Pointer – 실습

Section 1 실습 예제
Section 2 제출 실습 문제

▪ 실습 문제

- **카이사르 암호법**

- **검사 받은 사람은 집에 갑시다.**
  - 먼저 하고 먼저 가자.
  - 가기 전에 제출 필수.

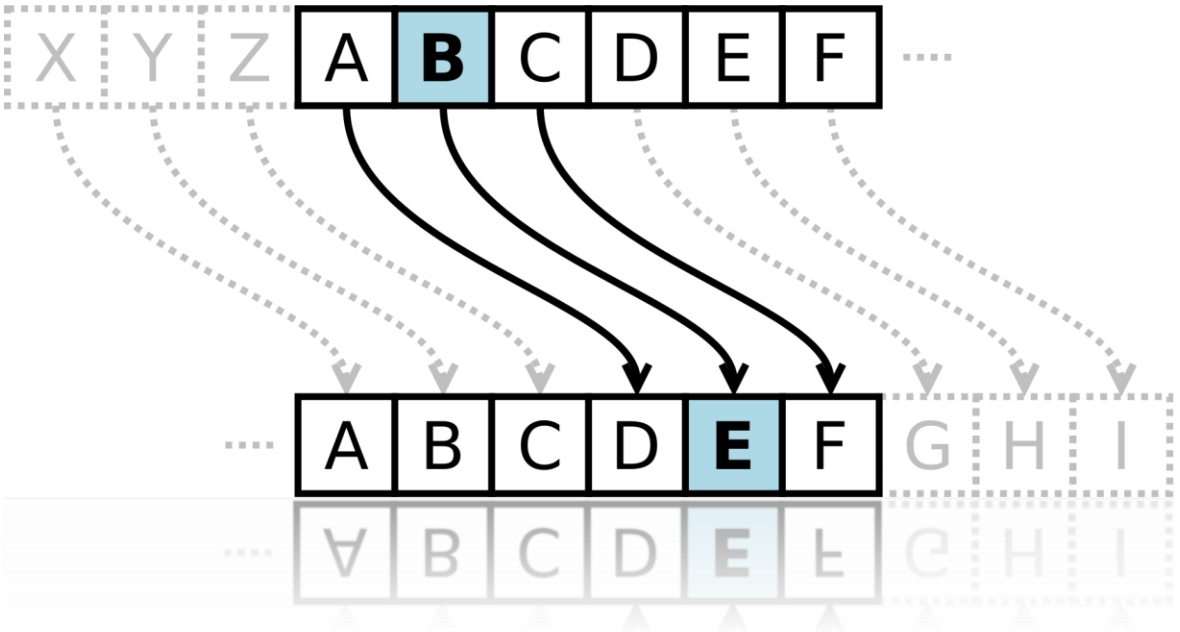
1. 실습 예제

카이사르 암호법 – 이전 과제

- **목적**
  - 문자형 (ascii 코드)의 정의 형태 이해
  - 문자형의 연산에 따른 값 변화 이해
  - 문자열에서 각 자리 문자의 접근법 이해

define

- **입력**
  - n자리 문자열: 길이는 자유, but 영문자에 한해 수행된다 가정
  - 키 공간: 이동시킬 칸 수
  - 암호화 또는 복호화



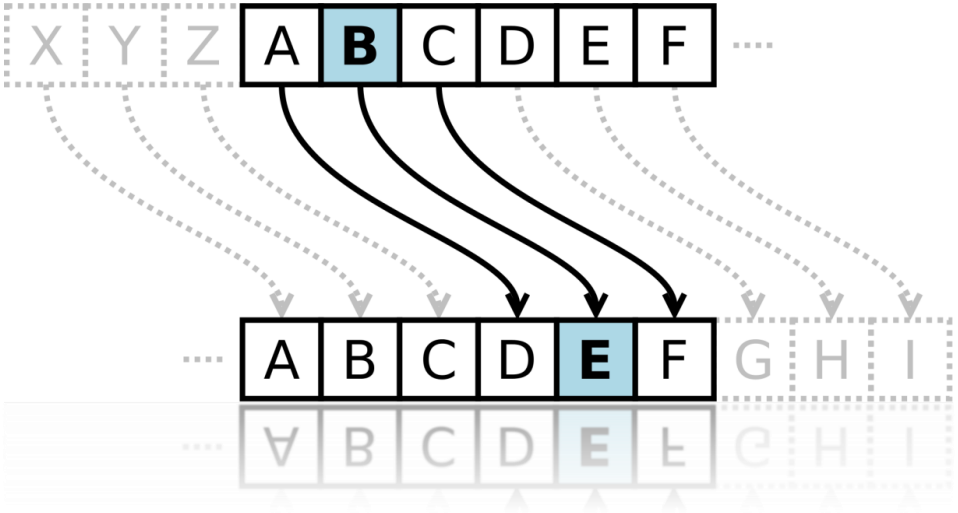
- **출력**
  - 암호화/복호화된 문자열

- **제한사항**
  - 문자와 숫자에 한정

1. 실습 예제

Caesar Cipher (Shift) – 포인터 응용

- **목적**
  - 문자형 (ascii 코드)의 정의 형태 이해
  - 문자형의 연산에 따른 값 변화 이해
  - 문자열에서 각 자리 문자의 접근법 이해
  - **포인터의 이해**
  - **배열 이름의 실체 이해**



- **입력**
  - n자리 문자열: 길이는 자유, but 영문자에 한해 수행된다 가정
    - 암호화되지 않은 문자열을 plainText, 암호화된 문자열을 cipherText라 한다.
  - 암호화 또는 복호화
  - 키 공간: 이동시킬 칸 수

- **출력**
  - 암호화/복호화 결과물

▪ 제출 실습문제

- 참고 – 소수 판별기

- n번째 소수 찾아내기

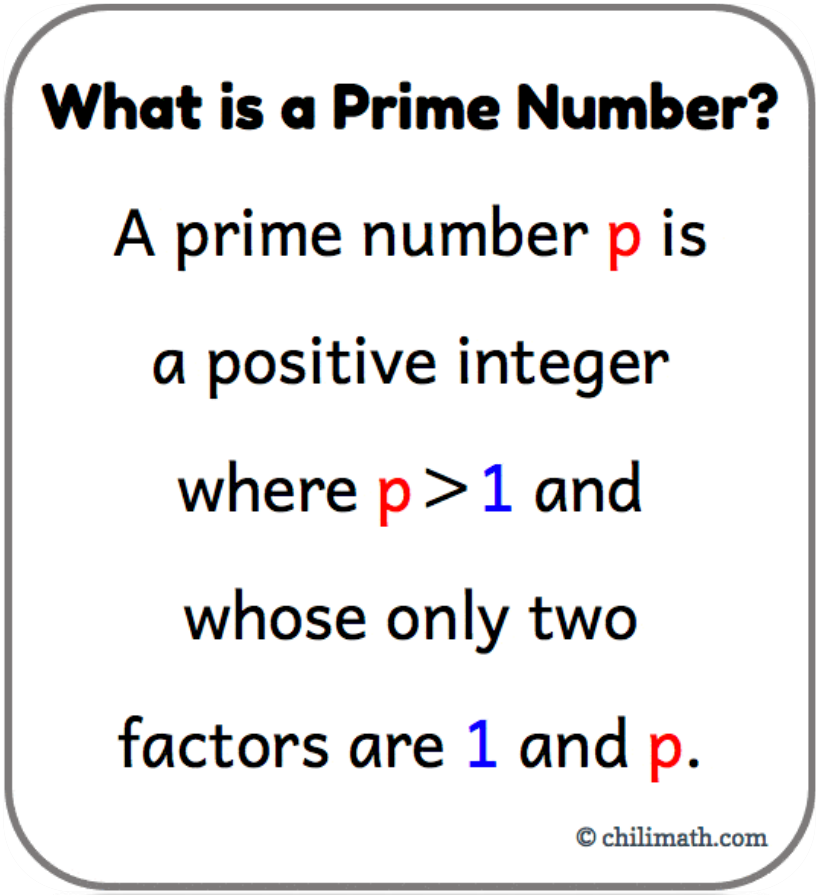
2. 제출 실습문제

참고 – 소수 판별기

- **목적**
  - 배열의 이해도 제고
  - for문의 응용
  - Pointer와 Index 개념의 정립

- **입력**
  - 1보다 큰 자연수

- **출력**
  - 소수인지 아닌지



2. 제출 실습문제

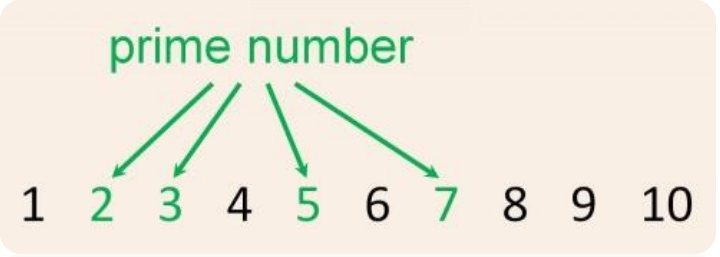
n번째 소수 찾아내기 – 개요

- **목적**
  - 1. 소수에 대한 이해
  - 2. 배열의 이해
  - 3. while 반복문 및 for 반복문 활용력 키우기
  - +) **인덱스**와 **포인터**의 이해

- **입력**
  - n: 구하고자 하는 소수는 몇 번째 소수인가?
    - Ex1) n = 1, 소수는 2
    - Ex2) n = 10, 소수는 29

- **출력**
  - n번째 소수
    - Ex1) n = 1, 소수는 2
    - Ex2) n = 10, 소수는 29

1~k까지 무한히 탐색



n번째 소수는 무엇?

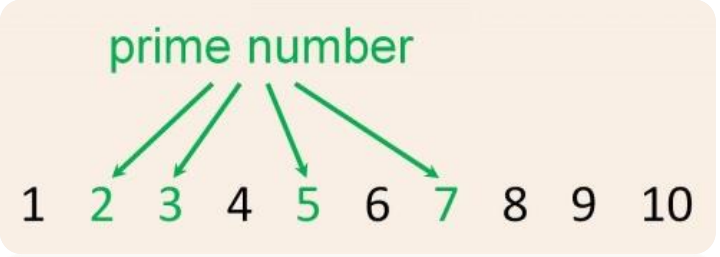
	2	3	5	7	11	13	17	19	23
29	31	37	41	43	47	53	59	61	67
71	73	79	83	89	97	101	103	107	109
113	127	131	137	139	149	151	157	163	167
173	179	181	191	193	197	199	211	223	227
229	233	239	241	251	257	263	269	271	277
281	283	293	307	311	313	317	331	337	347
349	353	359	367	373	379	383	389	397	401
409	419	421	431	433	439	443	449	457	461
463	467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593	599
601	607	613	617	619	631	641	643	647	653
659	661	673	677	683	691	701	709	719	727
733	739	743	751	757	761	769	773	787	797
809	811	821	823	827	829	839	853	857	859
863	877	881	883	887	907	911	919	929	937
941	947	953	967	971	977	983	991	997	

2. 제출 실습문제

n번째 소수 찾아내기 – 착안점

- **n번째 소수가 무엇인지 n을 입력 받아 n번째 소수를 구해보자!**
  - n번째가 무엇인가
  - 특정할 수 있나?

1~k까지 무한히 탐색



n번째 소수는 무엇?

	2	3	5	7	11	13	17	19	23
29	31	37	41	43	47	53	59	61	67
71	73	79	83	89	97	101	103	107	109
113	127	131	137	139	149	151	157	163	167
173	179	181	191	193	197	199	211	223	227
229	233	239	241	251	257	263	269	271	277
281	283	293	307	311	313	317	331	337	347
349	353	359	367	373	379	383	389	397	401
409	419	421	431	433	439	443	449	457	461
463	467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593	599
601	607	613	617	619	631	641	643	647	653
659	661	673	677	683	691	701	709	719	727
733	739	743	751	757	761	769	773	787	797
809	811	821	823	827	829	839	853	857	859
863	877	881	883	887	907	911	919	929	937
941	947	953	967	971	977	983	991	997	

- **그렇다면 사용해야 하는 반복문의 형태는?**
  - 1. for
  - 2. while
  - 3. do-while
  - 이 중 어떤 것이 가장 적절할까?

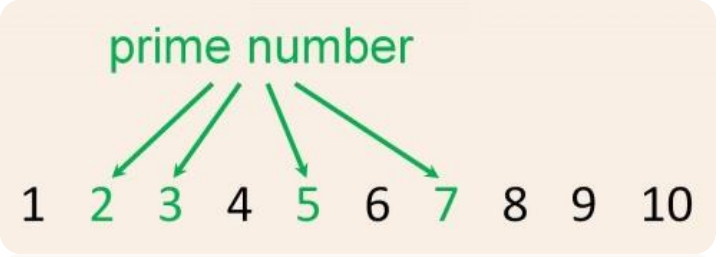
- **반복문의 중첩은 필요한가?**
  - 이전에 생성한 소수의 배열에 들어있는가
  - 이번에 찾은 숫자가 소수라면 N번째인가

2. 제출 실습문제

n번째 소수 찾아내기 – 구상

- **소수를 처음부터 하나씩 찾아가보자**
  - 첫 소수는 얼마?
  - 그러면 끝 소수는 얼마?

1~k까지 무한히 탐색



  - 발견된 소수는 어디에 저장?
    - 충분히 큰 크기의 배열
    - 거의 무한히 반복되는 반복문을 이용하여 작은 소수부터 하나씩 찾으며 담아가보자.

n번째 소수는 무엇?

	2	3	5	7	11	13	17	19	23
29	31	37	41	43	47	53	59	61	67
71	73	79	83	89	97	101	103	107	109
113	127	131	137	139	149	151	157	163	167
173	179	181	191	193	197	199	211	223	227
229	233	239	241	251	257	263	269	271	277
281	283	293	307	311	313	317	331	337	347
349	353	359	367	373	379	383	389	397	401
409	419	421	431	433	439	443	449	457	461
463	467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593	599
601	607	613	617	619	631	641	643	647	653
659	661	673	677	683	691	701	709	719	727
733	739	743	751	757	761	769	773	787	797
809	811	821	823	827	829	839	853	857	859
863	877	881	883	887	907	911	919	929	937
941	947	953	967	971	977	983	991	997	

  - 소수를 저장하는 배열이 어디까지 사용 중인지
  - 알아야 하지 않을까?
    - Index? 또는 Pointer? 를 이용하여

  - Index 또는 Pointer는 언제 바뀌어야 하는가?

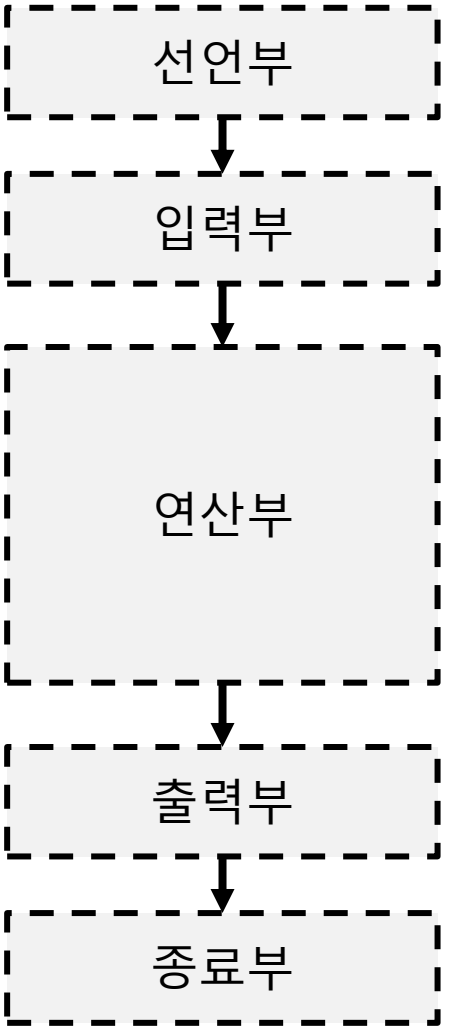
2. 제출 실습문제

n번째 소수 찾아내기 – 개략적 절차

- **입력부**
  - n을 0보다 큰 정수가 들어올 때까지 입력 받음

- **연산부**
  - 충분히 큰 정수의 배열을 먼저 하나 선언하자.
    - Int prime[1000]; 또는 Int prime[10000];
    - 본인이 생각하는 적당한 크기로
  - 어디까지 사용하고 있는지를 가리키는 변수를 선언하자
    - int* pmax 또는 int mid
  - While 문을 이용하여 소수의 배열을 차곡차곡 만들어가자
    - [2, 3, 5, 7, 9, 11, ….]
  - 기준) 현재 만들어진 소수의 배열로 무엇을 할 수 있을까?
    - 현재 가진 모든 소수로 나누어 떨어지는 수를 발견하면?
    - 현재 가진 모든 소수로 나누어 떨어지지 않으면?

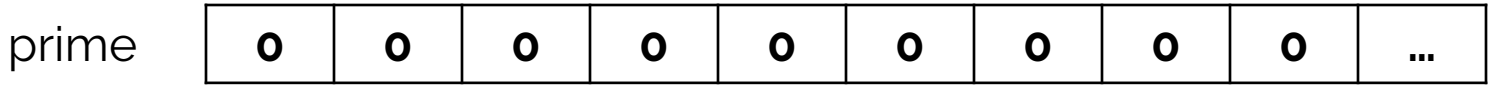
- **출력부**
  - n번째 소수 출력



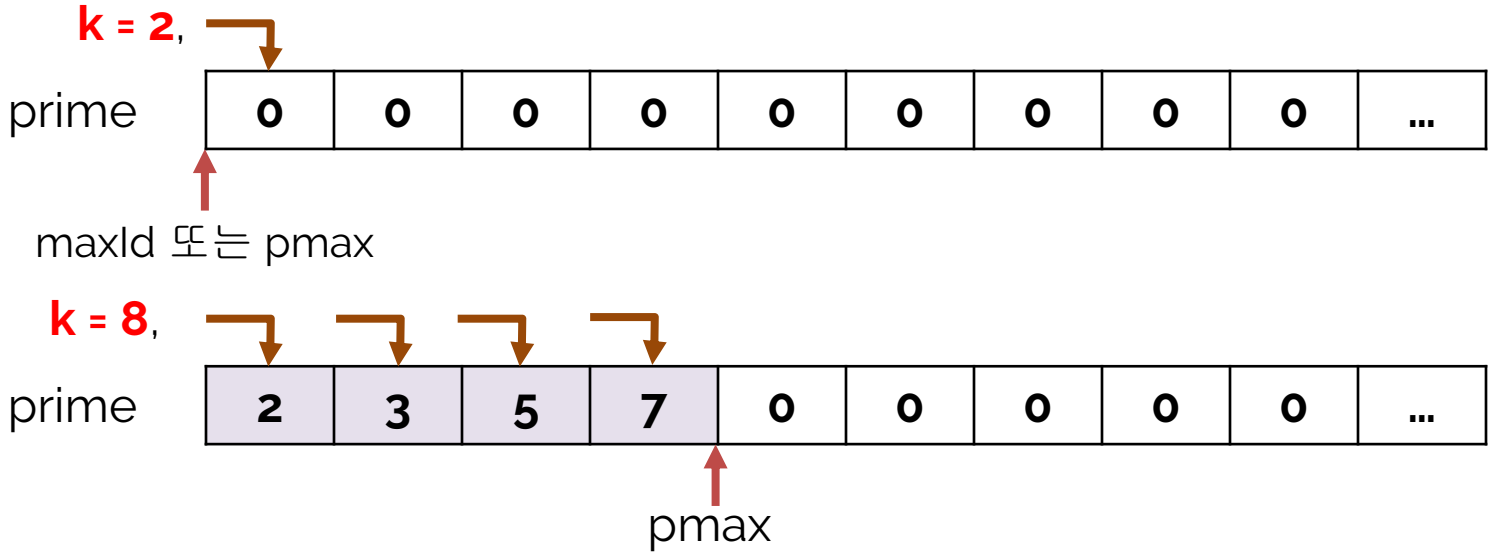
2. 제출 실습문제

n번째 소수 찾아내기 – 알고리듬 부가 설명

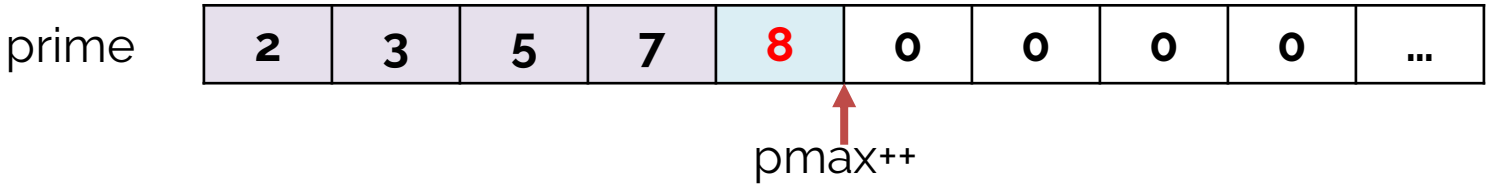
- **소수를 저장할 배열은 빈 배열로 시작**



- **입력 숫자 k를 2 ~ 무한대까지 1씩 증가해 가며**
  - 저장된 모든 pnum의 숫자로 나누어 떨어지는지 검사



- **pmax 앞의 숫자까지 모두 나누어 떨어지지 않으면 pmax를 옮기고 k를 삽입**



2. 제출 실습문제

참고 사항

- **배열 선언의 문제**
  - int prime[10000] ➔ 가능
    - 배열 크기를 1, 5, 60, 700, 8000 등의 상수로 지정하는 것은 가능
  - int k;
  - scanf("%d", &k)
  - int prime[k] ➔ 불가능
    - 배열 크기를 변수로 지정하는 것은 불가능

- **배열의 마지막 값을 가리키는 변수 활용은 어떻게?**
  - int maxId ➔ 가능
    - 배열의 index의 개념으로 정수형 변수를 사용하는 것이 가장 직관적
    - 1000번째 소수를 구하려면 maxId가 1000일 때 prime[maxId - 1]로 쉽게 찾을 수 있음
  - int* pmax ➔ 가능
    - 직관적이지 않음
    - 주소 값과 데이터형의 크기를 이용해서 k번째가 어디에 있는지 구하거나
    - 별도의 인덱스로 기록하는 것이 편함.

- **포인터를 이용한 연산도 제출시 최소 2점 가산점 부여**

▪ 실습 제출 문제

- **시작해주세요**

- **검사 받은 사람은 집에 갑시다.**
  - 먼저 하고 먼저 가자.
  - 가기 전에 제출 필수.

- **소스 코드 제출 형식**
  - Caesar Cipher – ex_6.c
  - 소수 찾기: pr_6.c

- **캡처 파일 제출 형식**
  - 확장자만 png, jpg로 하여 동일 이름으로 제출
  - 되도록 각 실습에 해당되는 캡처 하나로 높은 직관성을 갖도록 편집하여 제출
  - **추가 조건**: 여러 개의 경우 practice_5_1_0.png, practice_5_1_1.png 등으로 구분 가능하게 할 것