영타변환기

김택서

1. 작품 개요 (개발 동기, 기대효과)

A. 개발 동기

가끔, 꼭 한글이 안쳐져 어려움을 겪을 때 가 있곤 했습니다. 그리고 저 뿐만이 아니라

다른 몇몇의 사람들도 한글이 입력이 되지 않아 도움을 구하는 글을 인터넷상에서 쉽게

찾아 볼 수 있었습니다. 하지만 문제의 원인은 운영체제마다, 상황마다 다르고

컴퓨터를 처음 접하는 사람에게는 해결하기 어려운 부분도 있었기에 이럴 때 문제가 완전히

해결되지는 않더라도 어느정도 불편을 감소시키고 도움이 되어 한글을 좀더 쉽게 사용할 수

있도록 하기 위해서라는 이유와 또 후에 제가 필요할지도 모르겠단 생각과 전부터

한번 만들어 보고 싶었던 프로그램이기에 영타를 한글로 변환하여 한글이 입력이 되지 않는

상황이라도 한글로 변환하여 사용 할 수 있도록 만들어 주는 프로그램을 개발하게 되었습니다.

B. 컨텐츠의 기대효과

한글이 입력이 되지 않을 경우 만약 컴맹이 아니라면 적어도 포털사이트에 해결책을 검색해

볼 것입니다. 하지만 문제의 원인은 다양한 경우 이기 때문에 한글이 입력이 되지 않는 상황에서

필요한 해결책을 찾는 것은 매우 어려울 것입니다. 웹 상에서 자바스크립트로 구현한

영타 변환기가 있기는 하지만 설상 가상으로 인터넷이 되지 않을 경우에는 이마저도 사용 할 수

없게 됩니다 하지만 이 프로그램을 사용하게 된다면 온/오프 라인에 관계없이

아무리 긴 문장이라도 오류없이 한글로 변환하여 주기 때문에 만약 한글이 입력이 되지 않는다

하더라도 이 프로그램을 사용하여 문서를 작성하는 등 꽤 많은 일을 할 수 도 있습니다.

하지만 이런 전문적인 부분 말고도 게임이나 유머와 관련된 글을 보다보면

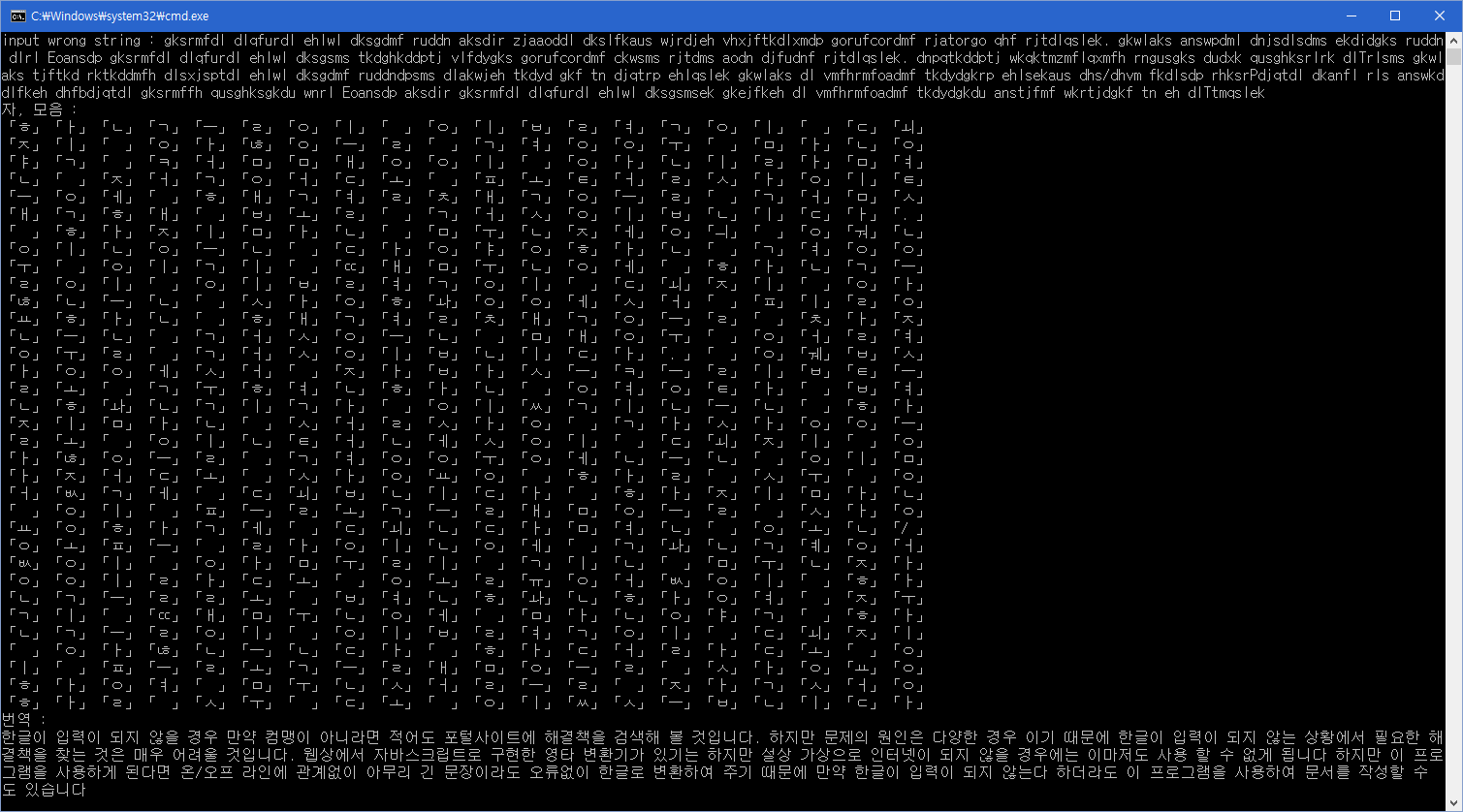
한글자판으로 쳐진 영어로된 글이 보이고 해석을 요구하는 등 일종의 암호로 쓰는것 또한

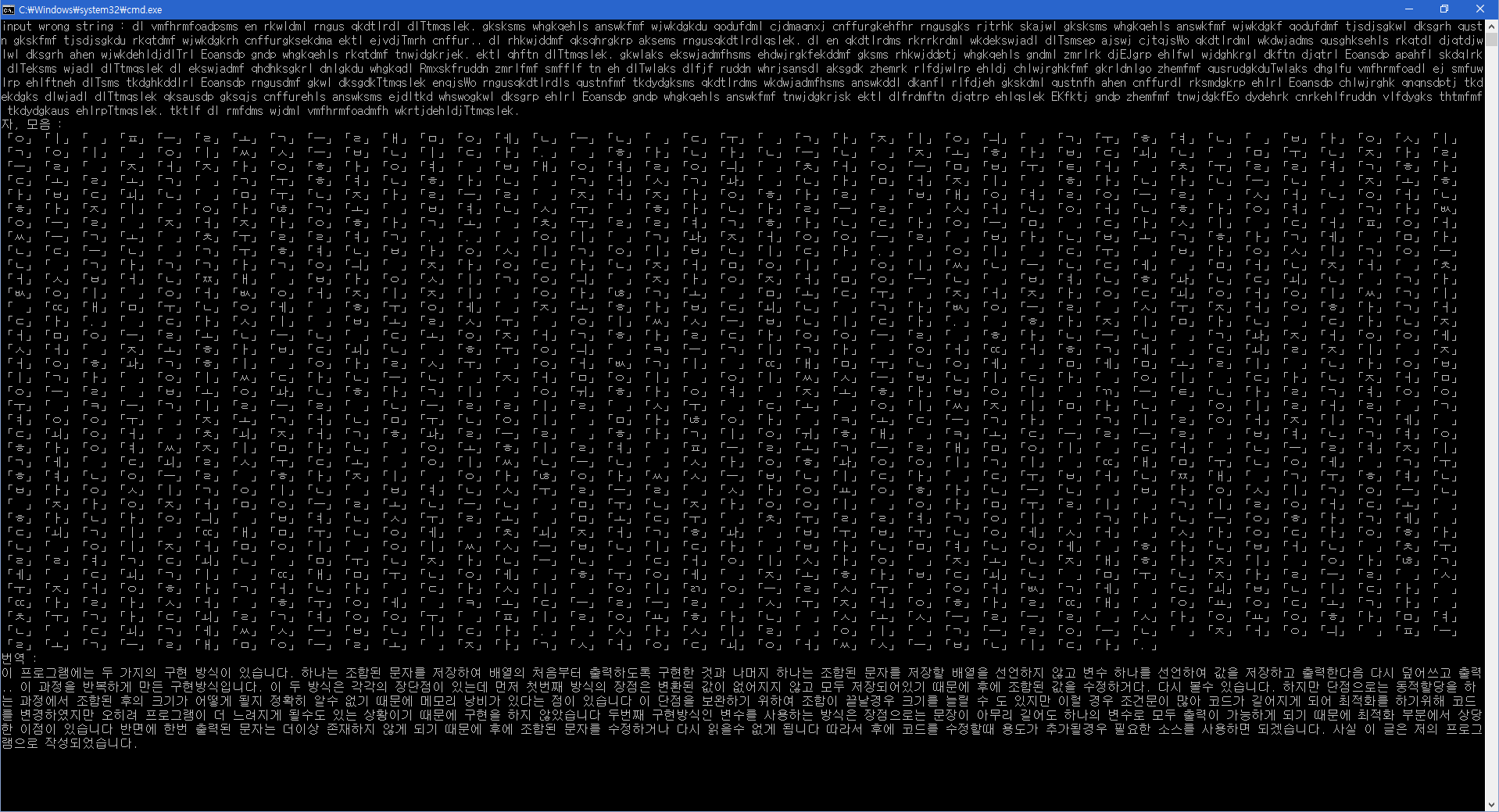
볼수 있는데 이때에서도 일일히 자판을 보지 않고서도 간편하게 한글로 변환하여 볼 수

있게됩니다. 사실 이 글의 많은 부분도 프로그램으로 번역되어 작성되었으며, 사진처럼

실제 사용을 할 수 있을 정도로 프로그램의 활용도 또한 높아 한글이 입력이 안될 경우에도

이 프로그램을 사용하면 한글을 사용 할 수 있게 됩니다.





C. 디자인

레이아웃을 나누자면 크게 세 부분으로 나눌 수 있는데 먼저 첫번째로는 영어 문자열을

입력 받는 부분입니다. 이 레이아웃에서는 잘못된 영문자열을 입력 받습니다

두번째 부분은 입력된 알파벳을 구분하여 한글의 음운으로 바꾸어 출력해 주는 부분입니다

프로그램 실행에 있어서 필수적인 부분은 아니지만 프로그램이 정상적으로 구동 되는지 와

만약 오타가 났다면 어느 부분인지 좀더 쉽게 찾아볼 수 있게 만들어 주는 부분입니다

마지막으로 핵심 부분인 출력 부분입니다. 이 부분에서는 위에서 출력된 음운들을 조합하여

출력해주는 부분으로 프로그램의 실행에 있어서 매우 필수적인 부분입니다.

-개발환경

visual studio 2015

-제작과정

|  |  |
| --- | --- |
| **날짜** | **개발과정** |
| 5/13 | 아이디어 확정 및 코드 구상 |
| 5/16 | 단순 조건문으로 음운구별 |
| 5/20 | 한글 조합구현 |
| 6/4 | 특수문자 출력지원 |
| 6/6 | 긴 문장 출력 지원 |
| 6/7 | 출력 알고리즘 최적화 |

2. 작품소개

-프로그램 내용 및 기능

이 프로그램에는 두 가지의 구현 방식이 있습니다.

하나는 조합된 문자를 저장하여 배열의 처음부터 출력하도록 구현한 것과

나머지 하나는 조합된 문자를 저장할 배열을 선언하지 않고 변수 하나를 선언하여

값을 저장하고 출력한 다음 다시 덮어쓰고 출력.. 이 과정을 반복하게 만든 구현 방식입니다.

이 두 방식은 각각의 장단점이 있는데 먼저 첫번째 방식의 장점은 변환된 값이 없어지지 않고

모두 저장되어있기 때문에 후에 조합된 값을 수정하거나 다시 볼 수 있습니다.

하지만 단점으로는 동적 할당을 하는 과정에서 조합된 후의 크기가 어떻게 될지

정확히 알 수 없기 때문에 메모리 낭비가 있다는 점이 있습니다.

이 단점을 보완하기 위하여 조합이 끝날 경우 크기를 늘릴 수 도 있지만

코드를 짜 보면 조건문이 많아 코드가 길어지게 되어 최적화를 하기 위해 코드를 변경하였지만

오히려 프로그램이 더 느려지게 될 수도 있는 상황이기 때문에 구현을 하지 않았습니다.

두 변 째 구현 방식인 변수를 사용하는 방식은 장점으로는 문장이 아무리 길어도 하나의 변수로

모두 출력이 가능하게 되기 때문에 최적화 부분에서 상당한 이점이 있습니다.

반면에 한번 출력된 문자는 더이상 존재하지 않게 되기 때문에 후에 조합된 문자를 수정하거나

다시 읽을 수 없게 됩니다.

따라서 후에 코드를 수정할때 용도가 변경되거나 기능이 추가될경우 필요한 소스를 수정하면 되겠습니다.

프로그램의 구동방식

구현 방식이 두가지가 있는 만큼 소스도 다르기 때문에 먼저 변수를 사용하여 출력하는 방식부터

설명을 하고 그 다음에 배열을 사용하는 방식과 비교하며 설명하도록 하겠습니다.

알고리즘 설명

이 프로그램의 코드를 전체적으로 설명하자면 두 방식 모두 chf라는 배열에 영문자를 입력을 받고

trans함수를 통해서 chm에 음운 형식으로 저장을 하여 조합을 한 후 반복문을 통해서 조합을

하는 원리입니다. 여기서 출력을 할 때 배열chl에 저장을 한 후 출력을 하는것과

변수에 저장을 하여 조합이 완성되면 출력을 하고 덮어 씌우는 것으로 나누어 집니다.

입력받는 부분입니다

setlocale(LC\_ALL, "Korean");

while ((tmp = getchar()) != '\n')//엔터까지 문자를 입력

{

chf = (char\*)realloc(chf, i + 1);//지속적으로 배열의 크기를 늘림

chf[i] = tmp;//배열에 입력된값 입력

i++;

}

if (i)

chf[i] = '\0';//확장된 배열의 마지막을 알려줌

chm = (wchar\_t\*)calloc(i \* 2, sizeof(wchar\_t));

setlocale로 한국어 사용을 가능하게 한다음 문자를 입력받을때마다 배열의 길이를 늘려

입력받을 문자열의 길이에 한계를 두지 안습니다.

----------------------------------

if (chf[j] == 'r'&&chf[j + 1] == 't')//ㄳ

{

for (i = 0; i < 14; i++)

{

if (ayaEN[i] == chf[j + 2])

dev += 1;

else

dev -= 1;

}

if (dev == -14)//겹밭침일때

{

chm[cnt + 1] = 65443;

j += 1, ir -= 2;

}

else//종성과 초성의 모음일때

chm[cnt + 1] = 65441;

}

겹칠 때를 구별해줍니다 (삯-삭신)

먼저 반복문으로 자음 뒤에 모음이 오는지 검사한다

모음이 온다면 dev에 1씩 더하고 아닐때는 1씩 감소시킵니다.

반복문을 모두돌고 나면 뒤에 자음이 오지 않을 경우는 dev가 -14가 되어

첫번째 조건문을 만족하여 chm에 해당하는 값을 더하고 알파벳 2개가

하나의 자음이 되었으므로 ir을 -2 시키고 값을 중복으로 변환하지 않기위해

J에 1을 더해줍니다.

----------------------------------

if (chf[j] == 'h'&&chf[j + 1] == 'k')//ㅘ

{

chm[cnt + 1] = 65480;

j += 1, ir -= 2;

}

겹모음을 구별해줍니다.

모음값이 두변 입력될 경우 해당되는 값을 입력해 주고 위처럼 알파벳 두개가 사용되었으니

J에 1을더해주고 ir에서 2를 빼줍니다

----------------------------------

if (chf[j] == 'a' || chf[j] == 'A')//a~z

chm[cnt + 1] = 65457;

일반적인 자~모음은 이렇게 처리합니다

cnt가아닌 cnt+1인 이유는 cnt는 특수문자가 아닌이상

모두 65444의 값이 들어가기 때문에 각각의 조건문에 모두 넣지 않고

조건문 전으로 넣었습니다.

----------------------------------

else//영문자 제외한 값은 그대로 전달

{

chm[cnt] = chf[j];

cnt -= 1, ir -= 1;

}

cnt += 2;

특수문자(아스키코드)인 경우 한글처럼 2Byte가 아닌 1Byte이기 때문에

cnt에 직접 넣고 cnt-1을 해주어 배열을 한칸으로 만든다음 ir-1로 전체 길이를 하나 줄입니다

cnt에 +2로 배열의 다음개체로 넘어갑니다

(한글, n+2 – 특수문자, n-1+2(+1))

----------------------------------

else

printf("「%c%c」", chm[j], chm[j + 1]);

중간에 음운을 출력하는 부분에서 한글을 출력할 때 에는 65444+값 으로 총 2Byte로 출력을 합니다

----------------------------------

if (chm[i + 1] == 65441)//ㄱ

{

if (chm[i - 1] > 65470 && chm[i - 1] <= 65491)

{

if (chm[i + 3] > 65470 && chm[i + 3] <= 65491)//초성일 경우

{

craft += 44032;

wprintf(L"%c", craft);

craft = 0;

}

else//종성일 경우

craft += 1;

}

else if (chm[i - 1] <= 127)

{

wprintf(L"%c", craft);

craft = 0;

}

else

{

craft += 44032;

wprintf(L"%c", craft);

craft = 0;

}

}

저장된 음운들을 조합하는 부분입니다. 이부분에서부터 변수 형식과 배열 형식이 나누어 지게 되는데,

첫번째 조건문 에서 전 문자가 자음인지를 구별하고 자음이라면 그 전 문자가 모음 이었는지를 확인합니다.

만약 아니라면 특수문자인지 구별하고 특수문자 일 경우 변수방식은 바로 특수문자를 출력한 후

변수를 초기화해주고 배열일 경우는 배열에 값을 넣어 준 후 배열의 다음 요소로 넘어갑니다.

특수문자도 아닐경우 전 문자가 받침이 있는 경우이므로 변수 방식은 44032를 더해주고 출력후 초기화

배열 방식일 경우는 배열의 값에 44032를 더한 후 다음 요소로 넘어갑니다.

만약 전 문자가 모음이고 그 다음 문자도 모음일 경우 받침이 없는 문자의 다음으로 온것이므로

변수일 경우 44032를 더하고 출력 후 초기화 배열일 경우 배열에 값을 넣고 다음 요소로 넘어가게됩니다

만약 그 다음 문자가 모음이 아닐경우 종성이란 의미이므로 해당 값을 더합니다

----------------------------------

else if (chm[i + 1] == 65459)//ㅃ

{

if (chm[i] < 128)

craft += 44032;

wprintf(L"%c", craft);

craft = 588 \* 8;

}

쌍자음일때 전 문자가 한글일경우 변수 형식일시 +44032로 조합을 마무리 지어주고 출력해 줍니다.

그다음 변수에 다시 쌍자음의 값으로 초기화 시켜줍니다

배열 형식일 시에는 배열의 요소에 44032를 더하고 다음 요소로 넘어가게 됩니다

----------------------------------

if (chm[i + 1] == 65471)//ㅏ

craft += 28 \* 0;

모음일 경우 해당되는 값을 변수 또는 배열에 더해줍니다.

----------------------------------

if (chm[i + 1] == 65443)//ㄳ

craft += 3;

쌍자음일 경우 해당값을 변수 또는 배열에 더해줍니다.

----------------------------------

else if (chm[i] < 128)

{

if (chm[i - 1] < 128)

{

wprintf(L"%c", craft);

craft = chm[i];

i--;

}

else

{

craft += 44032;

wprintf(L"%c", craft);

craft = chm[i];

}

}

특수문자일 경우 특수문자의 전 문자가 특수문자인지 문자인지 구별하여

특수 문자일 경우 특수 문자를 출력한 후 변수 또는 배열에 값을 넣어줍니다.

그다음 i—를 해주어 다음 음운배열의 다다음값이 아닌 다음값을 읽어들입니다.

전 문자가 문자라면 44032를 더하여 조합을 마무리 해주고 출력을 한 후 값을 변수 또는 배열에 집어넣습니다

----------------------------------

else if (chm[i] == 65444 && chm[i + 1] == 65444)//ㄴ

{

if (chm[i - 1] <= 127)

{

wprintf(L"%c", craft);

craft = 588 \* 2;

}

else

craft = 588 \* 2;

}

ㄴ의 경우 첫번째 값과 두번째 값이 65444로 일치하여 &&로 두 조건을 단후

전 맨 처음의 경우이므로 문자가 특수문자인지만 구별하여 출력합니다.

----------------------------------

else

{

if (chm[i - 1] < 128 || i == 0)//처음이거나 전문자가 특수문자일경우

{

if (i)

craft = chm[i];

wprintf(L"%c", craft);

i--;

}

else//전문자가 일반 문자일 경우

{

craft += 44032;

wprintf(L"%c", craft);

craft = chm[i];

i--;

}

}

처음부터 특수문자가 올 경우 그러니까 한글 때문에 모든 카운트는 2로 했는데 처음 시작되는 경우

기존의 구별 방식에서 전의 음운을 구별하게 되어서 컴파일러 오류가 발생하게 되어 i가 2보다 작을 때

(처음일 경우)와 2보다 클 때 를 나누었는데 처음부터 특수문자가 오거나 특수문자 특수문자가 올 경우

별도로 구별하기위해 사용되는 부분입니다. 첫번째 조건문에서는 전 문자가 특수문자이거나

(특수문자 특수문자 인 경우) i가 0일 때 (특수문자)인 경우를 포함하여 i가 0일때는 바로 출력을 하고

0이 아닐 경우에는 값을 다시 덮어 씌운 후 출력을 하게 됩니다. 전 문자가 일반 문자일 경우는 44032를

더하여 조합을 완성시키고 출력 후 다시 특수문자로 초기화 시킵니다.

후 특수문자는 1Byte이므로 i—로 음운배열에서 한칸들 당겨서 값을 읽어들입니다.

----------------------------------

if (chm[i - 1] < 128)

wprintf(L"%c\n", craft);

else if (chm[i - 2] > 128)

{

craft += 44032;

wprintf(L"%c\n", craft);

}

마지막 출력구문입니다 변수형식에만 포함되며 배열방식에서는 반복문으로 출력을 하게됩니다

마지막 글자가 특수문자 일 경우는 그대로 출력하게 되며 문자일 경우는 44032를 더하여 조합을

다 한 후 출력을 하도록 합니다

----------------------------------

느낀 점

처음 아이디어를 구상할 때에는 제 수준에 맞는 프로젝트라 생각했었습니다

지금까지 컴퓨터를 사용해 오면서 한글을 사용하는 것에 대하여 큰 어려움이 없었고 마치

한국인이 한국어를 사용하는 것처럼 자연스러워 입출력에 대해서는 크게 상관 쓰지 않았습니다.

또 굳이 관심을 주지 않아도 되는 부분이었으며 그렇기 때문에 더욱 프로그램에서 한글을

다룬다는 것이 쉬운 것처럼 보였습니다. 하지만 구체적으로 코드를 짤 생각을 해보니

도저히 감이 오지 않았습니다. 유니코드란것을 처음 공부해 보았고 그 유니코드라는 것도

C에서 어떻게 사용해야 할지도 몰라 코드를 몇 번이고 다시 짰습니다.

관련된 예제 소스를 찾아 참고하려고도 했었으나 C로 구현된 소스는 찾지도 못하고

혼자의 힘으로 소스를 짜야 한다는 부담감 또한 컸습니다.

하지만 제가 선택한 주제였고 동시에 저의 목표이기도 하였기에 포기할 수 도 없었습니다.

그렇게 프로그램을 완성하기 위해 시간이 날 때면 코드를 짜고 생각하며 필요한 기능들을

하나하나 구현해내며 몇일 동안 고민하던 문제를 해결하여 최적화까지 끝냈을 때 의 기분은 정말

더 할 나위 없이 상쾌하고 목표를 이룬 성취감은 몇 주 몇 일 동안의 긴 시간 동안 고민하며

코드들을 쓰고 지우고 했었던 그 나날들로 배가되었습니다. 이번 개인 프로젝트는 코드를 짜며

사용된 문법 중 어렵거나 새로운 문법은 많지 않았지만 단순히 한글을 표현하는 것이

처음이고 어려움이 많아서, 그리고 혼자 제작한 것이기에 더욱이 어렵게 느껴진 것 같습니다.

여러 어려움도 많은 프로젝트 였지만 그만큼 재미가 있었고 노력이란것을 알 수 있게 해준

프로젝트였습니다.