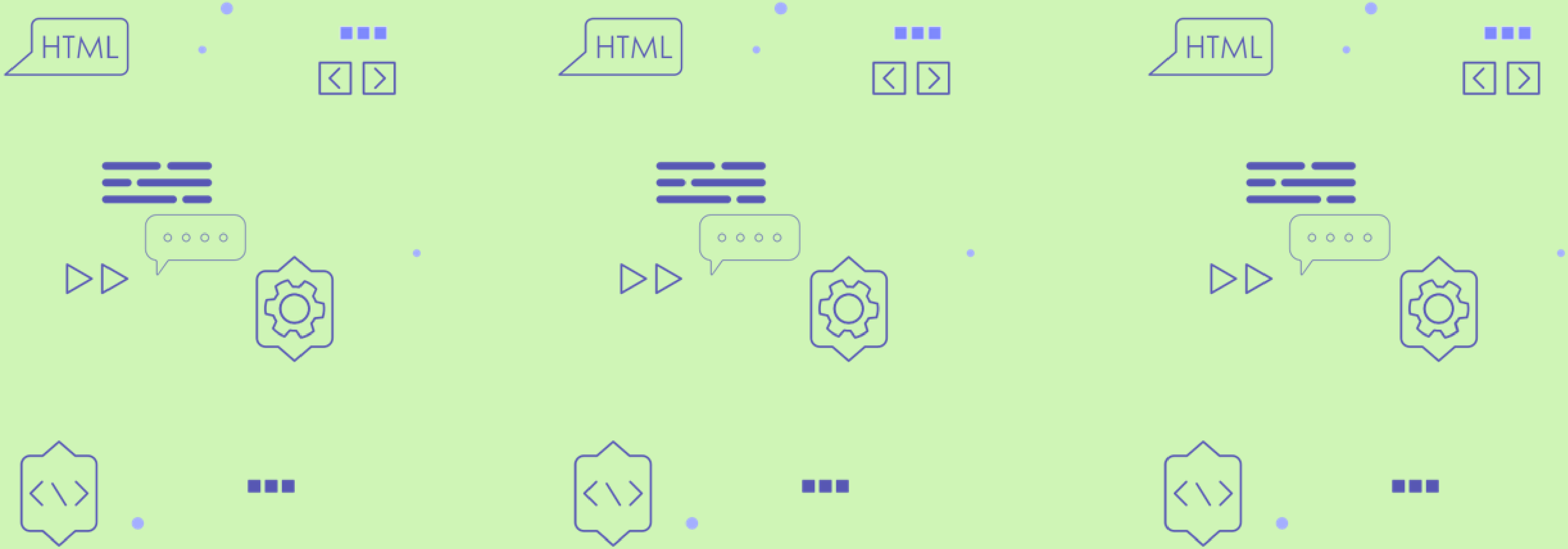


수학과 문자열

소프트웨어학과 알고리즘 동아리 SWAG 1기

아스키코드

ASCII CODE TABLE

10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자
0	0x00	NULL	22	0x16	STN	44	0x2C	,	66	0x42	B	88	0x58	X	110	0x6E	n
1	0x01	SOH	23	0x17	ETB	45	0x2D	-	67	0x43	C	89	0x59	Y	111	0x6F	o
2	0x02	STX	24	0x18	CAN	46	0x2E	.	68	0x44	D	90	0x5A	Z	112	0x70	p
3	0x03	ETX	25	0x19	EM	47	0x2F	/	69	0x45	E	91	0x5B	[	113	0x71	q
4	0x04	EOT	26	0x1A	SUB	48	0x30	0	70	0x46	F	92	0x5C	₩	114	0x72	r
5	0x05	ENQ	27	0x1B	ESC	49	0x31	1	71	0x47	G	93	0x5D	]	115	0x73	s
6	0x06	ACK	28	0x1C	FS	50	0x32	2	72	0x48	H	94	0x5E	^	116	0x74	t
7	0x07	BEL	29	0x1D	GS	51	0x33	3	73	0x49	I	95	0x5F	_	117	0x75	u
8	0x08	BS	30	0x1E	RS	52	0x34	4	74	0x4A	J	96	0x60	`	118	0x76	v
9	0x09	HT	31	0x1F	US	53	0x35	5	75	0x4B	K	97	0x61	a	119	0x77	w
10	0x0A	₩n	32	0x20	SP	54	0x36	6	76	0x4C	L	98	0x62	b	120	0x78	x
11	0x0B	VT	33	0x21	!	55	0x37	7	77	0x4D	M	99	0x63	c	121	0x79	y
12	0x0C	FF	34	0x22	"	56	0x38	8	78	0x4E	N	100	0x64	d	122	0x7A	z
13	0x0D	₩r	35	0x23	#	57	0x39	9	79	0x4F	O	101	0x65	e	123	0x7B	{
14	0x0E	SO	36	0x24	$	58	0x3A	:	80	0x50	P	102	0x66	f	124	0x7C	|
15	0x0F	SI	37	0x25	%	59	0x3B	;	81	0x51	Q	103	0x67	g	125	0x7D	}
16	0x10	DLE	38	0x26	&	60	0x3C	<	82	0x52	R	104	0x68	h	126	0x7E	~
17	0x11	DC1	39	0x27	'	61	0x3D	=	83	0x53	S	105	0x69	i	127	0x7F	DEL
18	0x12	DC2	40	0x28	(	62	0x3E	>	84	0x54	T	106	0x6A	j			
19	0x13	DC3	41	0x29	)	63	0x3F	?	85	0x55	U	107	0x6B	k			
20	0x14	DC4	42	0x2A	*	64	0x40	@	86	0x56	V	108	0x6C	l			
21	0x15	NAK	43	0x2B	+	65	0x41	A	87	0x57	W	109	0x6D	m			

컴퓨터는 문자를 읽을 수 X
숫자로 변환해서 읽어요!


문자열 다루기


string은 문자열을 담기 위한 자료형!
C언어의 char 배열을 이용한 문자열 처리보다 유용한 연산들을 제공

size() length()

substr()

append()

find()

compare()

.

.

문자열 다루기

size() length() : 문자열의 길이 반환

```
6   int main()
7   {
8       string str = "Hello";
9
10      cout << str.length() << endl;
11      cout << str.size() << endl;
12  }
```

Microsoft Visual Studio 디버그 콘솔

```
5
5
```

문자열 다루기

append() : 문자열 추가하기

```
 6   int main()
 7   {
 8       string str1 = "Hi";
 9       string str2 = "Hi";
10
11       cout << str1.append(" Hello") << endl;
12       cout << str2.append(" Hello", 1, 3) << endl;
13   }
```

Microsoft Visual Studio 디버그 콘솔

```
Hi Hello
HiHel
```

문자열 다루기

find() : 원하는 문자열 찾기

```cpp
int main()
{
    string str1 = "Hello";


    cout << str1.find("lo") << endl;
    cout << str1.find('l') << endl;
    cout << str1.find('o', 3) << endl;
    if (str1.find('k') == string::npos) {
        cout << "해당 문자열을 찾지 못했습니다." << endl;
    }
    else {
        cout << "해당 문자열을 찾았습니다." << endl;
    }
```

Microsoft Visual Studio 디버그 콘솔

```
3
2
4
해당 문자열을 찾지 못했습니다.
```

문자열 다루기

compare() : 두 문자열 비교

```cpp
int main()
{
    string str1 = "Hello";
    string str2 = "Hi";

    if (str1.compare(str2) == 0) {
        cout << "문자열 일치" << endl;
    }
    else if (str1.compare(str2) < 0) {
        cout << str1 << "가 사전편찬 순서 앞" << endl;
    }
    else {
        cout << str1 << "가 사전편찬 순서 뒤" << endl;
    }

}
```

Microsoft Visual Studio 디버그 콘솔

```
Hello가 사전편찬 순서 앞
```

문자열 다루기

substr() : 문자열 자르기

문자열.substr(시작 위치, 길이)

```
int main() {
    string str = "abcde";

    cout << str.substr(0, 1); // a
    cout << str.substr(1, 1); // b
    cout << str.substr(2, 1); // c

    cout << str.substr(0, 2); // ab
    cout << str.substr(1, 2); // bc

    return 0;
}
```

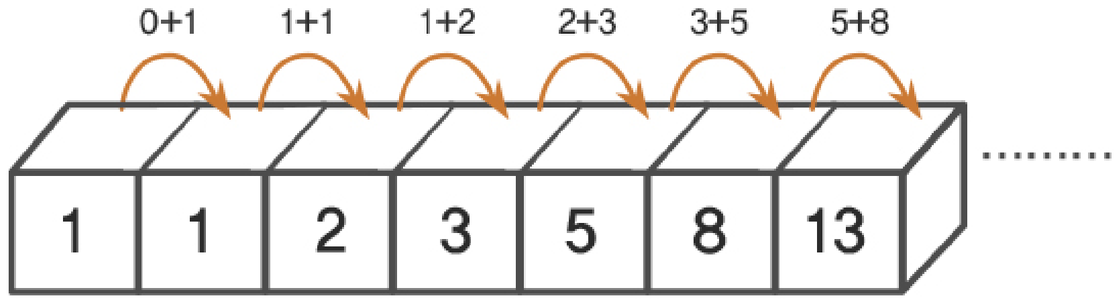
문자열.substr(시작 위치)

```
int main() {
    string str = "abcde";

    cout << str.substr(1); // bcde
    cout << str.substr(2); // cde
    cout << str.substr(3); // de

    return 0;
}
```

피보나치 수열



첫째 및 둘째 항이 1이며 그 뒤의 모든 항은 바로 앞 두 항의 합인 수열

피보나치 수열

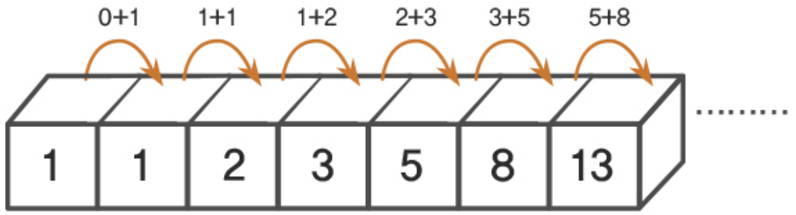
반복문 사용

```
int fibonacci(int n) {
    if (n <= 1)
        return n;

    int first = 1;
    int second = 1;
    int result;

    for (int i = 3; i <= n; ++i) {
        result = first + second;
        first = second;
        second = result;
    }

    return result;
}
```




유클리드 호제법


두 수의 최대공약수?

유클리드 호제법

$60 = 2 \times 2 \times 3 \times 5$

$48 = 2 \times 2 \times 3 \times 4$

최대공약수 : 12

유클리드 호제법

유클리드 호제법

두 수 의 최대공약수를 빠르게 구할 수 있는 알고리즘

1. 큰 수를 작은 수로 나눈 나머지를 구한다.
2. 나눴던 수와 나머지로 또 나머지 연산을 한다.
3. 이 과정을 반복하다가 나머지가 0이 되면
4. 나눴던 수가 최대공약수가 된다.

30과 24의 최대공약수

30 % 24 = 6

24 % 6 = 0

유클리드 호제법

반복문 사용

구현

```
int gcd(int a,int b){ //a>b 가정
    int n;

    while(b!=0){
        n=a%b;
        a=b;
        b=n;
    }

    return a;
}
```

소수 판별

소수 [Prime Number]

1 과 자기 자신만을 약수로 갖는 자연수

소수 판별

방법 1

" N 보다 작은 자연수들로 모두 나눠본다. "

방법 2

" √N 이하의 자연수들로 모두 나눠본다. "

1 3 9 27 81

소수 판별

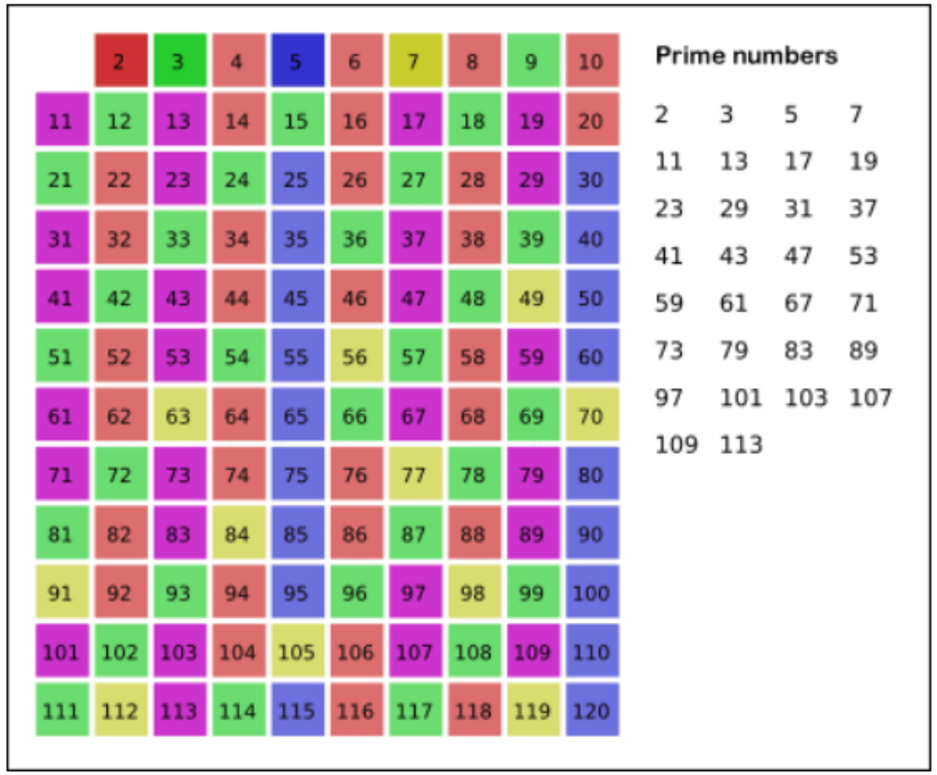
방법 3

" 에라토스테네스의 체"

1. 체크가 되어있지 않은 1보다 큰
 가장 작은 수를 찾는다.

2. 그 수의 배수를 모두 체크한다.

3. 다시 1번으로 돌아간다.



소수 판별

BAE/<JOON>
ONLINE JUDGE

1978번: 소수 찾기

첫 줄에 수의 개수 N이 주어진다. N은 100이하이다. 다음으로 N개의 수가 주어지는데 수는 1,000 이하의 자연수이다.

/<> Baekjoon Online Judge /

소수 판별

소수 찾기 구현 (1)

```
bool isPrime(int n){
    for(int i=2; i<n; i++){
        if(n%1==0){
            return false;
        }
    }
    return true;
}
```

소수 판별

소수 찾기 구현 (2)

```cpp
#include <cmath>

bool isPrime(int n){
    for(int i=2; i<=sqrt(n); i++){
        if(n%1==0){
            return false;
        }
    }
    return true;
}
```

소수 판별

“에라토스테네스의 체” 구현

```
void eratosthenes(int n){


    bool isPrime[n+1]; //n까지 수 중 소수 찾기

    //2부터 n까지의 숫자를 소수로 가정하고 초기화
    for(int i=2; i<=n; i++){
        isPrime[i] = true;
    }

    //에라토스테네스의 체
    for(int p=2; p*p<=n; p++){
        if(isPrime[p]){
            //현재 숫자가 소수인 경우, 그 배수들은 소수가 아님
            for(int i=p*p; i<=n; i+=p){
                isPrime[i] = false;
            }
        }
    }

}
```

2609 : 최대공약수와 최소공배수

1032 : 명령 프롬프트

2747 : 피보나치 수

1978 : 소수 찾기

11654 : 아스키코드

부록

재귀함수로 구현한 피보나치

```
int fibonacci(int n){
    if(n <= 2){
        return 1;
    }else{
        return fibonacci(n-1) + fibonacci(n-2);
    }
}
```

부록

재귀함수로 구현한 유클리드 호제법

```
int gcd(int a,int b){
    if(b==0){
        return a;
    }else{
        return gcd(b,a%b);
    }
}
```

끝!