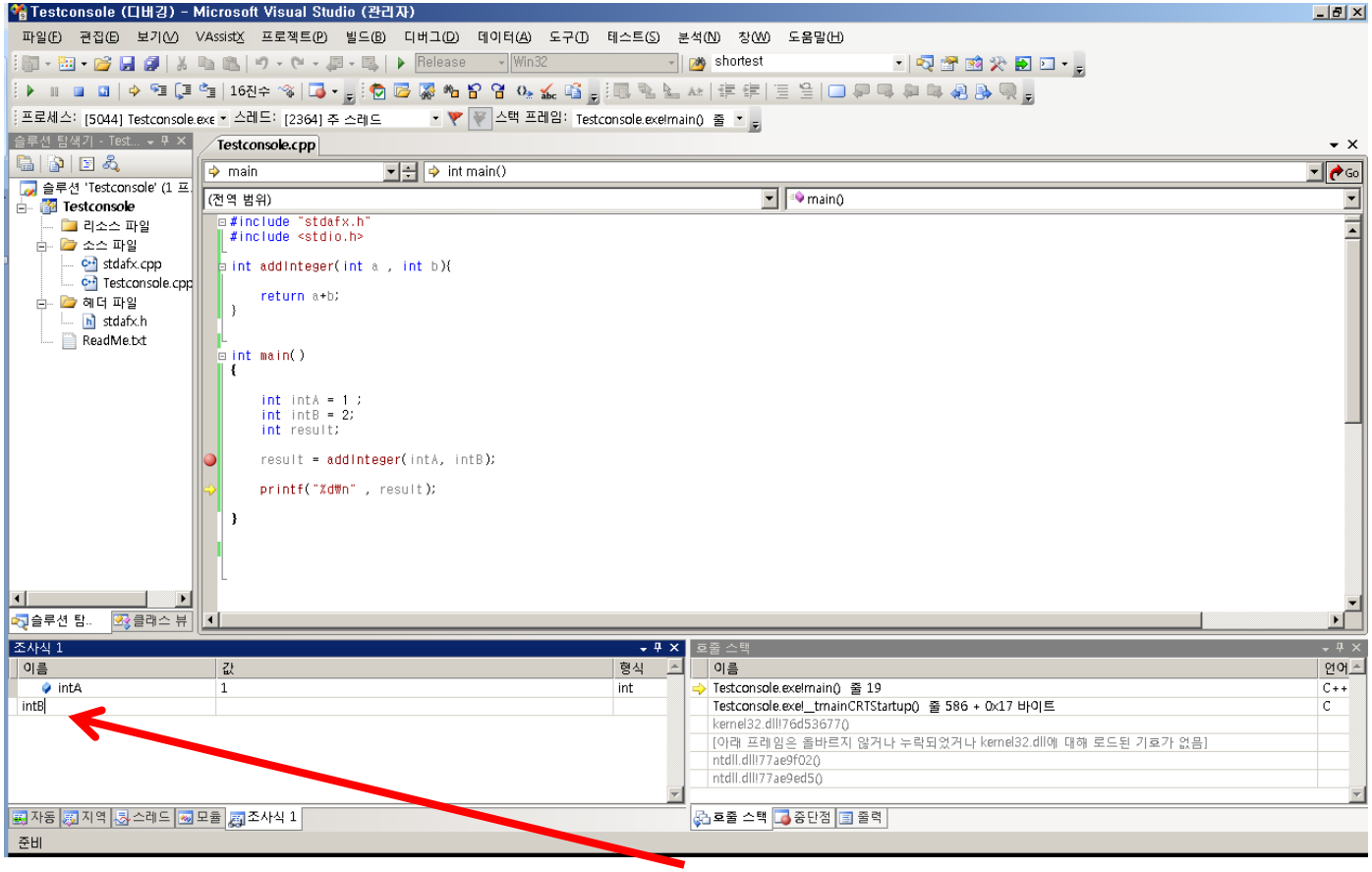
- You can confirm the variables' value.
- Red character means that the variable's value is changed.

Check value of variables



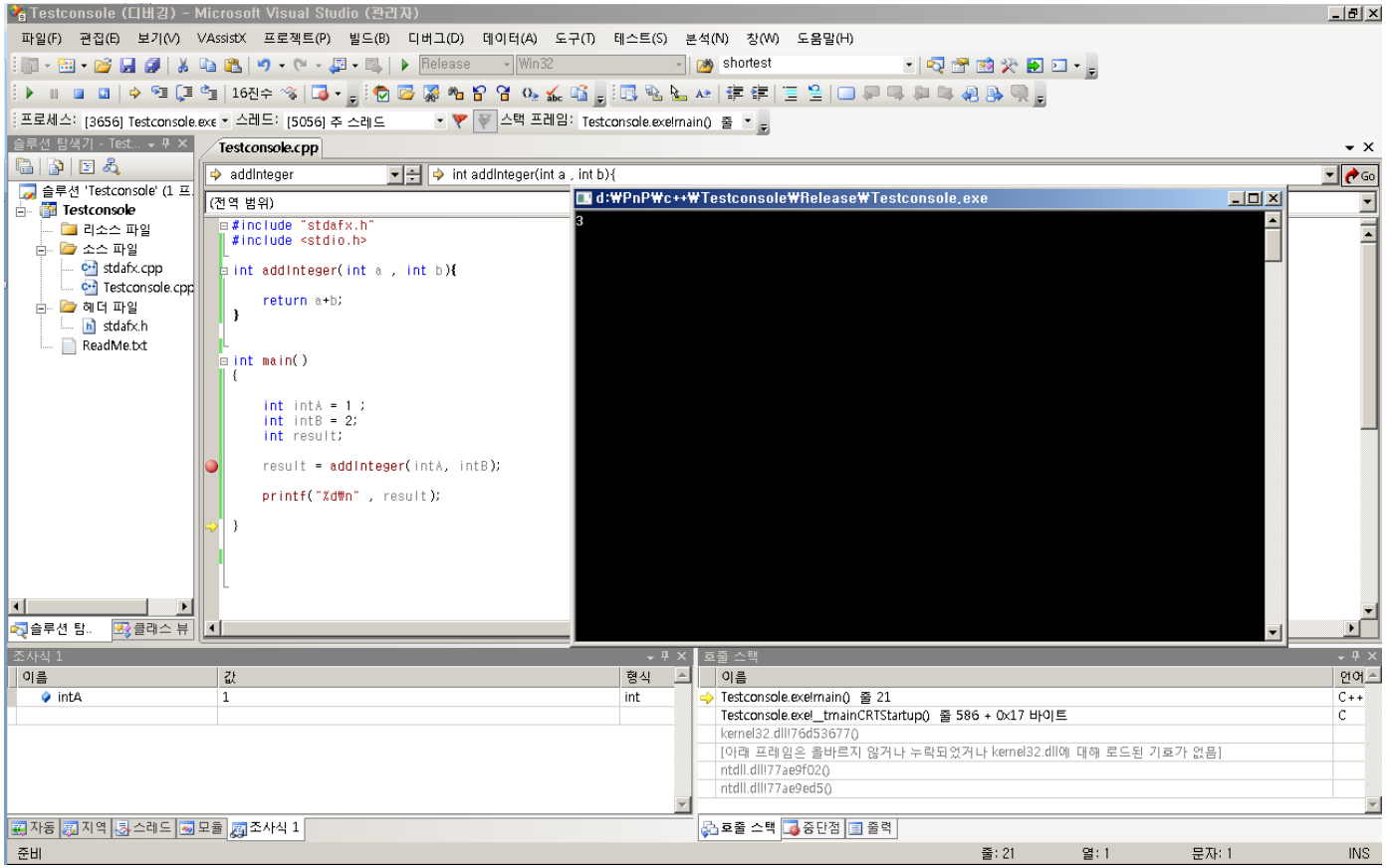
◎ You can confirm the variables' value by expression(조사식) window too.

Check value of variables



◎ You can also add variables that you want to confirm the value.

End of debug


Testconsole (디버깅) - Microsoft Visual Studio (관리자)
Testconsole.cpp
addInteger
int addInteger(int a , int b){
d:₩PnP₩c++₩Testconsole₩Release₩Testconsole.exe
3
#include "stdafx.h"
#include <stdio.h>
int addInteger(int a , int b){
return a+b;
}
int main( )
{
int intA = 1 ;
int intB = 2;
int result;
result = addInteger(intA, intB);
printf("%d₩n" , result);
}
intA
1
int
Testconsole.exe!main() 줄 21
Testconsole.exe!__tmainCRTStartup() 줄 586 + 0x17 바이트
kernel32.dll!76d536770
[아래 프레임은 올바르지 않거나 누락되었거나 kernel32.dll에 대해 로드된 기호가 없음]
ntdll.dll!77ae9f02()
ntdll.dll!77ae9ed5()
준비