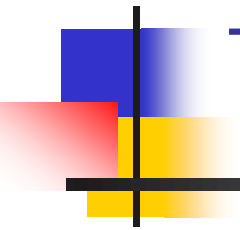


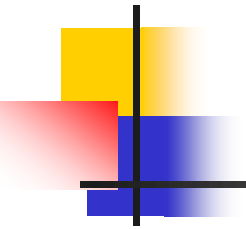
Program Patterns:
Transformation Patterns

J.W. Choi

2024

Data Processing Program Patterns

- Data Search
- Data Update
- Data Copying & Moving
- Data Transformation
- Data Reorganization
- Data Derivation

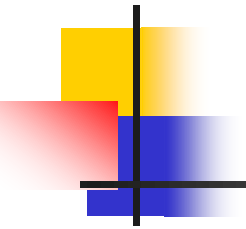


Roadmap

- Data conversion
- Data compression
- Data encryption

Data Transformation

- data conversion
  - integer to string, string to integer
  - integer/float to categorical data
  - date/time, money, measurement units
- data compression and decompression
- data encryption and decryption



Data Type Conversion

- Implicit conversion
- Explicit conversion (cast)

Implicit Conversion

```
float    avg, total;
int      num_students, num_trunc;

total = 40120;
num_students = 50;
num_trunc =  total / num_students;
```

“Char” Data Type

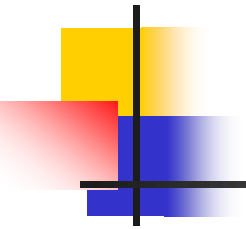
- Each English character is stored as a binary byte (ASCII or ANSI code).
  - English letters (upper case, lower case)
    - “a” (01100001), “A” (01000001),…..
  - digits (0….9)
  - special symbols (+ $ # ! @ % ^ & * ( ) + - = { } [ ] ? < >….)

ASCII Code Table (Char to Binary)

(American Standard Code for Information Interchange)

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000



Explicit Data Type Conversion

- Cast operator
- Use of library functions

Cast Operator

```
float    avg, total;
int      num_students, A_score;

total = 40120;
num_students = 50;
A_score = (int) (1.3 * total / num_students);
```

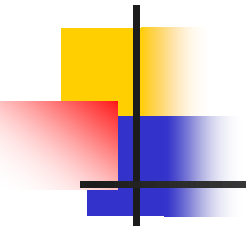
Conversion Library Functions

- #include <stdlib.h>
  - atoi (ASCII string to integer)
  - atof (ASCII string to double precision value)
  - itoa (reverse of atoi)
  - ftoa (reverse of atof)

Conversion Library Functions

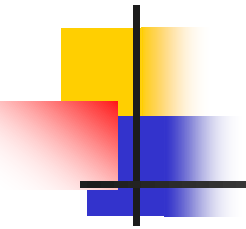
- #include <ctype.h>

함수	설명
문자 검사	
int **isalnum** (int c);	c가 *알파벳 또는 숫자*이면 0이 아닌 값을 반환한다.
int **isalpha** (int c);	c가 *알파벳*이면 0이 아닌 값을 반환한다.
int **iscntrl** (int c);	c가 *제어 문자*이면 0이 아닌 값을 반환한다.
int **isdigit** (int c);	c가 *숫자*이면 0이 아닌 값을 반환한다.
int **isgraph** (int c);	c가 *그래픽 문자*이면 0이 아닌 값을 반환한다.
int **islower** (int c);	c가 *소문자*이면 0이 아닌 값을 반환한다.
int **isprint** (int c);	c가 *출력할 수 있는 문자*이면 0이 아닌 값을 반환한다.
int **ispunct** (int c);	c가 *구두점 문자*이면 0이 아닌 값을 반환한다.
int **isspace** (int c);	c가 *공백 문자*이면 0이 아닌 값을 반환한다.
int **isupper** (int c);	c가 *대문자*이면 0이 아닌 값을 반환한다.
int **isxdigit** (int c);	c가 *16진 숫자*이면 0이 아닌 값을 반환한다.
문자 변환	
int **tolower** (int c);	c를 소문자로 변환한다.
int **toupper** (int c);	c를 대문자로 변환한다.
int **__toascii** (int c);	c를 아스키 코드로 변환한다.



Roadmap

- Data conversion
- Data compression
- Data encryption



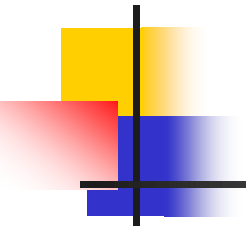
Data Compression

- Removing redundancy
  - repeated bit sequence
    - 11111111 -> (8)1
  - repeated byte (char, number) sequence
    - kkkkkkkkk -> (9)k
- Decompression
  - getting the original data back
- Lossless compression
  - getting the exact original data back
- Lossy compression
  - getting almost the original data back
  - image, video, audio data compression

Example

772-7628 772-8601 772-0113 773-3429 774-9833
773-4319 774-3920 772-0893 772-9934 773-8923
773-1134 772-4930 772-9390 774-9992 772-2314
[…]

/* compress unnecessary characters:
- spaces and first 77 */



Exercise

- Write the following C program:
  - Read file address.txt
  - Write to file compressed.txt so that its size is smaller than address.txt
- Then, write the following C program:
  - Read file compressed.txt
  - Write to file decompressed.txt, which is exactly the same as address.txt
- Assumption: every telephone number consists of 7 numbers (e.g. 123-4567)

Exercise

compression.c

```
FILE* input = fopen("address.txt", "r");
FILE* output = fopen("compressed.txt", "w");

char ch;
while ((ch = fgetc(input)) != EOF) {
    if (ch != '-' && ch != ' ') {
        fputc(ch, output);
    }
}
```

Exercise

decompression.c

```
FILE* input = fopen("compressed.txt", "r");
FILE* output = fopen("decompressed.txt", "w");

char ch;
int count = 0;
while ((ch = fgetc(input)) != EOF) {
    fputc(ch, output);
    count++;

    // Re-insert hyphen after every 3 characters (for phone numbers)
    if (count == 3) {
        fputc(' ', output);
        fputc('-', output);
        fputc(' ', output);
    }
    else if (count == 7) {
        count = 0;  // Reset count after one phone number is processed
        fputc(' ', output);
        fputc(' ', output);
        fputc(' ', output);
    }
}
```

Huffman Encoding

/* assign a short bit string for a frequent symbol */

Encoding	Symbol
1	7
010	2
011	3
00000	4
00001	5
00010	6
00011	8
00100	9
00101	0
00111	1

772 7628 -->
1 1 010 1 00010 010 00011

Image Compression Example

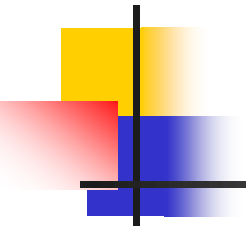
- GIF Compression: using nearest color

256 colors (24.8KB)

64 colors (17.2KB)

16 colors (9.8KB)

4 colors (3.9KB)



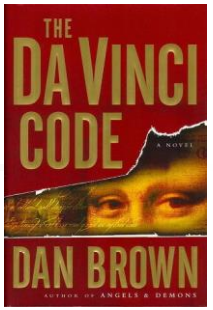
Roadmap

- Data conversion
- Data compression
- Data encryption

Data Encryption

- Transforming plaintext into ciphertext
- Need decryption key to convert ciphertext back to plaintext

- Many Uses
  - credit card information on Internet shopping
  - sensitive communication
    - military, police, government, businesses
  - personal information stored in databases
  - …



13-3-2-21-1-1-8-5
O, Draconian devil!
Oh, lame saint!
P.S. Find Robert Langdon

Anagrams

13-3-2-21-1-1- 8- 5
1-1-2- 3-5-8-13-21

O, Draconian devil!
Leonardo da Vinci!

Oh, lame saint!
The Mona Lisa!

Caesar Substitution

- A very simple encryption method
  (but easily broken)
- Encryption
  - Left or right shift each plaintext character "x" by "n" positions
  - $E_n(x) = (x+n) \bmod 26$
- Decryption (opposite of encryption)
  - Right or left shift each encrypted (ciphertext) character "x" by "n" positions
  - $D_n(x) = (x-n) \bmod 26$

Illustration

Caesar Substitution

Shift char by 3 positions ("a" -> "d")

plaintext

"hong gil dong jumped over a fence"

ciphertext

"krqj jlo grqj mxpshg ryhu d ihqfh"

Vigenere Cipher

- https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher
- The Vigenère cipher has several Caesar ciphers in sequence with different shift values.
- To encrypt, a table of alphabets can be used (called a tabula recta, Vigenère square or Vigenère table).
- It has the alphabet written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers.
- At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.

Illustration (1/4)

- the plaintext to be encrypted is

  ATTACKATDAWN

- The person sending the message chooses a keyword and repeats it until it matches the length of the plaintext, for example, the keyword "LEMON":

  LEMONLEMONLE

Vigenere Table (Square)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Illustration (2/4)

- Each row of the Vigenere table starts with a key letter. The rest of the row holds the letters A to Z (in shifted order).
- Although there are 26 key rows shown, a code will use only as many keys (different alphabets) as there are unique letters in the key string. In this example, we have just 5 keys: {L, E, M, O, N}.
- For successive letters of the message, successive letters of the key string will be taken and each message letter enciphered by using its corresponding key row.
- The next letter of the key is chosen, and that row is searched to find the column heading that matches the message character. The letter at the intersection of [key-row, msg-col] is the enciphered letter.

Illustration (3/4)

- For example, the first letter of the plaintext, A, is paired with L, the first letter of the key.
  - (* key letter: row, plaintext letter: column)
- Therefore, row L and column A of the Vigenère square are used, namely L.
- Similarly, for the second letter of the plaintext, the second letter of the key is used. The letter at row E and column T is X.
- The rest of the plaintext is enciphered in a similar fashion.

plaintext:	ATTACKATDAWN
key:	LEMONLEMONLE
ciphertext:	LXFOPVEFRNHR

Illustration (4/4)

plaintext: ATTACKATDAWN

key: LEMONLEMONLE

ciphertext: LXFOPVEFRNHR

- Decryption is performed by looking up the row in the table corresponding to the key, finding the position of the ciphertext letter in that row and then using the column's label as the plaintext.
  - (* key letter: row, ciphertext letter: column)
- For example, in row L (from LEMON), the ciphertext L appears in column A, which is the first plaintext letter.
- Next, in row E (from LEMON), the ciphertext X is located in column T. Thus T is the second plaintext letter.

Transposition

- https://en.wikipedia.org/wiki/Transposition_cipher
- Transposition cipher is a method of encryption by which the positions held by units of plaintext are shifted according to a regular system (password).
- In the ciphertext, the plaintext is reordered.
- There are many types of transition.
  - rail fence cipher, Scytale, route cipher
  - column transposition, double transposition, Myszkowski transposition
  - Disrupted transposition

Column Transposition (1/4)

- In a column transposition, the plaintext is written out in rows of a fixed length, and then read column by column,
- The columns are chosen in some scrambled order.
- Both the width of the rows and the permutation of the columns are usually defined by a keyword.
- For example, the keyword ZEBRAS is of length 6 (so the rows are of length 6), and
- the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

Column Transposition (2/4)

- In a regular columnar transposition cipher, any spare spaces are filled with nulls.
- In an irregular columna transposition cipher, the spaces are left blank.
- Finally, the message is read off in columns, in the order specified by the keyword.

Column Transposition (3/4)

- For example, suppose we use the keyword ZEBRAS and the message WE ARE DISCOVERED. FLEE AT ONCE.
- In a regular columnar transposition, we write this into the grid as follows:

6	3	2	4	1	5
W	E	A	R	E	D
I	S	C	O	V	E
R	E	D	F	L	E
E	A	T	O	N	C
E	Q	K	J	E	U

- Q K J E U are random characters that fill empty positions.

Column Transposition (4/4)

- The ciphertext is formed by reading the plaintext in the column order specified in the keyword:

  EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE

Double Transposition

- A single column transposition could be attacked by guessing possible column lengths, writing the message out in its columns (but in the wrong order, as the key is not yet known), and then looking for possible anagrams.
- Thus to make it stronger, a double transposition was often used.
- This is simply a column transposition applied twice. The same key can be used for both transpositions, or two different keys can be used.

Double Transposition

- Let us take the result of the irregular column transposition example.
- Let us perform a second encryption with a different keyword, STRIPE, which gives the permutation "564231":

5	6	4	2	3	1
E	V	L	N	A	C
D	T	E	S	E	A
R	O	F	O	D	E
E	C	W	I	R	E
E					

- The resulting ciphertext is

CAEEN SOIAE DRLEF WEDRE EVTOC

Lab - Caesar Substitution

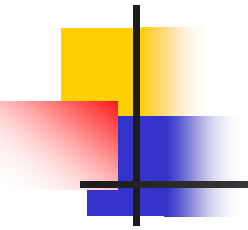
- Write the following C program:
  - Read file original.txt
  - Receive a number (encryption key) from user (using scanf)
  - Use Caesar substitution to encrypt the text, and save the ciphertext to file cypher.txt

- Write the following C program:
  - Read file cypher.txt
  - Receive a number from user (using scanf)
  - Use Caesar substitution in reverse direction to obtain the original text, and write the original text to decrypted.txt
  - Compare decrypted.txt with original.txt

Lab - Column Transposition

- Write the following C program:
  - Read file original.txt
  - Receive a password string from user
  - Use Column Transposition to encrypt the original text. Save the cypher text to cypher.txt

- Write the following C program:
  - Read file cypher.txt
  - Receive a password string from user
  - Use Column Transposition in reverse direction to obtain the original text, and write the original text to decrypted.txt
  - Compare decrypted.txt with original.txt



End of Lecture