Projekt skorowidza

Do przechowania i porządkowania haseł użyjemy liniowej listy dynamicznej struktur postaci

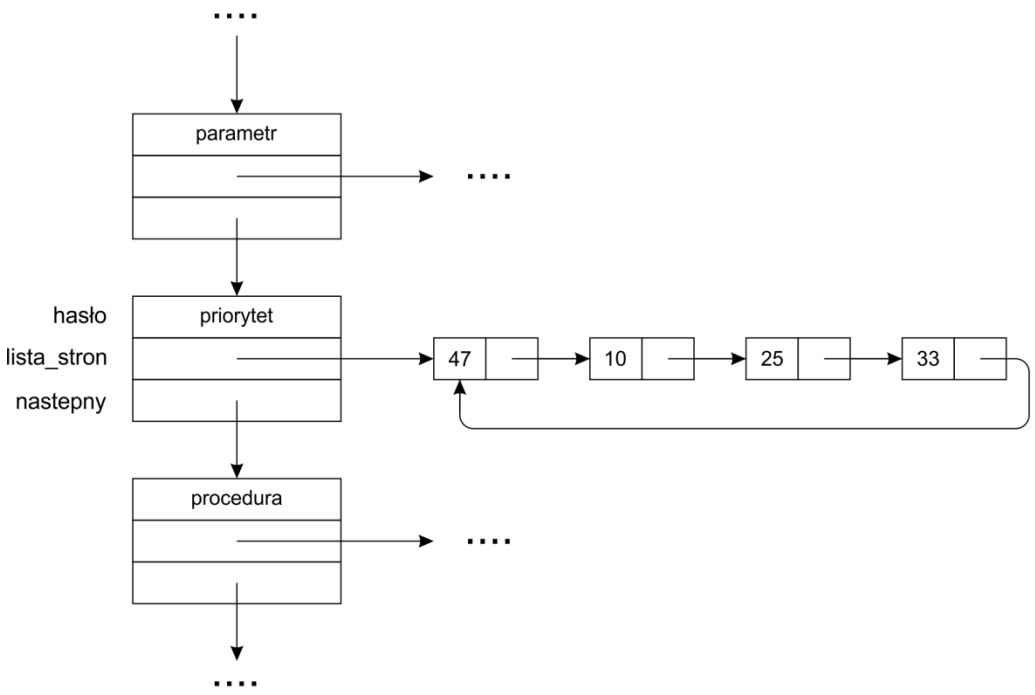
```
struct el{
    char  haslo[n];
    ellst  lista_stron;
    struct el  *nastepny;
} ell;
```

gdzie `ellst` to element listy stron

```
struct els{
    int  str;
    struct els  *nast;
} ellst;
```

Typ wskaźnika na element skorowidza `ell` nazywa się `well`, natomiast `wellst` to wskaźnik na element listy stron `ellst`.

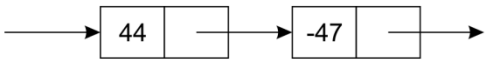
Dodawanie nowych haseł do skorowidza odbywa się jako dodawanie do prostej listy uporządkowanej alfabetycznie (można użyć wartownika z hasłem typu '`ZZZZZZZZZ`'), natomiast lista stron jest zorganizowana jako lista cykliczna, przy czym wskaźnik `lista_stron` pokazuje zawsze jej ostatni element. Nowe strony do dodania pojawiają się w porządku rosnącym, dlatego wystarczy dodawać nowe elementy na koniec tej cyklicznej listy bez konieczności jej porządkowania. Cykliczność umożliwia natychmiastowy dostęp do pierwszego elementu (jako następnika ostatniego, wskazywanego przez `lista_stron`) w chwili, gdy drukujemy dane hasło, bez dodatkowych czynności takich jak odwracanie listy lub jej przewijanie przy dodawaniu na koniec — czynności te byłyby niezbędne, gdyby lista nie była zamknięta cyklicznie.



Dodawanie do skorowidza: np. `dodaj_haslo(&S, ”procedura”, 44);`
Zakładamy obecność wartownika na końcu listy haseł. Operacje porównania łańcuchów dla uproszczenia zapisane są tu w pseudokodzie jako ”h1<h2” itp. Ostatni parametr oznacza numer strony.

```
void dodaj_haslo(well *S, string h, int st){
   well P;
   wellst Q;
   while (h > (*S)->haslo) S =  &((*S)->następny);
   if (h == (*S)->haslo) {                  //  odwołanie do hasła już znajdującego się w skorowidzu
     if (st != (*S)->lista_stron->str){     //  czy aktualna strona nie została już wcześniej zapisana
       Q = malloc(sizeof(ellst));           //  (hasło może pojawić się na tej samej stronie wielokrotnie)
       Q->str = st;
       Q->nast = (*S)->lista_stron->nast;   //  dodanie nowego elementu za ostatnim wskazywanym
       (*S)->lista_stron->nast = Q;         //  przez lista_stron
       (*S)->lista_stron = Q;               //  przesunięcie wskaźnika na nowy ostatni element
     }
   }
   else {                                   //  hasło nie pojawiło się jeszcze w skorowidzu
     P = malloc(sizeof(ell));
     P->haslo = h;                          //  tworzenie opisu nowego hasła w miejscu
     P->nastepny = *S;                      //  wskazywanym przez *S
     *S = P;
     Q = malloc(sizeof(ellst));
     Q->str = st;
     Q->nast = Q;                           //  inicjalizacja 1-elementowej cyklicznej listy stron
     P->lista_stron = Q;
   }
}
```

Drukowanie skorowidza: Ciągłe zakresy stron dla pojedynczego hasła (np. 44->45->46->47) można wykrywać podczas drukowania i odpowiednio modyfikować postać wydruku (44-47) lub też można rozpoznawać je już podczas tworzenia indeksu i zapamiętywać końcowy nr strony z takiego zakresu jako liczbę ujemną:



co później upraszcza proces drukowania, bo minus przy liczbie zastąpi znak zakresu, 44-47.
Np. drukowanie listy stron dla hasła zapamiętanego pod wskaźnikiem `*S` wyglądałoby w tej wersji następująco:

```
Q = Q1 = (*S)->lista_stron->nast;              //  pierwsza strona listy:  np.  44
printf(”%i”, Q->str);
while ((Q=Q->nast) != Q1)                      //  Q  przebiega kolejne elementy listy cyklicznej
   if (Q->str > 0) { printf(”, %i”, Q->str); } //  drukowanie kolejnej odrębnej strony:  44, 51
   else { printf(”%i”, Q->str); }              //  lub zakresu (bez odstępu):      44-47
printf(”\n”);
```

Alfabetyczne porządkowanie listy haseł odbywa się ***tylko raz***, na etapie tworzenia, na potrzeby jednorazowego odczytu przy drukowaniu, w związku z tym nie ma przekonywującego uzasadnienia dla używania bardziej skomplikowanej struktury, takiej jak np. drzewo z porządkiem, nie planujemy bowiem wielokrotnego wyszukiwania lub usuwania już zapisanych haseł.