



Capitolul 5. Tastatura

Deși Windows acceptă și mouse-ul ca dispozitiv de intrare, tastatura deține încă poziția principală

Tastatura nu poate fi tratată ca un dispozitiv de intrare independent de celelalte funcții ale programului. De exemplu, programele răspund deseori la intrările de la tastatură afișând caracterele introduse în zona client a ferestrei. Astfel, manipularea intrărilor de la tastatură și afișarea textului trebuie să fie tratate împreună.

Tastatura – elemente de bază

- Programul „află” despre apăsarea unor taste prin intermediul mesajelor care ajung la procedura de fereastră.
 - atunci când utilizatorul apasă și eliberează tastele, driverul de tastatură transmite sistemului de operare informațiile legate de acțiunile asupra tastelor.
 - Windows salvează aceste acțiuni (sub formă de mesaje) în coada de așteptare a sistemului.
 - Mesajele de la tastatură sunt apoi transferate în coada de mesaje a programului căruia îi aparține fereastra ce deține „cursorul de intrare”.
 - Programul distribuie mesajele procedurii de fereastră corespunzătoare.
- Motivul acestui proces în două etape este legat de sincronizare.
- Windows trimite programelor opt tipuri de mesaje
- parte a sarcinii de manipulare a tastaturii constă în a ști ce mesaje sunt importante.

Ignorarea tastaturii

Programele nu trebuie să reacționeze la toate mesajele de la tastatură. Multe funcții ale tastaturii sunt tratate chiar de Windows. De exemplu, puteți să ignorați apăsările de taste legate de funcții sistem.

„Acceleratori” pentru opțiunile de meniu folosite mai frecvent - combinații de taste funcționale - sau alte taste corespunzătoare unor caractere - cu tasta Ctrl.

Casetele de dialog au și ele o interfață cu tastatura, dar programele, în general, nu trebuie să monitorizeze tastatura cât timp este activă o casetă de dialog.

Interfața cu tastatura este manipulată de Windows și acesta trimite programului mesaje prin care îi comunică efectele tastelor apăsate.

Cursorul de intrare

Tastatura trebuie să fie partajată de toate aplicațiile rulate simultan. Unele aplicații pot avea mai multe ferestre, iar tastatura trebuie să fie partajată de toate ferestrele din cadrul aceleiași aplicații. Atunci când este apăsată o tastă, o singură fereastră trebuie să primească mesajul privind apăsarea tastei respective. Fereastra care primește acest mesaj este fereastra care deține „cursorul de intrare” („input focus”).

Conceptul cursorului de intrare este strâns legat de conceptul de „fereastră activă”. Fereastra care deține cursorul de intrare este fie fereastra activă, fie o fereastră descendent a ferestrei active.

Ferestrele descendent (butoanele de apăsare, butoanele radio, casetele de validare, barele de derulare, casetele de editare și casetele listă) nu sunt niciodată ferestre active. Dacă o fereastră descendent deține cursorul de intrare, atunci fereastra activă este fereastra părinte.

O procedură de fereastră poate să afle când are cursorul de intrare prin interceptarea mesajelor WM_SETFOCUS și WM_KILLFOCUS. Mesajul WM_SETFOCUS indică faptul că fereastra primește cursorul de intrare (input focus), iar mesajul WM_KILLFOCUS - că fereastra pierde cursorul de intrare.

Acționări de taste și caractere

Mesajele privind tastatura fac diferența între „keystrokes” și „caractere”. Aceste noțiuni sunt legate de cele două moduri în care puteți să priviți tastatura.

În primul rând, tastatura este o colecție de taste. Tastatura are o singură tastă A; apăsarea tastei A este o acționare de tastă, iar eliberarea tastei A este tot o acționare de tastă.

Tastatura este în același timp și un dispozitiv de intrare care generează caractere afișabile. Tasta A poate să genereze mai multe caractere, în funcție de starea tastelor Ctrl, Shift și Caps Lock.

Pentru acționările de taste care generează caractere afișabile, Windows trimite programului atât mesaje pentru acționarea de taste, cât și mesaje pentru caractere.

Unele taste nu generează caractere. Astfel de taste sunt Shift, tastele funcționale, tastele de deplasare și tastele speciale, precum Insert și Delete. În cazul acestor taste, Windows generează numai mesaje pentru acționari de taste.

Mesaje pentru acționări de taste

Atunci când apăsați o tastă, Windows inserează în coada de așteptare a ferestrei care deține cursorul de intrare un mesaj WM_KEYDOWN sau un mesaj WM_SYSKEYDOWN. Atunci când eliberați fasta, Windows inserează în coada de așteptare a ferestrei un mesaj WM_KEYUP sau un mesaj WM_SYSKEYUP

	<i>Tasta a fost apăsată</i>	<i>Tasta a fost eliberată</i>
<i>Tastă obișnuită</i>	WM_KEYDOWN	WM_KEYUP
<i>Tastă de sistem</i>	WM_SYSKEYDOWN	WM_SYSKEYUP

De obicei, mesajele de apăsare și de eliberare a tastei sunt trimise în pereche. Ca toate mesajele trimise prin coada de așteptare, mesajele pentru acționări de taste conțin informații de timp. Puteți să obțineți momentul relativ în care a fost apăsată sau eliberată o tastă apelând funcția *GetMessageTime*

Taste obișnuite și taste de sistem

Particula „SYS” din mesajele WM_SYSKEYDOWN și WM_SYSKEYUP - taste apăsată în combinație cu tasta Alt. Aproape toate mesajele care afectează fereastra programului trec mai întâi prin procedura de fereastră. Windows prelucrează aceste mesaje numai dacă le retransmiteți funcției *DefWindowProc*.

Dacă în procedura de fereastră adăugați următoarele linii:

```
case WM_SYSKEYDOWN :  
case WM_SYSKEYUP :  
case WM_SYSCHAR :  
return 0 ;
```

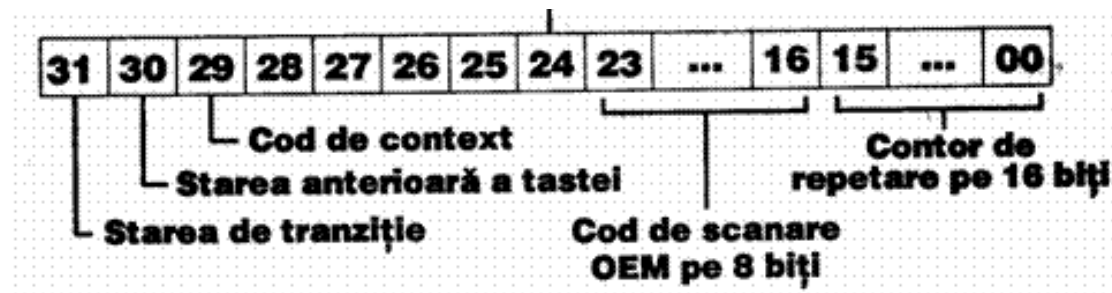
dezactivați toate operațiile legate de combinațiile Alt+tastă

Mesajele WM_KEYDOWN și WM_KEYUP sunt generate, de obicei, pentru tastele apăsată și eliberate fără tasta Alt.

Programul poate să folosească sau să ignore aceste mesaje. Sistemul de operare le ignoră.

Variabila *IParam*

- *IParam* - șase câmpuri: contorul de repetare, codul de scanare OEM, indicatorul flag pentru taste extinse, codul de context, starea anterioară a tastei și starea de tranziție



- Contorul de repetare (CR) specifică numărul de acționari de taste reprezentat de un mesaj - rata de autorepetare a tastelor depășește posibilitățile de prelucrare ale programului.

Alte câmpuri

Codul de scanare OEM

- Codul de scanare OEM (OEM Scan Code) este codul de scanare al tastaturii, generat de componentele hardware. (este identic cu cel transmis în registrul AH, în timpul întreruperii apelului BIOS 16H.)

Indicatorul flag pentru taste extinse

- Indicatorul flag pentru taste extinse (Extended Key Flag) are valoarea 1 dacă mesajul este generat de una dintre tastele suplimentare de pe tastatura IBM extinsă.

Alte câmpuri

Codul de context

- Codul de context (Context Code) are valoarea 1 dacă este apăsată tasta Alt.

Starea anterioară a tastei

- Starea anterioară a tastei (Previous Key State) are valoarea 0 dacă tasta nu a fost anterior apăsată, și valoarea 1 dacă tasta a fost apăsată.

Starea de tranziție

- Starea de tranziție (Transition State) are valoarea 0 dacă tasta este apăsată și valoarea 1 dacă tasta este eliberată.

Coduri virtuale de taste

- wParam conține codul virtual care identifică tasta apăsată sau eliberată.
- Codurile virtuale pe care le veți folosi cel mai des au nume definite în fișierele antet din Windows.

<i>Decimal</i>	<i>Hexa</i>	<i>Identificator WINDOWS.H</i>	<i>Necesar</i>	<i>Tastatură IBM</i>
1	01	VK_LBUTTON		
2	02	VK_RBUTTON		
3	03	VK_CANCEL	✓	Ctrl-Break
4	04	VK_MBUTTON		
8	08	VK_BACK	✓	Backspace
9	09	VK_TAB	✓	Tab
12	0C	VK_CLEAR	✓	Tasta numerică 5 cu tasta Num Lock inactivă
13	0D	VK_RETURN	✓	Enter
16	10	VK_SHIFT	✓	Shift
17	11	VK_CONTROL	✓	Ctrl
18	12	VK_MENU	✓	Alt

Starea tastelor de modificare

- *lParam* și *wParam* nu spun nimic programului despre starea tastelor de modificare. Puteți să obțineți starea oricărei taste virtuale folosind funcția *GetKeyState*. (Shift, Alt și Ctrl) și (Caps Lock, Num Lock și Scroll Lock). De exemplu:
GetKeyState (VK_SHIFT) ;
returnează o valoare negativă dacă tasta Shift este apăsată. Valoarea returnată de apelul:
GetKeyState (VK_CAPITAL) ;
are un 1 în bitul cel mai puțin semnificativ dacă tasta Caps Lock este activă.
- Dacă aveți nevoie de starea curentă a unei taste, puteți să folosiți funcția *GetAsyncKeyState*.

Utilizarea mesajelor de acționare a tastelor

- În general, programele pentru Windows folosesc mesajele WM_KEYDOWN pentru tastele care nu generează caractere.
- De cele mai multe ori veți prelucra mesajele WM_KEYDOWN numai pentru tastele de deplasare a cursorului. Atunci când prelucrați mesajele trimise de tastele de deplasare puteți să verificați și starea tastelor Shift și Ctrl, folosind funcția *GetKeyState*:
- Una dintre cele mai bune metode de a vă hotărî cum să folosiți tastatura este să studiați modul de utilizare a acesteia în programele Windows existente. Dacă nu vă place cum este folosită, puteți să faceți ceva diferit. Dar nu uitați că orice noutate înseamnă creșterea perioadei de care are nevoie un utilizator pentru a se obișnui cu programul dumneavoastră.

ÎMBUNĂȚIREA PROGRAMULUI SYSMETS

```
case WM_KEYDOWN :
iVscrollInc = iHscrollInc = 0 ;
    switch (wParam)
    {
        case VK_HOME:           // la fel ca WM_VSCROLL, SB_TOP
            iVscrollInc = -iVscrollPos ;
            break ;

        case VK_END :           // la fel ca WM_VSCROLL, SB_BOTTOM:
            iVscrollInc = iVscrollMax - iVscrollPos ;
            break ;

        case VK_UP :            // la fel ca WM_VSCROLL, SB_LINEUP :
            iVscrollInc = -1 ;
            break ;

        case VK_DOWN :          // la fel ca WM_VSCROLL, SB_LINEDOWN
            iVscrollInc = 1 ;
                                   break ;

        default :
            break ;
    }
if (iVscrollInc = max (-iVscrollPos, min(iVscrollInc, iVscrollMax-iVscrollPos)))
{
    iVscrollPos += iVscrollInc ;
    ScrollWindow (hwnd, 0, -cyChar * iVscrollInc, NULL, NULL);
    SetScrollPos (hwnd, SB_VERT, iVscrollPos, TRUE);
    UpdateWindow (hwnd) ;
}
if (iHscrollInc = max (-iHscrollPos,min(iHscrollInc, iHscrollMax - iHscrollPos)))
{
    iHscrollPos += iHscrollInc;
    ScrollWindow (hwnd, -cxChar*iHscrollInc, 0, NULL, NULL) ;
    SetScrollPos (hwnd, SB_HORZ, iHscrollPos, TRUE) ;
}
return 0 ;
```

Transmiterea mesajelor

Nu ar fi mai bine să transformăm mesajele WM_KEYDOWN în mesaje echivalente WM_VSCROLL și WM_HSCROLL și să facem cumva funcția *WndProc* să creadă că a primit mesajele WM_VSCROLL și WM_HSCROLL, poate chiar transmitând proceduri de fereastră aceste mesaje contrafăcute? Windows vă permite să faceți acest lucru. Funcția necesară este *SendMessage* și primește aceiași parametri ca și procedura de fereastră:

SendMessage (hwnd, message, wParam, lParam) ;

Atunci când apelați funcția *SendMessage*, Windows apelează procedura ferestrei identificată de parametrul *hwnd* și îi transmite cei patru parametri. După ce procedura de fereastră încheie prelucrarea mesajului, Windows reia execuția de la următoarea linie după apelul funcției *SendMessage*. Procedura de fereastră căreia îi trimiteți mesajul poate fi aceeași procedură de fereastră, o altă procedură de fereastră din același program sau chiar o procedură de fereastră dintr-o altă aplicație.

Iată cum putem folosi funcția *SendMessage* pentru prelucrarea mesajelor WM_KEYDOWN în programul SYSMETS:

```
case WM_KEYDOWN :
    switch (wParam)
    case VK_HOME :
        SendMessage (hwnd, WM_VSCROLL, SB_TOP, 0L);
        break;
    case VK_END:
        SendMessage (hwnd, WM_VSCROLL, SB_BOTTOM, 0L);
        break ;

    case VK_PRIOR :
        SendMessage (hwnd, WM_VSCROLL, SB_PAGEUP, 0L);
        break ;
    [alte linii de program]
```

MESAJE CARACTER

Am discutat mai devreme despre ideea transformării mesajelor generate de acționarea tastelor în mesaje caracter ținând cont de starea tastelor de modificare și v-am avertizat că acest lucru nu este suficient: trebuie să țineți seama și de configurația diferită a tastaturii de la o țară la alta. Din acest motiv, nu trebuie să încercați să transformați dumneavoastră mesajele generate de acționarea tastelor în mesaje caracter. Windows o poate face în locul dumneavoastră. Ați mai văzut această secvență de cod:

```
while (GetMessage (&msg, NULL, 0, 0))  
{  
    TranslateMessage (&msg);  
    DispatchMessage (&msg);  
}
```

Funcția *GetMessage* preia următorul mesaj din coada de așteptare și completează câmpurile structurii *msg*. Funcția *DispatchMessage* transmite mesajul procedurii de fereastră corespunzătoare. Între cele două funcții este apelată funcția *TranslateMessage*, care transformă mesajele generate de acționarea tastelor în mesaje caracter. Dacă mesajul este *WM_KEYDOWN* sau *WM_SYSKEYDOWN* și dacă tasta apăsată, în funcție de starea tastelor de modificare, generează un caracter, atunci funcția *TranslateMessage* inserează un mesaj caracter în coada de așteptare. Acesta va fi următorul mesaj pe care îl va prelua funcția *GetMessage* după mesajul generat de acționarea tastei. Există patru mesaje caracter:

MESAJE CARACTER

	<i>Caractere</i>	<i>Caractere „moarte”</i>
<i>Caractere non-sistem</i>	WM_CHAR	WM_DEADCHAR
<i>Caractere sistem</i>	WM_SYSCHAR	WM_SYSDEADCHAR

Mesajele WM_CHAR și WM_DEADCHAR sunt obținute din mesaje WM_KEYDOWN. Mesajele WM_SYSCHAR și WM_SYSDEADCHAR sunt obținute din mesaje WM_SYSKEYDOWN. În majoritatea cazurilor programul poate să ignore toate celelalte mesaje în afară de WM_CHAR. Parametrul *lParam* transmis procedurii de fereastră are același conținut ca și parametrul *lParam* al mesajului generat de acționarea tastei din care a fost obținut mesajul caracter. Parametrul *wParam* conține codul ASCII al caracterului. Mesajele caracter sunt transmise procedurii de fereastră între mesajele generate de acționarea tastelor. De exemplu, dacă tasta Caps Lock nu este activă și apăsați și eliberați tasta A, procedura de fereastră primește următoarele trei mesaje

<i>Mesaj</i>	<i>Tastă sau cod</i>
WM_KEYDOWN	Tasta virtuală A
WM_CHAR	Codul ASCII al caracterului a
WM_KEYUP	Tasta virtuală A

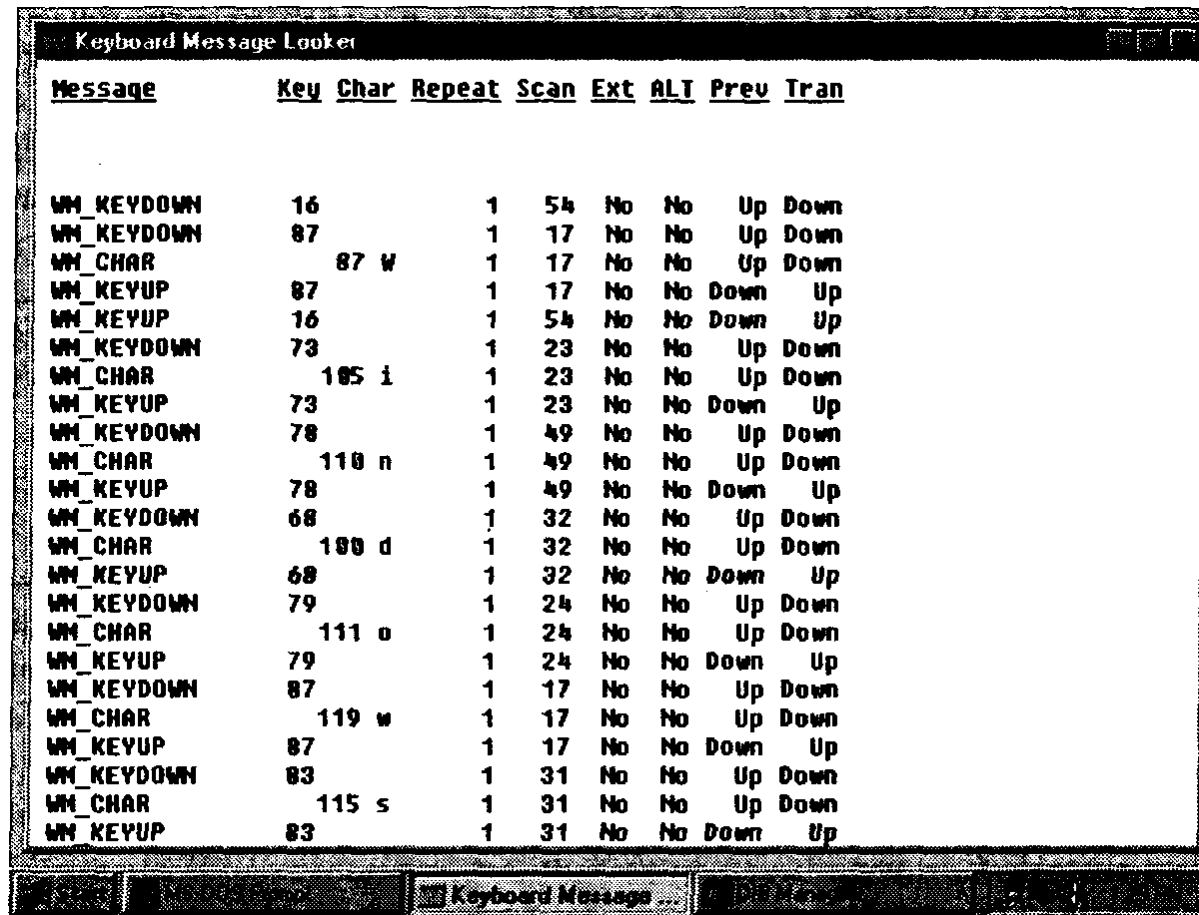
Dacă introduceți un caracter A apăsând tasta Shift, apoi tasta A, eliberând tasta A și apoi eliberând tasta Shift, procedura de fereastră primește cinci mesaje

<i>Mesaj</i>	<i>Tastă sau cod</i>
WM_KEYDOWN	Tasta virtuală VK_SHIFT
WM_KEYDOWN	Tasta virtuală A
WM_CHAR WM_KEYUP	Codul ASCII al caracterului a
WM_KEYUP	Tasta virtuală A
	Tasta virtuală VK_SHIFT

Tasta Shift nu generează un mesaj caracter. Dacă țineți tasta A apăsată până când intră în acțiune autorepetarea, veți primi un mesaj caracter pentru fiecare caracter WM_KEYDOWN

<i>Mesaj</i>	<i>Tastă sau cod</i>
WM_KEYDOWN	Tasta virtuală A
WM_CHAR	Codul ASCII al caracterului a
WM_KEYDOWN	Tasta virtuală A
WM_CHAR	Codul ASCII al caracterului a
WM_KEYDOWN	Tasta virtuală A
WM_CHAR	Codul ASCII al caracterului a
WM_KEYDOWN	Tasta virtuală A
WM_CHAR	Codul ASCII al caracterului a
WM_KEYUP	Tasta virtuală A

Examinarea mesajelor de la tastatură



The screenshot shows a window titled "Keyboard Message Looker". It contains a table with the following columns: Message, Key, Char, Repeat, Scan, Ext, ALT, Prev, and Tran. The table lists various keyboard events such as WM_KEYDOWN, WM_CHAR, and WM_KEYUP for different keys like 'W', 'I', 'n', 'd', 'o', 'w', and 's'.

<u>Message</u>	<u>Key</u>	<u>Char</u>	<u>Repeat</u>	<u>Scan</u>	<u>Ext</u>	<u>ALT</u>	<u>Prev</u>	<u>Tran</u>
WM_KEYDOWN	16		1	54	No	No	Up	Down
WM_KEYDOWN	87		1	17	No	No	Up	Down
WM_CHAR		87 W	1	17	No	No	Up	Down
WM_KEYUP	87		1	17	No	No	Down	Up
WM_KEYUP	16		1	54	No	No	Down	Up
WM_KEYDOWN	73		1	23	No	No	Up	Down
WM_CHAR		105 i	1	23	No	No	Up	Down
WM_KEYUP	73		1	23	No	No	Down	Up
WM_KEYDOWN	78		1	49	No	No	Up	Down
WM_CHAR		110 n	1	49	No	No	Up	Down
WM_KEYUP	78		1	49	No	No	Down	Up
WM_KEYDOWN	68		1	32	No	No	Up	Down
WM_CHAR		100 d	1	32	No	No	Up	Down
WM_KEYUP	68		1	32	No	No	Down	Up
WM_KEYDOWN	79		1	24	No	No	Up	Down
WM_CHAR		111 o	1	24	No	No	Up	Down
WM_KEYUP	79		1	24	No	No	Down	Up
WM_KEYDOWN	87		1	17	No	No	Up	Down
WM_CHAR		119 w	1	17	No	No	Up	Down
WM_KEYUP	87		1	17	No	No	Down	Up
WM_KEYDOWN	83		1	31	No	No	Up	Down
WM_CHAR		115 s	1	31	No	No	Up	Down
WM_KEYUP	83		1	31	No	No	Down	Up

CURSorul DE EDITARE (CARET)

Atunci când introduceți text într-un program, poziția în care va apărea următorul caracter este indicată de o liniuță de subliniere sau de un mic dreptunghi, în Windows pentru acest semn se folosește termenul „cursor de editare”.

Funcții pentru cursorul de editare

Există cinci funcții principale pentru cursorul de editare:

CreateCaret - creează un cursor de editare asociat unei ferestre.

SetCaretPos - stabilește poziția cursorului de editare în fereastră.

ShowCaret - afișează cursorul de editare.

HideCaret - maschează cursorul de editare.

DestroyCaret - distruge cursorul de editare creat.

Mai există și alte funcții pentru obținerea poziției cursorului de editare (*GetCaretPos*) și pentru stabilirea și obținerea intervalelor de licărire a acestuia (*SetCaretBlinkTime* și *GetCaretBlinkTime*).

Cursorul de editare este, de obicei, o linie ori un bloc orizontal de dimensiunea unui caracter, sau o linie verticală. Linia verticală este recomandată în cazul folosirii unui font proportional, cum ar fi fontul sistem prestabilit din Windows. Deoarece caracterele din fonturile proporționale nu au aceeași lățime, linia sau blocul orizontal nu poate avea lățimea exactă a unui caracter.

Cursorul de editare nu poate fi creat pur și simplu în timpul prelucrării mesajului WM_CREATE și nici distrus în timpul prelucrării mesajului WM_DESTROY. El este ceea ce se numește o „resursă de sistem”. Aceasta înseamnă că în sistem există un singur cursor de editare. De fapt, atunci când un program trebuie să afișeze un cursor de editare în fereastra proprie, el „împrumută” acest semn de la sistem.

Cursorul de editare

Afișarea unui cursor de editare într-o fereastră are sens numai dacă fereastra respectivă deține cursorul de intrare (input focus). Cursorul de editare indică utilizatorului faptul că poate introduce text în program. La un moment dat, o singură fereastră poate deține cursorul de intrare.

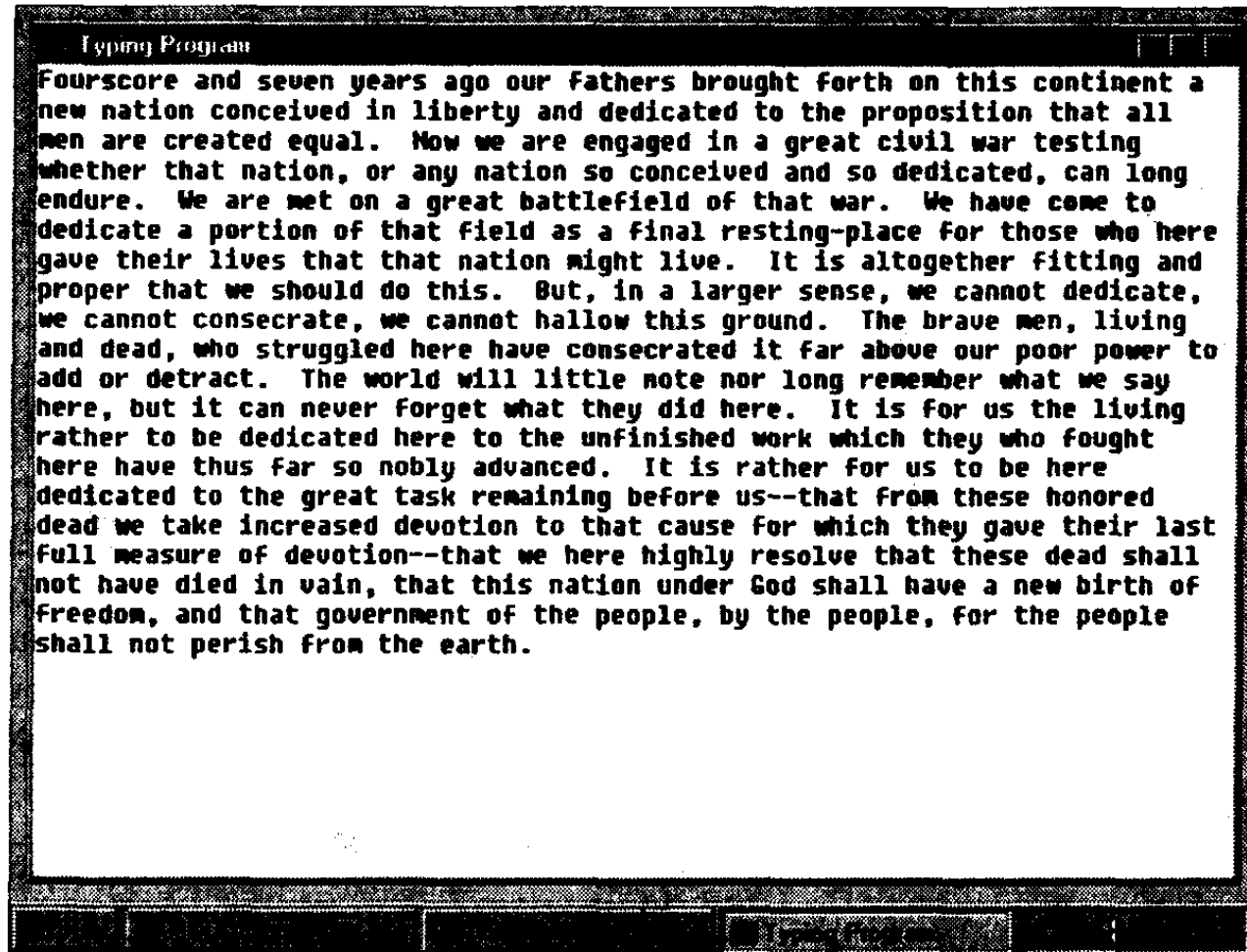
Un program poate determina dacă deține cursorul de intrare prin prelucrarea mesajelor WM_SETFOCUS și WM_KILLFOCUS. O procedură de fereastră recepționează un mesaj WM_SETFOCUS atunci când primește cursorul de intrare și un mesaj WM_KILLFOCUS atunci când pierde cursorul de intrare. Aceste mesaje sunt transmise în pereche. O procedură de fereastră primește întotdeauna un mesaj WM_SETFOCUS înainte de a primi un mesaj WM_KILLFOCUS și întotdeauna va primi un număr egal de mesaje WM_SETFOCUS și WM_KILLFOCUS până la distrugerea ferestrei.

Principala regulă de folosire a cursorului de editare este simplă. O procedură de fereastră apelează funcția *CreateCaret* în timpul prelucrării mesajului WM_SETFOCUS și funcția *DestroyCaret* în timpul prelucrării mesajului WM_KILLFOCUS.

Există și alte câteva reguli: la creare, cursorul de editare nu este afișat. După apelarea funcției *CreateCaret*, programul trebuie să apeleze funcția *ShowCaret* pentru a face vizibil cursorul de editare. În plus, procedura de fereastră trebuie să-l mascheze, apelând funcția *HideCaret*, ori de câte ori desenează ceva pe ecran în timpul prelucrării unui alt mesaj decât WM_PAINT. După terminarea operației de desenare, programul apelează funcția *ShowCaret* ca să afișeze din nou cursorul de editare.

Efectul funcției *HideCaret* este aditiv: dacă apelați funcția *HideCaret* de mai multe ori fără să apelați funcția *ShowCaret*, atunci când doriți ca acest cursor de editare să devină din nou vizibil, trebuie să apelați funcția *ShowCaret* de tot atâtea ori de câte ori ați apelat și funcția *HideCaret*.

Programul TYPER





Soluția Unicode

Dezvoltatorii de programe implicați în crearea unor aplicații pentru piața internațională au fost nevoiți să inventeze diferite soluții pentru rezolvarea deficiențelor codului ASCII, cum ar fi paginile de coduri sau seturile de caractere pe doi octeți. Este nevoie de o soluție mai bună, și aceasta este Unicode.

Unicode este un standard de codificare a caracterelor, care folosește un cod uniform pe 16 biți pentru fiecare caracter. Acest cod permite reprezentarea tuturor caracterelor, din toate limbajele scrise, care pot fi folosite în comunicațiile prin calculator, inclusiv simbolurile chinezești, japoneze și coreene. Unicode a fost dezvoltat de un consorțiu de companii din industria calculatoarelor (inclusiv cele mai mari dintre acestea).

Evident, adaptarea programelor (și a modului de gândire al programatorilor) la ideea folosirii codurilor pe 16 biți este o sarcină laborioasă, dar merită dacă în acest fel vom putea afișa pe ecran și vom putea tipări la imprimantă texte în toate limbajele scrise existente.