03.03.2020

YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM2512 VERİ YAPILARI VE ALGORİTMALAR 1.ÖDEV RAPORU

AHMET SAİD SAĞLAM

17011501

KONU: Cache Buffer (Önbellek Tamponu) Tasarımı

ÇÖZÜM:

```
1   /*
2   @file
3   BLM2512 2019-2020 BAHAR ODEV-1
4   Bu programda Double Linked List yapısı kullanılarak bir Cache Buffer tasarımı yapılmıştır.
5
6   @author
7   İsim: Ahmet Said SAĞLAM
8   Öğrenci No: 17011501
9   Tarih: 03.03.2020
10  E-Mail: l1117501@std.yildiz.edu.tr
11  Compiler: TDM-GCC 4.9.2 64 bit-Release
12  IDE: DEV-C++ (version 5.11)
13  İşletim Sistemi: Windows 10 Pro 64 bit
14  */
15
16  #include <stdio.h>
17  #include <stdlib.h>
18  #include <string.h>
19  #include <conio.h>
20  #include <stdbool.h>
21  #define LENGTH 20              //maksimum string uzunluğu
```

Kodun ilk kısmında gerekli olabilecek kütüphaneler eklenmiş, daha sonra kod içerisinde gerekli bir makro atanmış ve programın geliştiricisi ve geliştirilip çalıştırıldığı ortam hakkında bilgiler verilmiştir.

```
23  //double linkli liste yapı tanımı
24  typedef struct node {
25      struct node* next;
26      struct node* prev;
27      int count;
28      char word[LENGTH];
29  } node;
```

Öncelikle problemi çözmek için ödev dokümantasyonunda belirtildiği üzere double linkli liste oluşturmak amacıyla “node” isimli bir yapı oluşturulmuştur.

Bu yapıda bir önceki ve bir sonraki node’ların adresini işaret etmek amacıyla next ve prev isimli pointerlar, her node içinde saklanması gereken stringleri tutması amacıyla word isimli karakter dizisi ve bu stringin kaç defa kullanıcı veya dosyadan alındığını tutması amacıyla count isimli değişken tanımlanmıştır.

```
31  // double linkli liste basa eleman ekleme fonksiyonu
32  node* basaEkle(node* head, char val[LENGTH]) {
33      node* newnode;
34      newnode = (node*) malloc(sizeof(node));
35      strcpy(newnode->word, val);
36      newnode->count = 1;
37      newnode->next = head;
38      newnode->prev = NULL;
39      if(head != NULL){
40          head->prev = newnode;
41      }
42      head = newnode;
43      return head;
44  }
```

basaEkle fonksiyonu çağırıldığında yeni bir node oluşturur ve bu node'u linkli listenin head node'u olarak ayarlar. Ayrıca oluşturduğu node'a kullanıcıdan veya dosyadan alınan stringi vererek count değerini "1" olarak ayarlar. Böylece sonradan aynı string girildiğinde count değeri bir arttırılır ve gerekli kontroller yapılabilir hale gelir.

```
46  //linkli listenin son düğümünü silen fonksiyon
47  void deleteTail (node* head) {
48      node* current = head;
49      node* temp;
50      while (current->next != NULL){
51          current = current->next;
52      }
53      temp = current;
54      current = current->prev;
55      current->next = NULL;
56      free(temp);
57  }
```

deleteTail fonksiyonu gerektiğinde çağırılarak double linkli listenin son node'una ilerler ve son node'u siler. Silme işlemi sebebiyle kalan son node'un next değerini NULL olarak ayarlar böylece listenin son node'u yeniden oluşmuş olur.

```
59  //linkli listedeki tüm nodeları silen fonksiyon
60  void deleteAll(node*head) {
61      node* current = head;
62      node* temp;
63      while(current->next != NULL){
64          temp = current;
65          current = current->next;
66          current->prev = NULL;
67          temp->next = NULL;
68          free(temp);
69      }
70      free(current);
71  }
```

deleteAll fonksiyonu çağırılarak listedeki tüm node'lar silinebilir. Kullanıcı Cache Buffer'ı temizlemek istediğinde arka planda bu fonksiyon çalışmış olur.

```
73  //linkli listeyi sırayla yazdıran fonksiyon
74  void printList (node* head) {
75      node* current = head;
76      printf("\n");
77      while (current->next != NULL) {
78          printf("%s,%d  <->  ", current->word, current->count);
79          current = current->next;
80      }
81      printf("%s,%d",current->word, current->count);
82  }
```

printList fonksiyonu head node'dan başlayarak son node'a doğru tüm node'ları içerdiği string ve count değerleriyle birlikte sırayla yazdırır.

```
84  //kullanıcadan alınan string halihazırda mevcut linkli listede bulunuyor mu kontrol eden fonksiyon
85  node* arama(node* head, char val[]){
86      int bulgu = 1;
87      node* temp = head;
88      while (bulgu != 0 && temp->next != NULL){
89          bulgu = strcmp(temp->word, val);
90          temp = temp->next;
91      }
92      if (bulgu == 0){
93          temp = temp->prev;
94          return temp; // bulunmus demektir adresi disariya dondurulur
95      }
96      else{
97          if (strcmp(temp->word, val) == 0){
98              return temp; // son eleman aranan ise onun adresi donulur disari
99          }
100         return NULL; // eger liste bitmis ve bulunamamissa NULL donulur
101     }
102 }
```

Kullanıcı programın akışı esnasında string değerlerini girdiği sırada bu arama fonksiyonu arka planda sürekli çalışıyor vaziyette bulunur. Eğer kullanıcı daha önce girilenlerden farklı bir string girişi yaptıysa arama fonksiyonu dışarıya NULL döndürür. Aksi taktirde kullanıcı zaten Cache Buffer'da mevcut olan bir string girişi yaptıysa arama fonksiyonu var olan o stringe ait node'un adresini dışarıya döndürür.

```
104  //eşik değerini geçen ara node'u head node yapan fonksiyon
105  node* swap(node* head, node* temp){
106      temp->next->prev = temp->prev;
107      temp->prev->next = temp->next;
108      head->prev = temp;
109      temp->next = head;
110      temp->prev = NULL;
111      head = temp;
112      return head;
113  }
114
115  //eşik değerini geçen son node'u head node yapan fonksiyon
116  node* swapTail(node* head, node* temp) {
117      temp->prev->next = NULL;
118      head->prev = temp;
119      temp->next = head;
120      temp->prev = NULL;
121      head = temp;
122      return head;
123  }
```

swap ve swapTail fonksiyonları programın akışı esnasında eğer bir node'un count değeri eşik değerini aşarsa çağırılırlar ve o node'u yeni head node olarak ayarlayarak diğer node'ların bağlantılarını da bozmadan düzenlerler.

swap fonksiyonu ara bir node'un count değeri eşiği aşması durumunda çağırılır ve o ara node'u head node olarak ayarlar. Ayrıca aradan çekilen node yüzünden oluşabilecek sıkıntıları önlemek amacıyla node'un bir önceki ve bir sonraki node'larını birbirlerine double linkli liste yapısını bozmayacak şekilde bağlar.

swapTail fonksiyonun swap fonksiyonundan farkı ara bir node yerine son node'un head node yapılmak istendiği durumda çağırılacak olmasıdır ve bağlantıları da son node'a göre ayarlar.

```
125  //kullanıcadan alınan string daha önceden mevcut ise bu fonksiyon çağırılır.
126  //fonksiyon count değerlerini günceller ve gerekirse ilgili node'u head node yapmaya yarayan fonksiyonu çağırır.
127  node* controlFunc(node* head, node* temp, int T) {
128      if(temp->prev == NULL) {
129          temp->count++;
130          return head;
131      }
132      else if(temp->next != NULL) {
133          temp->count++;
134          if(temp->count > T){
135              head = swap(head, temp);
136          }
137          return head;
138      }
139      else if(temp->next == NULL) {
140          temp->count++;
141          if(temp->count > T) {
142              head = swapTail(head, temp);
143          }
144          return head;
145      }
146  }
```

controlFunc fonksiyonu programda kullanıcının girdiği node zaten mevcut ise çağırılır. İlgili node, head node ise sadece count değeri arttırılır ve fonksiyondan çıkılır. Eğer head node değilse ve ara bir node ise yine count değeri arttırılır ve bu count değeri eşik değerini geçiyor mu diye kontrol edilir. Eğer geçiyorsa swap fonksiyonu çağırılır ve fonksiyondan çıkılır.

Son olarak eğer ilgili node ne head node ne de ara node ise son node demektir ve count değeri bir arttırılıp eşik değerini geçiyor mu diye kontrol edilir, geçiyorsa swapTail fonksiyonu çağırılır ve fonksiyondan çıkılır.

```
149 int main() {
150     FILE *fp;                   //file pointer
151     int clearAll = 0;           //program sonlanırken cache'i temizlemek veya olduğu gibi bırakmak durumunu kontrol eden değişken
152     int T, L;                   //esik değeri ve buffer kapasitesi
153     int tur = 0;                //verinin dosyadan mı yoksa kullanıcıdan elle mi alınacığını kontrol eden değişken
154     int Lsayaci = 1;            //buffer kapasitesini kontrol eden değişken
155     node* head = NULL;          //ilk adımda head NULL olarak belirlenir
156     node* temp;                 //geçici node
157     char fileName[50];          //kullanıcının gireceği dosya ismini tutan karakter dizisi
158     char temiz[LENGTH];         //ekranı temizlemeye yarayan karakter dizisi
159     char clean[LENGTH] = "cls"; //ekranı temizlemeye yarayan kontrolü sağlayan karakter dizisi
160     char stop[LENGTH] = "esc";  //programı sonlandırmaya yarayan kontrolü sağlayan karakter dizisi
161     char string[LENGTH];        //kullanıcının girdiği stringleri tutan karakter dizisi
162     printf("Cache Buffer'in esik degerini (T) giriniz.\n");
163     scanf("%d",&T);
164     printf("Cache Buffer'in kapasitesini (L) giriniz.\n");
165     scanf("%d",&L);
166     printf("Stringi elle girmek icin 1'e, dosyadan okumak icin 2'ye basiniz.\n");
167     scanf("%d",&tur);
```

main fonksiyonunun içinde öncelikle kullanılacak değişkenlerin, dizilerin ve pointerların tanımı yapılır. Ardından kullanıcıdan cache buffer'ın eşik değeri ve kapasitesi alınır. Daha sonra kullanıcıya stringleri elle mi girileceği yoksa dosyadan mı alınacağı sorulur. Kullanıcının seçimine göre program akışı devam eder.

```c
if(tur == 1) {
    printf("String giriniz.\n");
    scanf("%s",string);
    head = basaEkle(head, string);
    printList(head);
    printf("\nString giriniz.\nCikmak icin esc yazip onaylayiniz!\nCache'i temizlemek icin cls yazip onaylayiniz!\n");
    scanf("%s",string);
    while (strcmp(string, stop) != 0) {
        strcpy(temiz, string);
        if(strcmp(temiz, clean) != 0) {
            temp = arama(head, string);
            if(temp == NULL){
                head = basaEkle(head, string);
                Lsayaci++;
                if(Lsayaci > L){
                    deleteTail(head);
                }
            }
            else{
                head = controlFunc(head, temp, T);
            }
            printList(head);
        }
        else {
            printf("Buffer Temizleniyor...\n");
            deleteAll(head);
            head = NULL;
            Lsayaci = 1;
            printf("\nString giriniz.\n");
            scanf("%s",string);
            head = basaEkle(head, string);
            printList(head);
        }
        printf("\n\nString Giriniz.\nCikmak icin esc yazip onaylayiniz!\nCache'i temizlemek icin cls yazip onaylayiniz!\n");
        scanf("%s",string);
    }
}
```

Kullanıcı değerleri elle girmeyi seçtiğinde program ilk olarak bir string ister ve bunu head node olarak ayarlar. Daha sonra bir while döngüsü içinde kullanıcının programı sonlandırmak isteyip istemediği kontrol edilir. Kullanıcı programı sonlandırmak için “esc” yazıp onaylamalıdır ve bunun bilgisi kendisine program tarafından verilir. Hemen ardından kullanıcının buffer’ı temizlemek isteyip istemediği kontrol edilir. Kullanıcı buffer’ı temizlemek için “cls” yazıp onaylamalıdır ve yine bunun bilgisi program tarafından kullanıcıya verilir.

Program kullanıcı sonlandırmadığı sürece eşik değeri ve buffer kapasitesi kontrollerini yaparak ve gerekli koşullar oluştuğunda node ekleyip çıkararak veya olan node’ların sırasını düzenleyerek ve nihayetinde bunları ekrana yazdırarak çalışmaya devam eder.

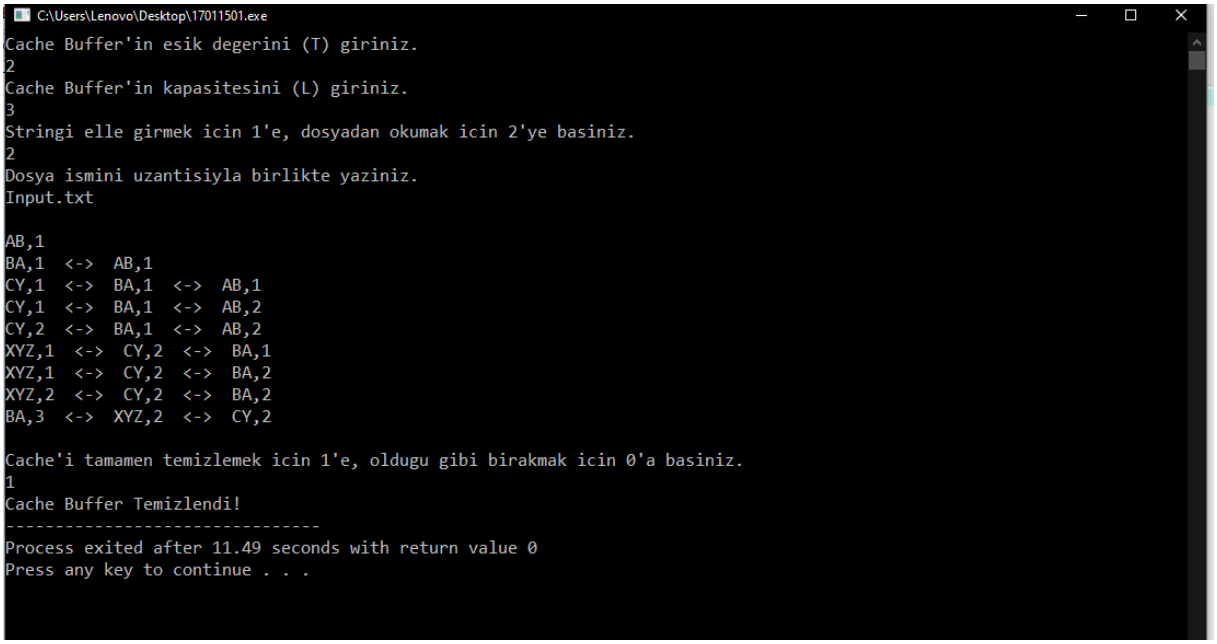
```
205     else{
206         printf("Dosya ismini uzantisiyla birlikte yaziniz.\n");
207         scanf("%s",fileName);
208         if((fp = fopen(fileName,"r")) == NULL) {
209             printf("Dosya acilamadi!\n");
210             return 0;
211         }
212         else {
213             fscanf(fp,"%s",string);
214             head = basaEkle(head, string);
215             printList(head);
216             while(!feof(fp)) {
217                 fscanf(fp,"%s",string);
218                 temp = arama(head, string);
219                 if(temp == NULL) {
220                     head = basaEkle(head, string);
221                     Lsayaci++;
222                     if(Lsayaci > L) {
223                         deleteTail(head);
224                     }
225                 }
226                 else {
227                     head = controlFunc(head, temp, T);
228                 }
229                 printList(head);
230             }
231             fclose(fp);
232         }
233     }
```

Kullanıcı verileri dosya aracılığıyla almayı seçtiğinde program kullanıcıya dosya ismini sorar ve eğer mevcutsa o dosyayı açar. Eğer dosya mevcut değilse kullanıcıya uyarı verilir. Dosya açıldığında içerisindeki stringler satır satır okunarak aynı kontroller ve gerekli işlemlerden geçirilerek buffer ekrana adım adım yazdırılır.

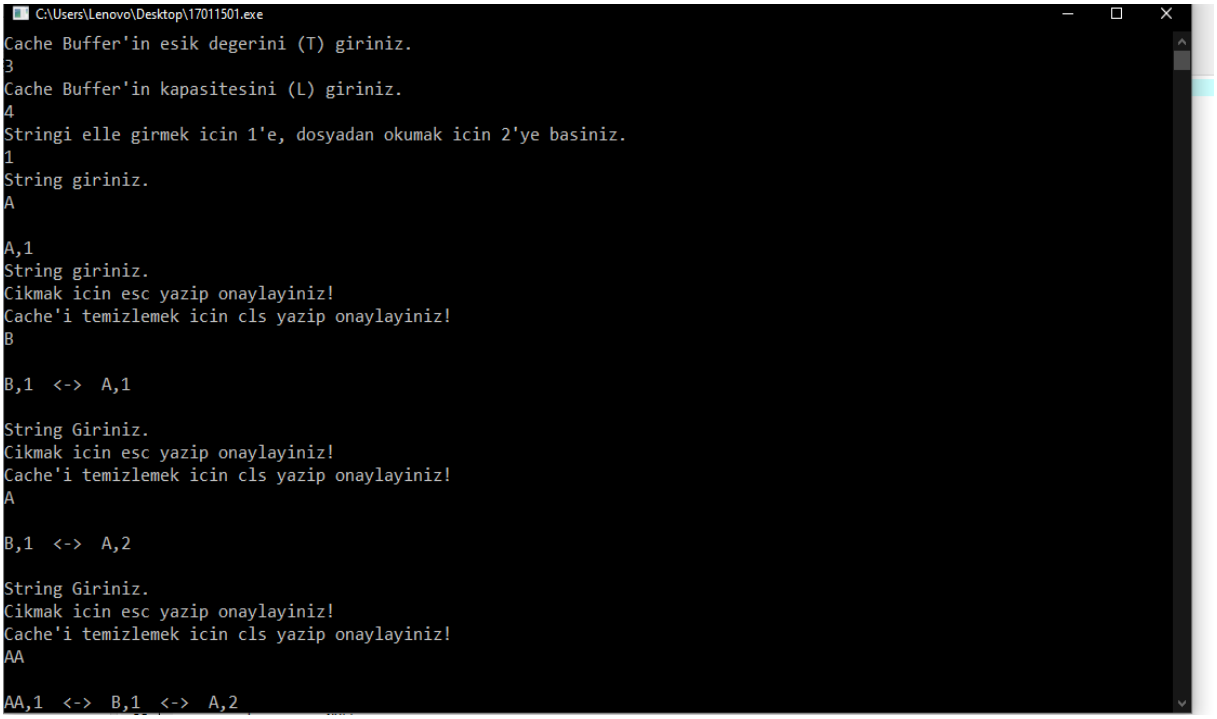
```
234     printf("\n\nCache'i tamamen temizlemek icin 1'e, oldugu gibi birakmak icin 0'a basiniz.\n");
235     scanf("%d",&clearAll);
236     if(clearAll == 1){
237         deleteAll(head);
238         printf("Cache Buffer Temizlendi!");
239     }
240     else {
241         printf("\nCache temizlenmeden cikiliyor.\nCache:\n");
242         printList(head);
243     }
244     return 0;
245 }
```

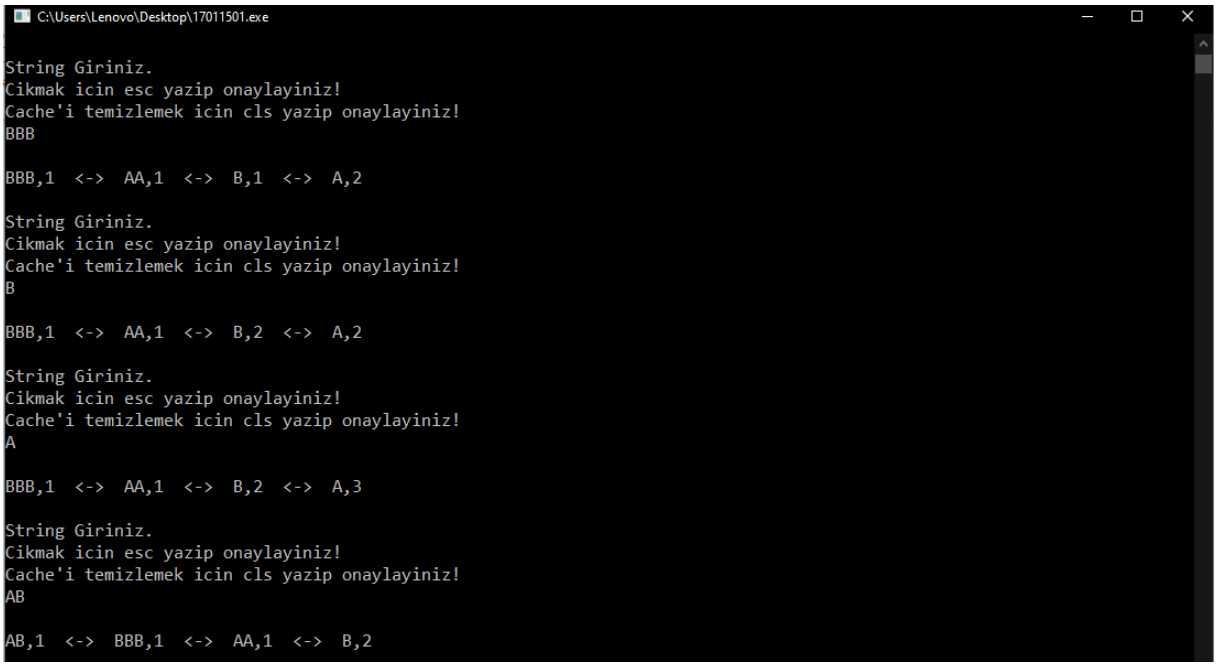
En son adımda ise program kullanıcıya cache buffer'ı tamamen silip temizlemek isteyip istemediğini sorar ve aldığı cevaba göre buffer'ı temizler veya olduğu gibi bırakır.

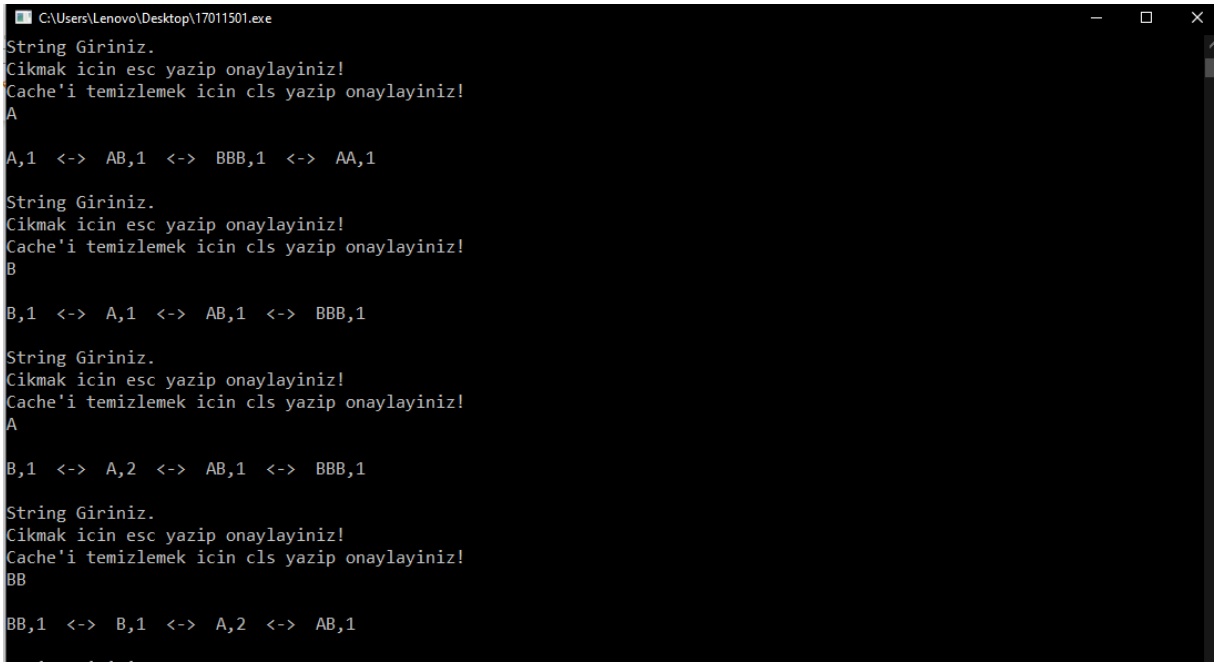
PROGRAM ÇIKTILARI

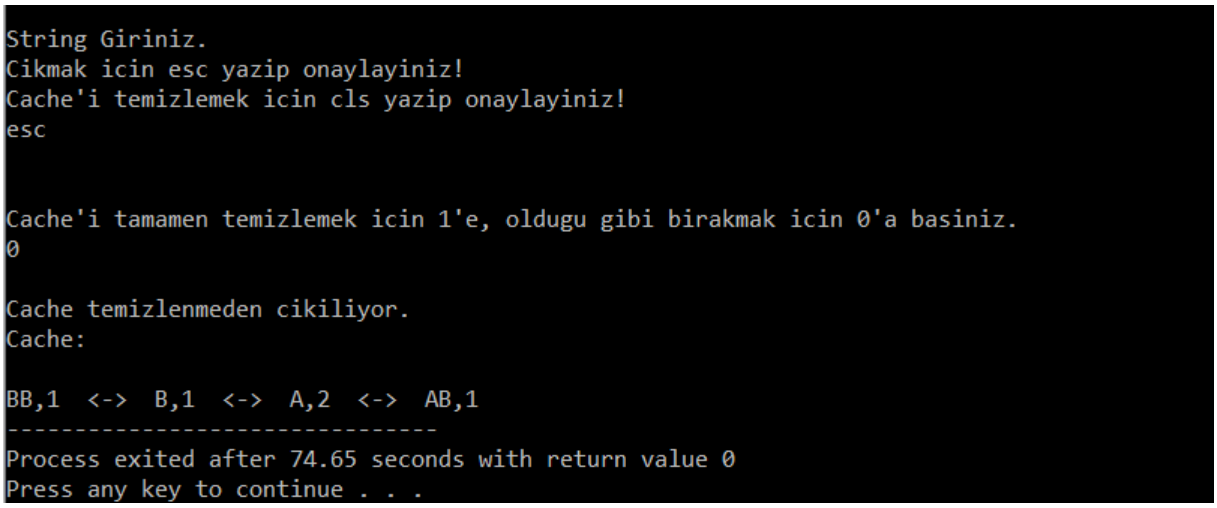


Verinin Dosyadan Alınması ve Buffer'ın Temizlenmesi




C:\Users\Lenovo\Desktop\17011501.exe
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
BBB
BBB,1 <-> AA,1 <-> B,1 <-> A,2
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
B
BBB,1 <-> AA,1 <-> B,2 <-> A,2
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
A
BBB,1 <-> AA,1 <-> B,2 <-> A,3
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
AB
AB,1 <-> BBB,1 <-> AA,1 <-> B,2


C:\Users\Lenovo\Desktop\17011501.exe
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
A
A,1 <-> AB,1 <-> BBB,1 <-> AA,1
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
B
B,1 <-> A,1 <-> AB,1 <-> BBB,1
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
A
B,1 <-> A,2 <-> AB,1 <-> BBB,1
String Giriniz.
Cikmak icin esc yazip onaylayiniz!
Cache'i temizlemek icin cls yazip onaylayiniz!
BB
BB,1 <-> B,1 <-> A,2 <-> AB,1



Verinin Kullanıcıdan Alınması ve Buffer'ın Temizlenmeden Bırakılması

SOURCE CODE

```
1.  /*
2.  @file
3.  BLM2512 2019-2020 BAHAR ODEV-1
4.  Bu programda Double Linked List yapısı kullanılarak bir Cache Buffer tasarımı yapılm
    ıştır.
5.
6.  @author
7.  İsim: Ahmet Said SAĞLAM
8.  Öğrenci No: 17011501
9.  Tarih: 03.03.2020
10. E-Mail: l1117501@std.yildiz.edu.tr
11. Compiler: TDM-GCC 4.9.2 64 bit-Release
12. IDE: DEV-C++ (version 5.11)
13. İşletim Sistemi: Windows 10 Pro 64 bit
14. */
15.
16. #include <stdio.h>
17. #include <stdlib.h>
18. #include <string.h>
19. #include <conio.h>
20. #include <stdbool.h>
21. #define LENGTH 20               //maksimum string uzunluğu
22.
23. //double linkli liste yapı tanımı
24. typedef struct node {
25.     struct node* next;
26.     struct node* prev;
27.     int count;
28.     char word[LENGTH];
29. } node;
30.
31. // double linkli liste basa eleman ekleme fonksiyonu
32. node* basaEkle(node* head, char val[LENGTH]) {
33.     node* newnode;
34.     newnode = (node*) malloc(sizeof(node));
35.     strcpy(newnode->word, val);
36.     newnode->count = 1;
37.     newnode->next = head;
38.     newnode->prev = NULL;
39.     if(head != NULL){
40.         head->prev = newnode;
41.     }
42.     head = newnode;
43.     return head;
44. }
45.
46. //linkli listenin son düğümünü silen fonksiyon
47. void deleteTail (node* head) {
48.     node* current = head;
49.     node* temp;
50.     while (current->next != NULL){
51.         current = current->next;
52.     }
53.     temp = current;
54.     current = current->prev;
55.     current->next = NULL;
56.     free(temp);
57. }
58.
59. //linkli listedeki tüm nodeları silen fonksiyon
60. void deleteAll(node*head) {
61.     node* current = head;
62.     node* temp;
```

```
63.     while(current->next != NULL){
64.         temp = current;
65.         current = current->next;
66.         current->prev = NULL;
67.         temp->next = NULL;
68.         free(temp);
69.     }
70.     free(current);
71. }
72. 
73. //linkli listeyi sırayla yazdıran fonksiyon
74. void printList (node* head) {
75.     node* current = head;
76.     printf("\n");
77.     while (current->next != NULL) {
78.         printf("%s,%d  <->  ", current->word, current->count);
79.         current = current->next;
80.     }
81.     printf("%s,%d",current->word, current->count);
82. }
83. 
84. //kullanıcadan alınan string halihazırda mevcut linkli listede bulunuyor mu kontrol
    eden fonksiyon
85. node* arama(node* head, char val[]){
86.     int bulgu = 1;
87.     node* temp = head;
88.     while (bulgu != 0 && temp->next != NULL){
89.         bulgu = strcmp(temp->word, val);
90.         temp = temp->next;
91.     }
92.     if (bulgu == 0){
93.         temp = temp->prev;
94.         return temp; // bulunmus demektir adresi disariya dondurulur
95.     }
96.     else{
97.         if (strcmp(temp->word, val) == 0){
98.             return temp; // son eleman aranan ise onun adresi donulur disari
99.         }
100.                return NULL; // eger liste bitmis ve bulunamamissa NULL donulur
101.            }
102.        }
103. 
104.        //eşik değerini geçen ara node'u head node yapan fonksiyon
105.        node* swap(node* head, node* temp){
106.            temp->next->prev = temp->prev;
107.            temp->prev->next = temp->next;
108.            head->prev = temp;
109.            temp->next = head;
110.            temp->prev = NULL;
111.            head = temp;
112.            return head;
113.        }
114. 
115.        //eşik değerini geçen son node'u head node yapan fonksiyon
116.        node* swapTail(node* head, node* temp) {
117.            temp->prev->next = NULL;
118.            head->prev = temp;
119.            temp->next = head;
120.            temp->prev = NULL;
121.            head = temp;
122.            return head;
123.        }
124. 
125.        //kullanıcadan alınan string daha önceden mevcut ise bu fonksiyon çağırılır.
```

```
126.       //fonksiyon count değerlerini günceller ve gerekirse ilgili node'u head node
   yapmaya yarayan fonksiyonu çağırır.
127.       node* controlFunc(node* head, node* temp, int T) {
128.           if(temp->prev == NULL) {
129.               temp->count++;
130.               return head;
131.           }
132.           else if(temp->next != NULL) {
133.               temp->count++;
134.               if(temp->count > T){
135.                   head = swap(head, temp);
136.               }
137.               return head;
138.           }
139.           else if(temp->next == NULL) {
140.               temp->count++;
141.               if(temp->count > T) {
142.                   head = swapTail(head, temp);
143.               }
144.               return head;
145.           }
146.       }
147.
148.
149.       int main() {
150.           FILE *fp;                    //file pointer
151.           int clearAll = 0;            //program sonlanırken cache'i temizlemek veya
    olduğu gibi bırakmak durumunu kontrol eden değişken
152.           int T, L;                    //esik değeri ve buffer kapasitesi
153.           int tur = 0;                 //verinin dosyadan mı yoksa kullanıcıdan elle
    mi alınacığını kontrol eden değişken
154.           int Lsayaci = 1;             //buffer kapasitesini kontrol eden değişken
155.           node* head = NULL;           //ilk adımda head NULL olarak belirlenir
156.           node* temp;                  //geçici node
157.           char fileName[50];           //kullanıcının gireceği dosya ismini tutan ka
   rakter dizisi
158.           char temiz[LENGTH];          //ekranı temizlemeye yarayan karakter dizisi

159.           char clean[LENGTH] = "cls"; //ekranı temizlemeye yarayan kontrolü sağlaya
   n karakter dizisi
160.           char stop[LENGTH] = "esc";  //programı sonlandırmaya yarayan kontrolü sağ
   layan karakter dizisi
161.           char string[LENGTH];         //kullanıcının girdiği stringleri tutan karak
   ter dizisi
162.           printf("Cache Buffer'in esik degerini (T) giriniz.\n");
163.           scanf("%d",&T);
164.           printf("Cache Buffer'in kapasitesini (L) giriniz.\n");
165.           scanf("%d",&L);
166.           printf("Stringi elle girmek icin 1'e, dosyadan okumak icin 2'ye basiniz.\
   n");
167.           scanf("%d",&tur);
168.           if(tur == 1) {
169.               printf("String giriniz.\n");
170.               scanf("%s",string);
171.               head = basaEkle(head, string);
172.               printList(head);
173.               printf("\nString giriniz.\nCikmak icin esc yazip onaylayiniz!\nCache'
   i temizlemek icin cls yazip onaylayiniz!\n");
174.               scanf("%s",string);
175.               while (strcmp(string, stop) != 0) {
176.                   strcpy(temiz, string);
177.                   if(strcmp(temiz, clean) != 0) {
178.                       temp = arama(head, string);
179.                       if(temp == NULL){
180.                           head = basaEkle(head, string);
181.                           Lsayaci++;
```

```
                            if(Lsayaci > L){
                                deleteTail(head);
                            }
                        }
                        else{
                            head = controlFunc(head, temp, T);
                        }
                        printList(head);
                    }
                    else {
                        printf("Buffer Temizleniyor...\n");
                        deleteAll(head);
                        head = NULL;
                        Lsayaci = 1;
                        printf("\nString giriniz.\n");
                        scanf("%s",string);
                        head = basaEkle(head, string);
                        printList(head);
                    }
                    printf("\n\nString Giriniz.\nCikmak icin esc yazip onaylayiniz!\n
    Cache'i temizlemek icin cls yazip onaylayiniz!\n");
                    scanf("%s",string);
                }
            }
            else{
                printf("Dosya ismini uzantisiyla birlikte yaziniz.\n");
                scanf("%s",fileName);
                if((fp = fopen(fileName,"r")) == NULL) {
                    printf("Dosya acilamadi!\n");
                    return 0;
                }
                else {
                    fscanf(fp,"%s",string);
                    head = basaEkle(head, string);
                    printList(head);
                    while(!feof(fp)) {
                        fscanf(fp,"%s",string);
                        temp = arama(head, string);
                        if(temp == NULL) {
                            head = basaEkle(head, string);
                            Lsayaci++;
                            if(Lsayaci > L) {
                                deleteTail(head);
                            }
                        }
                        else {
                            head = controlFunc(head, temp, T);
                        }
                        printList(head);
                    }
                    fclose(fp);
                }
            }
            printf("\n\nCache'i tamamen temizlemek icin 1'e, oldugu gibi birakmak ici
    n 0'a basiniz.\n");
            scanf("%d",&clearAll);
            if(clearAll == 1){
                deleteAll(head);
                printf("Cache Buffer Temizlendi!");
            }
            else {
                printf("\nCache temizlenmeden cikiliyor.\nCache:\n");
                printList(head);
            }
            return 0;
        }
```