YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM2031 YAPISAL PROGRAMLAMA DERSİ

DÖNEM PROJESİ RAPORU

Hazırlayan: Mehmet Hayri Çakır - 16011023

Sunulan/Dersin Yürütücüsü: Dr.Öğr.Üyesi Zeyneb Kurt

İstanbul, 2019

İçindekiler

1.1. Projenin Amacı

Projenin amacı, sözde komutlar içeren bir metin dosyasını okuyarak, C kaynak kodunu oluşturan ve derleyip çalıştıran bir program yazmaktır. (Source-to-Source Compiler yazımı)

1.2. Üretilen Çözüm Yöntemleri

Geliştirme aşamasında en büyük problem, değişkenlerin _degistentipi yazılmasıydı. Değişkenlerin programa nasıl tanıtılacağı konusunda yaptığım araştırmalar sonucunda strtok fonksiyonunun ve sembol tablosunun tam aradığım şey olduğunu fark ettim ve bütün programı strtok fonksiyonu üzerine, değişkenlerin implementasyonunu ise, değişkenlerin adını, değişkenlerin tipini, degiskenadi_degiskentipi halini, pointer olup olmadığını, dizi olup olmadığını, hangi fonksiyon içinde kullanıldığını bir bağlı listede tuttum. Reserved word'ler için ise ayrı bir veri yapısı kurup, sözde komutları ve C kodu karşılığını burada tuttum. Karşılaştığım diğer bir problem ise fonksiyon başlarındaki değişken deklarasyonlarını nasıl yapacağımı bilmememdi. Bunun için öncelikle uses_vars adında matris oluşturup değişken tiplerinin kullanılma durumuna göre TRUE veya FALSE olarak ayarladım. Matris olma sebebiyse, her fonksiyon için ayrı tutulması. Bağlı liste için yazdığım insert fonksiyonunda bir kontrol daha ekleyerek, sonraki elemanın tipi, o an yazılan değişken tipine eşit oluncaya kadar sonraki elemana geçmesini sağladım. Böylece sembol tablosunda int'ler arka arkaya, char'lar arka arkaya… şeklinde oldu. Syntax'ı diğer operatörlerden farklı olan for için ayrı bir fonksiyon yazdım. Aynı değişken iki kere tanımlanmaması için isDeclared isimli fonksiyonu yazdım.

1.3. Fonksiyonlar ve Kısa Özetleri

Bu projede toplam 21 fonksiyon kullanılmıştır. Bunlar

- int main()
- FILE *open_file(char *,char *)
- DATABASE * arrange_bools(char *,char *, char *,char *)
- DATABASE *create_list_element(unsigned int, unsigned int, unsigned int)
- char *strtok_2(char*,const char*)
- int find_file_size(FILE *)
- int for_loop_exception(FILE *, char *)
- void isDeclared(char*)
- void insert(DATABASE*)
- void arrange_tokens()
- void print_file(char *,char *)
- void put_tabs_new_lines(FILE *, int, int, int, int)
- void isLoop(FILE *, int)
- void isNot_Loop(FILE *, int)
- void isOpen_Bracelet(FILE *, int, int, int)
- void isClose_Bracelet(FILE *, int, int, int)
- void isArray(FILE *,char *)
- void print_Vars(FILE *,char *,int)
- void isFunc(FILE *,char *)
- void GetVal_PrintLine(FILE *, char *, int)
- void return_type(FILE *,char *,long int)

1.3.1. int main()

Programın ana fonksiyonudur. Pseudo kod dosyasının, yazılacak c kodu dosyasının ve output dosyasının isimleri burada alınır. Diğer yardımcı fonksiyonların bazıları buradan çağrılır.

1.3.2. FILE *open_file(char *,char *)

Parametre olarak, dosya ismi ve hangi modda açılacağını alır ve açar. Açılamazsa ekrana hata verir.

1.3.3. DATABASE * arrange_bools(char *,char *, char *,char *)

isDeclared fonksiyonun içinden çağrılır. Parametre olarak değişken adı, değişken tipi, değişkenadı_değişkentipi ve ifarray (dizi olup olmadığı) alır. Aldığı ilk üç parametrelerin uzunluğunu parametre kullanarak create_list_element fonksiyonunu çağırır. Ekstra yer ayrılmasını engellemek amaçlanmıştır. Yer ayrıldıktan sonra sembol tablosunda ptr -> data_type alanına, parametre olarak alınan değişken tipi yazılır. İfarray TRUE ise ptr -> isArray TRUE yapılır. ptr değeri döndürülür.

1.3.4. DATABASE *create_list_element(unsigned int, unsigned int, unsigned int)

Parametre olarak değişken adı, değişken tipi, değişkentipi_değişkenadı (var_i) uzunluklarını alır. Bu uzunluklara göre ilgili char * fieldlara malloc ile yer açılır. Bağlı liste olduğu için ptr -> db_next fieldı NULL yapılır. Yer açma başarısız olursa hata mesajı yazılır. ptr değeri döndürür.

1.3.5. char *strtok_2(char*,const char*)

String.h kütüphanesinde bulunan strtok fonksiyonuyla aynı görevi yapar. İlk parametresi bir string, 2. parametresiyse delimiter'lardır. String'i, bu delimiterlara göre parçalara böler. İki ayrı strtok fonksiyonuna ihtiyacım olduğu için bunu kullandım. Kendim yazmadım, internetten edindim. İlgili kaynak, belgenin sonundaki kullanılan kaynaklar ve bağlantılar kısmında yer almaktadır.

1.3.6. int find_file_size(FILE *)

Okunan Pseudo kod dosyasının pointer'ını parametre alır. fseek ve ftell stdio kütüphane fonksiyonlarını kullanarak dosyada kaç karakter olduğunu bulur. Dosya boyutunu döndürür. Sonrasında oluşturulan data_array stringine mallocla yer ayırmak için kullanılacaktır.

1.3.7. int for_loop_exception(FILE *, char *)

For operatörünün syntax'ı için yazılmıştır. Strtok fonksiyonunu kullanarak for döngüsünün case kısmını yazdırır. i değerini döndürür. Sonrasında bu i değeri put_tabs_new_lines fonksiyonuna parametre olarak gönderilecektir. Bu satırın sonuna noktalı virgül eklenmemesi için bu işlem yapılmaktadır.

1.3.8. void isDeclared(char*)

Parametre olarak data_array stringini alır. main fonksiyon içinde data_array içine Pseudo kod dosyası kopyalanmıştı. Delimiter'lar kullanılarak data_array, token'lara bölünür. Değişken daha önce deklare edilmediyse, tipi elde edilince switch((int)*dtype) yapısıyla değişken tipine göre arrange_bools fonksiyonu çağrılır ve uses_vars matrisinin ilgili alanı TRUE olarak ayarlanır. Değişken bir diziyse, ptr -> isarray kısmına array_size atanır. array_size default olarak 100'dür fakat token=DEFINE ise array_size ona göre ayarlanır. Bu fonksiyonda pointer olma durumu da kontrol edilir. Hangi fonksiyonda geçtiği ayarlanır. Kısacası ilk aşamada çalışan bu fonksiyon sembol tablosuyla ilgili bütün durumları ayarlar. Sonrasında insert fonksiyonunu çağırır.

1.3.9. void insert(DATABASE*)

create_list_element fonksiyonu ile oluşturulan ve field'ları doldurulmuş olan structure'ın pointer'ını parametre olarak alır. null bir for statement ile istenilen değişken tipine gelene veya liste bitene kadar dönülür. İstenilen durum oluşunca insert edilir. Farklı değişken tiplerinin sıralı olarak yerleştirilmesiyle ilgili ekran görüntüsü aşağıdadır.

PTR	PROC	TOKEN	IDENT.	DTYPE	ISARR?	ISPTR?	NEXT
10658168	MAIN	sayi_ip	sayi	int	0	1	10657856
10657856	MAIN	sayi1_i	sayi1	int	0	0	10665432
10665432	MAIN	str2_c	str2	char	20	0	10665128
10665128	MAIN	str_c	str	char	100	0	10664888
10664888	MAIN	ch_c	ch	char	0	0	0

1.3.10. void arrange_tokens()

Bağlı liste elemanlarının, identifier ve data_type kısımlarının birleştirilerek Pseudo haline geri dönüştürülmesi ve token kısmına yazılmasını sağlar. Eğer pointer ise sonuna p eklenir. Bunu yaparken head=NULL'dan başlayarak for döngüsü içinde gerçekleştirir.

1.3.11. void print_file(char *,char *)

Sembol tablosunu oluşturduk ve sıra geldi C kodu halini dosyaya yazmaya. 2 dosya da açılır ve Pseudo kod dosyasından okunarak C halini 2. Dosyaya yazar. İndentasyonun düzgün olması için tab_counter, open_bracelet ve close_bracelet değişkenleri kullanılır. EOL değişkeniyse Pseudo kod dosyasından fgetsle alınan komutun sonuna gelinmesi durumunda TRUE olur. İlk aşamada kontroller yapılır, hata yoksa void main() ve { karakteri dosyaya yazılır. sonrasında while döngüsü, yeni fonksiyon gelene kadar veya dosya sonuna gelene kadar dönerek çeşitli durumları kontrol ederek Pseudo kodu C koduna çevirir. Bu kısımda başta yazılan kod yetersiz kaldığı ve istisnaları olduğu için bu durumları aşmak adına 4 yerde goto kullanılmıştır. Bu fonksiyonda çeşitli kontrol değişkenleri mevcuttur. Bunlardan bazıları; iffunc(yeni fonksiyon mu?),iffunccall(Yoksa fonksiyon mu çağrılmış?),EOL,def(DEFINE mı var?),t(fonksiyon sırası),new_Proc(yeni fonksiyon mu geldi?).. Bu fonksiyon temelde switch yapısı üzerinden çalışmakta olup, sonradan eklenen kontroller ve durumlar için if operatörü kullanılmıştır. Pseudo kod yazımında BEGIN END sayıları farklıysa veya dizi yazarken syntax hatası var ise gerekli hata mesajı ekrana basılır. Bütün fonksiyonlar başlangıçta void olarak dosyaya yazılır. Fonksiyon sonunda RETURN komutu gelirse ve devamında VOID yazmıyorsa fonksiyon başına dönüş tipini yazabilmek adına, yeni fonksiyon geldiyse o anki file pointer'ı ftell ile bir değişkende (pos) tutarak sonrasında buraya dönmeyi sağladım.

1.3.12. void put_tabs_new_lines(FILE *, int, int, int, int)

Yazılan dosyanın pointer'ını, tab_counter'ı i'yi, close_bracelet veya open_bracelet'ı ve EOL'ı parametre olarak alır. switch(i) yapısını kullanarak (i, token'ın Reserved Word structure'ında hangi elemana denk geldiğini belirtiyor.) BEGIN ise isOpen_Bracelet, END ise isClose_Bracelet, döngüyse isLoop, değilse isNot_Loop fonksiyonunu çağırır. Bu dört fonksiyon 1.3.13. 1.3.14. 1.3.15. ve 1.3.16. kısımlarda anlatılacaktır.

1.3.13. void isLoop(FILE *, int)

Döngüyse çağrılır. Dosya pointer ve tab_counter'ı parametre alır. Alt satıra geçip tab counter kadar tab yazar.

1.3.14. void isNotLoop(FILE *, int)

Döngü değilse çağrılır. Yazılan dosya pointer'ı ve tab_counter'ı parametre alır. isLoop'a ek olarak satır sonuna noktalı virgül koyar. Sonrasında alt satıra geçip tab counter kadar tab yazar.

1.3.15. void isOpen_Bracelet(FILE *, int, int, int)

Yazılan dosya pointer'ı, tab_counter, i, open_bracelet'ı parametre alır. Tab karakterlerini yazar. Süslü parantez açma'yı yazar. Tab_counter ve open bracelet değişkenlerini artırır. Alt satıra geçip tekrar tableri yazar.

1.3.16. void isClose_Bracelet(FILE *, int, int, int)

Yazılan dosya pointer'ı, tab_counter, i, close_bracelet'ı parametre alır. Tab karakterlerini yazar. Süslü parantez kapatma'yı yazar. Tab_counter ve close bracelet değişkenlerini artırır. Alt satıra geçip tekrar tableri yazar.

1.3.17. void isArray(FILE *,char *)

Token, diziyse çağrılır. Yazılan dosya pointer'ı ve token'ı parametre alır. İlgili delimiter'lar kullanılarak token syntax olarak mantıklı parçalara bölünerek yazılır.

1.3.18. void print_Vars(FILE *,char *,int)

Yeni fonksiyon geldikçe çağrılır ve o fonksiyona ait değişkenler fonksiyonun ilk satırında tanımlanır. Yazılan dosya pointer'ı, fonksiyon adı, fonksiyon sırasını parametre olarak alır. İlgili kontrolleri yaparak tüm değişkenleri deklare eder. En sonunda noktalı virgül ekler.

1.3.19. void isFunc(FILE *,char *)

Token, yeni fonksiyon başlangıcıysa veya fonksiyon çağrımıysa çağrılır. Yazılan dosya pointer'ı ve token parametre olarak alınır. Dizi olma durumu pointer olma durumu, fonksiyon çağırma olma durumu, parametresiz bir fonksiyon olma durumu gibi çeşitli durumlar kontrol edilerek doğru şekilde dosyaya yazılır.

1.3.20. void GetVal_PrintLine(FILE *, char *, int)

GET_VAL veya PRINT_LINE sözde komutları bulunması durumunda çağrılır. Yazılan dosya pointer'ı, token ve i'yi parametre olarak alır. while döngüsü içinde sırayla kelimeler/değişkenler alınır. Alınan tokenda $ karakteri mevcutsa ilgili değişken tipine göre format kodları(%d,%c,vs.) yazılır. Eğer i=20 ise PRINT_LINE'dır ve ekstradan \n karakteri yazılır." " içindeki kısım yazıldıktan sonra, GET_VAL ise başına & ekleyerek değilse eklemeden, değişkenler yazılır.

1.3.21. void return_type(FILE *,char *,long int)

Eğer Pseudo fonksiyon sonunda RETURN yoksa çağrılmaz ve fonksiyon void olarak kalır. Eğer RETURN varsa ve VOID ise return void; yazılır. RETURN var fakat VOID değilse değişken yazılır. Sonrasında ilgili fonksiyon başına, pos değişkeniyle dönülür.(daha önce print_file fonksiyonu içinde PROCEDURE gelince pos'u o anki

file pointer konumunu atamıştık.) RETURN'den sonra gelen değişken tipi, void'in üzerine yazılır. Tüm kodu kaydırmak yerine Bu daha optimum bir çözüm diye düşünerek ilk başta maksimum ve minimum durum için(_ldp=long double *),(int) voidden sonra gerekli sayıda boşluk bırakmıştım. Bu yüzden üzerine yazarken bir sorun çıkması beklenmemektedir.

1.4. Fonksiyonların Birbirleriyle İlişkileri

- **int main()'in Çağırdıkları**
  - open_file
  - find_file_size
  - isDeclared
  - arrange_tokens
  - print_file
- **DATABASE * arrange_bools(char *,char *, char *,char *)'un Çağırdıkları**
  - create_list_element
- **DATABASE * arrange_bools(char *,char *, char *,char *)'u Çağıranlar**
  - isDeclared
- **DATABASE *create_list_element(unsigned int, unsigned int, unsigned int)'ı Çağıranlar**
  - arrange_bools
- **char *strtok_2(char*,const char*)'yi Çağıranlar**
  - isDeclared
  - isArray
- **int find_file_size(FILE *)'ı Çağıranlar**
  - main
- **int for_loop_exception(FILE *, char *)'ı Çağıranlar**
  - print_file
- **void isDeclared(char*)'in Çağırdıkları**
  - strtok(String kütüphanesi fonksiyonu)
  - strtok_2
  - arrange_bools
  - insert
- **void isDeclared(char*)'i Çağıranlar**
  - main
- **void insert(DATABASE*)'i Çağıranlar**
  - isDeclared
- **void arrange_tokens()'i Çağıranlar**
  - main
- **void print_file(char *,char *)'in Çağırdıkları**
  - open_file
  - print_Vars
  - strtok
  - isArray
  - isFunc

- for_loop_exception
- GetVal_PrintLine
- return_type
- put_tabs_new_lines

❖ **void print_file(char *,char *)'i Çağıranlar**
- main

❖ **void put_tabs_new_lines(FILE *, int, int, int, int)'in Çağırdıkları**
- isOpen_Bracelet
- isClose_Bracelet
- isLoop
- isNotLoop

❖ **void put_tabs_new_lines(FILE *, int, int, int, int)'i Çağıranlar**
- print_file

❖ **void isLoop(FILE *, int)'u Çağıranlar**
- put_tabs_new_lines

❖ **void isNotLoop(FILE *, int)'u Çağıranlar**
- put_tabs_new_lines

❖ **void isOpen_Bracelet(FILE *, int, int, int)'i Çağıranlar**
- put_tabs_new_lines

❖ **void isClose_Bracelet(FILE *, int, int, int)'i Çağıranlar**
- put_tabs_new_lines

❖ **void isArray(FILE *,char *)'in Çağırdıkları**
- strtok_2

❖ **void isArray(FILE *,char *)'i Çağıranlar**
- print_file
- isFunc
- GetVal_PrintLine

❖ **void print_Vars(FILE *,char *,int)'ı Çağıranlar**
- print_file

❖ **void isFunc(FILE *,char *)'ın Çağırdıkları**
- strtok
- isArray

❖ **void isFunc(FILE *,char *)'ı Çağıranlar**
- print_file

❖ **void GetVal_PrintLine(FILE *, char *, int)'ın Çağırdıkları**
- strtok
- isArray

❖ **void GetVal_PrintLine(FILE *, char *, int)'ı Çağıranlar**
- print_file

❖ **void return_type(FILE *,char *,long int)'ın Çağırdıkları**
- strtok

❖ **void return_type(FILE *,char *,long int)'ı Çağıranlar**

- print_file

1.5. Kaynak Kodu ve Açıklamaları

1.5.1. database.h dosyası

```
1   #define LINESIZE 100
2   #define KEYWORDS_SIZE 23
3   #define TRUE 1
4   #define FALSE 0
5   #define USES_VARS_SIZE 7
6   #ifndef NULL
7   #define NULL 0
8   #endif
9   typedef struct database{    //SEMBOL TABLOSU İÇİN OLUŞTURULAN BAĞLI LİSTE
10          char *token;
11          char *identifier;
12          char *data_type;
13          char proc[12];                  //Bulunduğu fonksiyon
14          unsigned int isarray;           //Dizi mi,diziyse eleman sayısı
15          unsigned int ispointer:1;       //Pointer mı
16          unsigned int isPara:1;          //Parametre mi?
17          struct database *db_next;
18  }DATABASE;                  //-----------------------------------------
19
20  typedef struct {            //Fonksiyon başlangıçlarını tutan struct
21          long int pos;
22          char funcname[12];
23  }POSITIONS;                 //--------------------------------------
24
25  typedef struct {     //RESERVED WORDLERİ TUTAN STRUCT
26          char *token;
27          char *c_code;
28  }RESERVED;          //-----------------------------
29  //FONKSIYON PROTOTİPLERİ
30  FILE *open_file(char *,char *);
31  DATABASE * arrange_bools(char *,char *, char *,char *);
32  DATABASE *create_list_element(unsigned int, unsigned int, unsigned int);
33  char *strtok_2(char*,const char*);
34  int find_file_size(FILE *);
35  int for_loop_exception(FILE *, char *);
36  void isDeclared(char*);
37  void insert(DATABASE*);
38  void arrange_tokens();
39  void print_file(char *,char *);
40  void put_tabs_new_lines(FILE *, int, int, int, int);
41  void isLoop(FILE *, int);
42  void isNot_Loop(FILE *, int);
43  void isOpen_Bracelet(FILE *, int, int, int);
44  void isClose_Bracelet(FILE *, int, int, int);
45  void isArray(FILE *,char *);
46  void print_Vars(FILE *,char *,int );
47  void isFunc(FILE *,char *,char *);
48  void GetVal_PrintLine(FILE *, char *, int);
49  void return_type(FILE *,char *,long int);
50  //---------------------------------
51  static DATABASE *head;
52  void res_keywords();
53  char uses_vars[3][USES_VARS_SIZE]={
54          /*uses_char=*/FALSE,
55          /*uses_double=*/FALSE,
56          /*uses_file=*/FALSE,
```

```
        /*uses_float=*/FALSE,
        /*uses_int=*/FALSE,
        /*uses_long=*/FALSE,
        /*uses_longdouble=*/FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE,
                            FALSE, };

char *vars[USES_VARS_SIZE]={
       "char",
       "double",
       "FILE",
       "float",
       "int",
       "long",
       "long double" };

static char iffunccall=FALSE;
int ARR_SIZE=100;
POSITIONS poss[5];
RESERVED keywords[KEYWORDS_SIZE];
void res_keywords()
{
       keywords[0].token="PROCEDURE";//ilk satırı alınca PROCEDURE MAIN()'le aynıysa direkt olarak "
main()" yazdırılacak, değilse hata verilecek.
       keywords[0].c_code="void         ";//sonraki satırlarda PROCEDURE gelirse, sonraki token olan
fonksiyon adı yazdırılacak. return gelmezse void kalacak. PROCEDURE gelirse satır sonuna noktalı
virgül ekleme
       keywords[1].token="BEGIN";
       keywords[1].c_code="{";
       keywords[2].token="END";
       keywords[2].c_code="}";
       keywords[3].token="RETURN";//RETURN geldiyse return yazdır ve sonraki token neyse direkt
yazdır. ve o tokenin tipini fonksiyonun başına yaz.
       keywords[3].c_code="return ";//return yoksa fonksiyon default olarak void kalacak.
       keywords[4].token="VOID";  //return void gelirse fonksiyonun tanımındaki void değişmeyecek.
       keywords[4].c_code="void"; //onun dışında VOID görülen yerlerde void yazılacak.
       keywords[5].token="<-";
       keywords[5].c_code="= ";
       keywords[6].token="<";
       keywords[6].c_code=" < ";
       keywords[7].token=">";
       keywords[7].c_code=" > ";
       keywords[8].token="AND";
       keywords[8].c_code="&&";
       keywords[9].token="OR";
       keywords[9].c_code="||";
       keywords[10].token="EQUAL";
       keywords[10].c_code=" == ";
       keywords[11].token="NOT_EQUAL";
       keywords[11].c_code=" != ";
       keywords[12].token="FOR";
       keywords[12].c_code="for (";
       keywords[13].token="WHILE";
```

```
        keywords[13].c_code="while ";
        keywords[14].token="IF";
        keywords[14].c_code="if ";
        keywords[15].token="DO";
        keywords[15].c_code="";
        keywords[16].token="THEN";
        keywords[16].c_code="";
        keywords[17].token="LOOP";
        keywords[17].c_code=")";
        keywords[18].token="(";
        keywords[18].c_code="(";
        keywords[19].token=")";
        keywords[19].c_code=")";
        keywords[20].token="PRINT_LINE";
        keywords[20].c_code="printf(\"";
        keywords[21].token="GET_VAL";
        keywords[21].c_code="scanf(\"";
        keywords[22].token="DEFINE";
        keywords[22].c_code="";
        //stricmp ile non-case sensitive olarak karşılaştır.
}
```

1.5.2. main.c Dosyası

```
int main()
{
        FILE *fp1;
    char p_code[LINESIZE];
        char *data_array,filep_name[100],filec_name[100],output_file[100],*sysout;
        int file_size;
        DATABASE *ptr;
        system("COLOR F5");
        printf(">PseudocodeConverter\n");
        printf("Pseudo kodun bulundugu dosya ismini giriniz: ");
        scanf("%s",filep_name);
        printf("C kodunun yazilacagi dosya ismini giriniz: ");
        scanf("%s",filec_name); //Pseudo kod dosyası ve C kodu yazılacak dosya ismi alındı.
        fp1=open_file(filep_name,"r"); //Pseudo kod dosyası açıldı.
        file_size=find_file_size(fp1);    // Dosya boyutu bulundu.
        rewind(fp1);  //Dosya başına dönüldü.
        printf("Pseudocode dosyasi okunuyor..\n");
        fgets(p_code,LINESIZE-1,fp1);     //Pseudo kod dosyasının ilk satırı p_code stringine alındı.
        data_array=(char*)malloc(file_size);     //data_array stringine yer açıldı
        strcpy(data_array,p_code);
        while(fgets(p_code,LINESIZE-1,fp1))
        {
                strcat(data_array,p_code); //Bütün pseudo kod dosyası data_array stringine alındı.
        }
        fclose(fp1);
        printf("Pseudocode dosyasi basariyla okundu.\n");
        isDeclared(data_array);//data_array hazır, sembol tablosu oluşturmak için isDeclared
        printf("Degiskenlerle ilgili bilgileri tutan symboltable, linkli liste olarak
olusturuldu.\n");
        arrange_tokens();//Sembol tablosu oluşturuldu. 1.3.10. kısımdaki olaylar için çağrılır.
        print_file(filep_name,filec_name);//Sembol tablosu hazır, C kodu dosyaya yazılmaya başlanır
        printf("C kodu dosyaya yazildi.\n");
        printf("Olusturulacak exe adini giriniz: ");
        scanf("%s",output_file);
        sysout=(char*)malloc(strlen(filec_name)+strlen(output_file)+10);
        strcpy(sysout,"gcc ");
        strcat(sysout,filec_name);
        strcat(sysout," -o ");
        strcat(sysout,output_file);
        system(sysout);      //Oluşturulan c kodu dosyası gcc ile derlenir.
        printf("Executable dosya olusturuldu.");
```

```
42          system("PAUSE");
43          system(output_file); //Derleme sonu oluşturulan dosya çalıştırılır.
44          return 1;
45  }

46  FILE *open_file(char *file_name,char* file_mode)
47  {
48          //Dosya Açma fonksiyonu
49          FILE *fp1;
50          if((fp1=fopen(file_name,file_mode))==NULL)
51          {
52                  printf("'%s' dosyasi '%s' modunda acilamadi.",file_name,file_mode);
53                  system("PAUSE");
54                  exit(1);
55          }
56          printf("'%s' dosyasi, '%s' modunda acildi.\n",file_name,file_mode);
57          return fp1;
58  }

59  int find_file_size(FILE *fp1)
60  {
61          //Dosya Boyutu bulan fonksiyon
62          fseek(fp1,0,SEEK_END);
63          return ftell(fp1);
64  }

65  void isDeclared(char* data_array)
66  {
67          DATABASE *ptr,*tmptr;
68          char *tokenlimiter=" $.\"()<>\n\t";
69          char *datalimiter="_[]";
70          char *token,*ident,*dtype,*tmptoken,new_Proc=FALSE,Proc[12];
71          static char ifarray=FALSE;
72          int k=0,array_size=100;
73          token=strtok(data_array,tokenlimiter);
74          if(!strcmp(token,"PROCEDURE")) //İlk satırda fonksiyon başlangıcı mı var?
75          {
76                  new_Proc=TRUE;
77          }
78          if(strchr(token,'[')) //Yoksa bir dizi mi?
79          {
80                  ifarray=TRUE;
81          }
82          tmptoken=(char*)malloc(strlen(token)+1); //token'ı kaybetmemek için oluşturulan
83          strcpy(tmptoken,token);                  //tmptoken'a token'ı alıyoruz
84          if(!_stricmp(token,"DEFINE"))     //DEFINE geldiyse
85          {                                                   //tokenize et
86                  ident=strtok_2(token,datalimiter);
87                  dtype=strtok_2(NULL,datalimiter);
88                  array_size=atoi(strtok_2(NULL,"]"));     //stringi int'e dönüştür.
89          }
90          else
91          {
92                  ident=strtok_2(token,datalimiter);       //DEFINE gelmediyse sadece tokenize et
93                  dtype=strtok_2(NULL,datalimiter);
94          }
95          goto label2;        //ilk çağrımda label2 ye git.
96  label1:
97          while(token=strtok(NULL,tokenlimiter)) //satır sonuna gelene kadar tokenize etmeye devam.
98          {
99                  array_size=100;             //dizi boyutu default olarak 100
100                 if(new_Proc)        //Satırda fonksiyon başlıyorsa new_Proc set edilmişti.
101                 {
102                         strcpy(Proc,token);  //Proc stringine fonksiyon adını alıyoruz
103                         new_Proc=FALSE;      //new_Proc default olarak FALSE
```

```
104                 }
105                 if(!strcmp(token,"PROCEDURE"))    //yeni fonksiyon geldiyse new_Proc set.
106                 {
107                         new_Proc=TRUE;
108                         k++;          //Fonksiyon sırasını tutar.
109                 }
110                 if(strchr(token,'['))       //[ varsa dizidir.
111                 {
112                         ifarray=TRUE;
113                 }
114                 strcpy(tmptoken,token);     //token'ı sonradan kullanacağım, değeri değişeceği için
115     tmptoken'a alıyorum.
116                 if(!_stricmp(token,"DEFINE"))//DEFINE mı geldi?
117                 {
118                         token=strtok(NULL," \n");
119                         ifarray=TRUE;
120                         ident=strtok_2(token,datalimiter);
121                         dtype=strtok_2(NULL,datalimiter);
122                         array_size=atoi(strtok_2(NULL,"]"));     //dizi boyutunu ayarlıyoruz
123                 }
124                 else
125                 {
126                         ident=strtok_2(token,datalimiter);       //dizi değilse normal devam.
127                         dtype=strtok_2(NULL,datalimiter);
128                 }
129                 for(tmptr=head;tmptr!=NULL;tmptr=tmptr->db_next)//Değişken daha önce tanımlandı mı?
130                 {
131                         if(!strcmp(ident,tmptr->identifier))
132                         {
133                                 if(!strcmp(Proc,tmptr->proc))
134                                 {
135                                         ifarray=FALSE;
136                                         goto label1;  //Tanımlandıysa label1 e gidip yeni token'ı al.
137                                 }
138                         }
139                 }
140     label2:     if(!dtype)    //İlk çevrimde direkt buraya atlıyordu.
141                 {                       //Token, değişken değilse dtype null olacak.
142                         goto label1;  //label1e gidip yeni token al
143                 }
144                 switch((int)*dtype)  //switch'in içindeki case, değişken tiplerinin
145                 {                                       //ilk harfinin ASCII'si
146                         case 99 :   //CHAR
147                                 ptr=arrange_bools(ident,"char",tmptoken,&ifarray);
148                                 uses_vars[k][0]=TRUE;
149                                 break;
150                         case 100 :  //DOUBLE
151                                 ptr=arrange_bools(ident,"double",tmptoken,&ifarray);
152                                 uses_vars[k][1]=TRUE;
153                                 break;
154                         case 102 :  //FLOAT ve FILE *
155                                 if(strlen(dtype)==4)       //FILE * mı?
156                                 {
157                                         ptr=arrange_bools(ident,"FILE",tmptoken,&ifarray);
158                                         uses_vars[k][2]=TRUE;
159                                 }
160                                 else                                        //FLOAT mı?
161                                 {
162                                         ptr=arrange_bools(ident,"float",tmptoken,&ifarray);
163                                         uses_vars[k][3]=TRUE;
164                                 }
165                                 break;
166                         case 105 :   //INT
167                                 ptr=arrange_bools(ident,"int",tmptoken,&ifarray);
168                                 uses_vars[k][4]=TRUE;
```

```
169                             break;
170                     case 108 :  //LONG
171                             if(strchr(dtype,'d'))      //LONG DOUBLE MI?
172                             {
173                                     ptr=arrange_bools(ident,"long double",tmptoken,&ifarray);
174                                     uses_vars[k][6]=TRUE;
175                             }
176                             else                                //LONG MU?
177                             {
178                                     ptr=arrange_bools(ident,"long",tmptoken,&ifarray);
179                                     uses_vars[k][5]=TRUE;
180                             }
181                             break;
182                     default :
183                             goto label1; //Değişken değilse buraya düşer.label1e gidip yeni token al.
184             }
185             if(strchr(dtype,'p'))//Değişken tipi, p içeriyorsa pointerdır.
186             {
187                     ptr->ispointer=TRUE;
188             }
189             else
190             {
191                     ptr->ispointer=FALSE;
192             }
193             ptr->identifier=ident;      //değişken adını ayarla.
194             if(ifarray)         //diziyse, boyutunu ayarla.
195             {
196                     ptr->isarray=array_size;
197                     ifarray=FALSE;
198             }
199             strcpy(ptr->proc,Proc);     //hangi fonksiyonda geçtiğini ayarla
200             insert(ptr);
201     }
202     return;
203 }

204 DATABASE *create_list_element(unsigned int strsize, unsigned int dtypesize, unsigned int tokensize)
205 {
206     DATABASE *ptr;
207     ptr=(DATABASE*) malloc(sizeof(DATABASE));       //structa yer aç
208     ptr->identifier=(char*)malloc(strsize+1);       //Değişken adına
209     ptr->data_type=(char*)malloc(dtypesize+1);      //Değişken tipine
210     ptr->token=(char*)malloc(tokensize+1);   //Değişken token'ına yer aç.
211     if(ptr==NULL)
212     {
213             printf("Creating list element with malloc failed.\n");
214             exit(1);
215     }
216     ptr->db_next=NULL;
217     return ptr;
218 }

219 void insert(DATABASE *e)
220 {
221     DATABASE *ptr;
222     int i;
223     if(head==NULL)      //İlk elemansa
224     {
225             head=e;       //baş eleman olarak ayarla.
226             return;
227     }
228     for(i=0;i<USES_VARS_SIZE;i++)     //Tüm değişken tipleri için dön.
229     {
230             if(_stricmp((e->data_type),vars[i])==0)  //Aradığım değişken tiplerinin
231             {                                        //tutulduğu alana kadar buraya girmeyecek.
```

```
                        if(i==4)            //int'se en başta tutulacak.
                        {
                                e->db_next=head;
                                head=e;
                                return;
                        }       //int değilse, ilgili değişken tipinin en son elemanı olarak eklenir.
                        for(ptr=head;((ptr->db_next!=NULL)&&(ptr->db_next-
>data_type!=vars[i]));ptr=ptr->db_next);
                        e->db_next=ptr->db_next;
                        ptr->db_next=e;
                        return;
                }
        }
}

void arrange_tokens()
{
        DATABASE *ptr;
        for(ptr=head;ptr!=NULL;ptr=ptr->db_next) //En baştan başla, sona kadar dön
        {
                strcpy(ptr->token,ptr->identifier);      //token'a değişken adını al
                switch((int)*ptr->data_type) //değişken tipine göre ilgili
                {                                        //uzantıyı ekle.
                        case 70:      //FILE *
                                strcat(ptr->token,"_file");
                                break;
                        case 99:      //CHAR
                                strcat(ptr->token,"_c");
                                break;
                        case 100:     //DOUBLE
                                strcat(ptr->token,"_d");
                                break;
                        case 102:     //FLOAT
                                strcat(ptr->token,"_f");
                                break;
                        case 105:     //INT
                                strcat(ptr->token,"_i");
                                break;
                        case 108:     //LONG
                                if(strlen(ptr->data_type)==11)
                                {             //long double olma durumunu kontrol et.
                                        strcat(ptr->token,"_ld");
                                }
                                else
                                {
                                        strcat(ptr->token,"_l");
                                }
                                break;
                }
                if(ptr->ispointer)  //pointersa p uzantısını da ekle
                {
                        strcat(ptr->token,"p");
                }
        }
}

char *strtok_2(char *s1, const char *delimit)
{
    static char *lastToken_2 = NULL; /* UNSAFE SHARED STATE! */
    char *tmp;
    /* Skip leading delimiters if new string. */
    if ( s1 == NULL ) {
        s1 = lastToken_2;
        if (s1 == NULL)         /* End of story? */
            return NULL;
```

```
    } else {
        s1 += strspn(s1, delimit);
    }
    /* Find end of segment */
    tmp = strpbrk(s1, delimit);
    if (tmp) {
        /* Found another delimiter, split string and save state. */
        *tmp = '\0';
        lastToken_2 = tmp + 1;
    } else {
        /* Last segment, remember that. */
        lastToken_2 = NULL;
    }
    return s1;
}

void print_file(char *filep_name, char *filec_name)
{
       FILE *fp1,*fp2;
       DATABASE *ptr;
       char *token,def=FALSE;
       char p_code[LINESIZE], Proc[12],new_Proc=FALSE;
       static char  iffunc=FALSE;
       int i,k,t=-1,tab_counter=0,open_bracelet=0,close_bracelet=0,EOL=0;
       res_keywords();
       fp1=open_file(filep_name,"r");    //pseudo kod dosyası aç.
       fp2=open_file(filec_name,"w+");   //c kodu yazılacak dosyayı aç.
       fputs("#include <stdio.h>\n",fp2);
       fgets(p_code,LINESIZE-1,fp1);     //ilk satırı al.
       if(_stricmp("PROCEDURE MAIN()\n",p_code))      //main var mı?
       {             //Yoksa label4 e git.
              goto label4;
       }
       else
       {
              poss[t+1].pos=ftell(fp2);  //fonksiyon başlangıcının pozisyonu
              strcpy(poss[t+1].funcname,"main()");//fonksiyon adı
              fputs("void main()       \n",fp2);//main başlangıcını yaz.
              strcpy(Proc,"MAIN");//şuanki fonksiyon main.
       }

label3:       //yeni fonksiyon başlangıcı gelirse buraya atlar.
       t++;   //fonksiyon sırasını tutan değişken artar.
       fgets(p_code,LINESIZE-1,fp1);     //yeni satırı al
       if(!_stricmp("BEGIN\n",p_code))   //BEGINse { koy
       {
              fputs("{\n\t",fp2);
              tab_counter++;       //indentasyon ayarlamaları.
              open_bracelet++;
       }
       else
       {
              printf("Mainden sonra suslu parantez eksik!!");
              system("PAUSE");
              exit(1);
       }
       print_Vars(fp2,Proc,t);    //t. sıradaki fonksiyona ait değişkenleri yaz.
       new_Proc=FALSE;

       while((!new_Proc)&&(fgets(p_code,LINESIZE-1,fp1)))//yeni fonksiyon gelene ya da
       {                                        //dosya sonuna gelene kadar yaz.
              iffunc=FALSE; //default olarak bunlar false olmalı.
              iffunccall=FALSE;
label4:       //ilk satırda mainden farklı fonksiyon varsa buraya atlar.
              token=strtok(p_code," \t\n");
```

```
359             if(!strcmp(token,"PROCEDURE"))    //yeni fonksiyon başlangıcı mı geldi?
360             {
361                     new_Proc=TRUE;
362                     if(!strcmp(Proc,"MAIN"))          //önceki fonksiyon mainse, system pause yaz.
363                     {                                 //
364                             fseek(fp2,-3,SEEK_CUR);   //
365                             putc('\t',fp2);           //
366                             fputs("system(\"PAUSE\");",fp2);
367                             putc('\n',fp2);           //
368                             putc('}',fp2);            //
369                             putc('\n',fp2);           //----------------------------------------
370                     }
371                     poss[t+1].pos=ftell(fp2);  //pozisyonu structa al.
372             }
373             i=0;   //while için hazırlık.
374             while((i<KEYWORDS_SIZE)&&(_stricmp(keywords[i].token,token)))
375             {      //reserved word structındaki eleman sayısına gelene kadar
376                     i++;//veya tokenlar eşleşene kadar dönecek ve i++ olacak.
377             }
378             switch(i)     //en son i ne kalmışsa ona göre ilgili
379             {                     //case'e girecek.
380                     case KEYWORDS_SIZE: //eğer i, boyuta eşitse, reserved word değildir.
381                             ptr=head;     //değişken midir diye bakılır.
382                             while((ptr!=NULL)&&(_stricmp(ptr->token,token)))
383                             {//struct sonuna gelene yada tokenlar eşleşene kadar ptr'ı next yap.
384                                     ptr=ptr->db_next;
385                             }
386                             switch((int)ptr)     //pointer değerine göre;
387                             {
388                                     case 0:       //eğer NULL'sa dizi olabilir.
389                                             if(strchr(token,'['))
390                                             {
391                                                     if(!strchr(token,']'))
392                                                     {
393                                                             printf("Brackets does not match!!\n");
394                                                             system("PAUSE");
395                                                     }
396                                                     isArray(fp2,token);  //diziyse isArray çağrılır
397                                             }
398                                             else if(strchr(token,']'))
399                                             {
400                                                     printf("Brackets does not match!!\n");
401                                                     system("PAUSE");
402                                             }
403                                             else   //dizi,reserved Word,değişken değilse
404                                             {//function call olursatır başındaki ilk token,sayı olamaz.
405                                                     fputs(token,fp2);//buraya girer token yazılır dosyay
406                                                     if(*token=='\"')
407                                                     {
408                                                             putc(' ',fp2);
409                                                     }
410                                                     iffunccall=TRUE;     //iffunccall set,
411                                                     iffunc=FALSE;        //iffunc false edilir ve,
412                                                     isFunc(fp2,token,Proc);     //isFunc çağrılır.
413                                             }
414                                             break;
415                                     default:
416                                             if(ptr->ispointer)  //ptr, pointersa
417                                             {                   //ve char değilse
418                                                     if(ptr->data_type!="char")
419                                                     {             //* yazılır.
420                                                             putc('*',fp2);
421                                                     }
422                                             }
423                                             fputs(ptr->identifier,fp2);//değişken adı yazılır.
```

```
424                                     fputs(" ",fp2);//okunabilirlik için boşluk koyulur.
425                                     break;
426                             }
427                             break;
428                     case 0:       //FONKSIYON MU ?
429                             fputs(keywords[i].c_code,fp2);
430                             iffunc=TRUE;
431                             iffunccall=FALSE;
432                             token=strtok(NULL,"("); //fonksiyon adı alınır
433                             strcpy(Proc,token);  //Proc'a atanır
434                             strcpy(poss[t+1].funcname,Proc);//pozisyonu alınır
435                             if(!_stricmp(token,"MAIN"))//main'se dosyaya yazılır.
436                             {
437                                     fputs("main",fp2);
438                             }
439                             else
440                                     fputs(token,fp2);    //değilse direkt basılır.
441                             putc('(',fp2);
442                             putc(' ',fp2);//isFunc yapısı gereği ('dan sonra bir boşluk olmalı.
443                             isFunc(fp2,token,Proc);     //isFunc çağrılır
444                             break;
445                     case 1:
446                             fputs(keywords[i].c_code,fp2);     //1=BEGIN
447                             tab_counter++;              //tab sayısı artmalı
448                             open_bracelet++;//süslü parantez açma sayısı artar
449                             for(k=0;k<tab_counter;k++)
450                             {
451                                     fputs("\t",fp2);     //gerekli tablar koyulur
452                             }
453                             break;
454                     case 2:       //2=END, tablar koyulmuştu. 1 tab geriye yazılmalı.
455                             fseek(fp2,-1,SEEK_CUR);     //1 karakter geri git.
456                             fputs(keywords[i].c_code,fp2);//END yaz.
457                             tab_counter--;       //tabların sayısı azalır.
458                             close_bracelet++;//süslü parantez kapatma sayısı artar.
459                             for(k=0;k<tab_counter;k++)
460                             {
461                                     fputs("\t",fp2);     //gerekli tablar koyulur.
462                             }
463                             break;
464                     case 12:      //12=FOR
465                             ////FOR icin ozel durum olustur.  for ( i = 0; i < n; i++ ) mesela.
466                             i=for_loop_exception(fp2, token);
467                             break;
468                     case 20: //20=PRINT_LINE
469                             //PRINT_LINE için özel durum oluştur.
470                             GetVal_PrintLine(fp2, token, i);
471                             break;
472                     case 21:  //21=GET_VAL
473                             //GET_VAL için özel durum oluştur.
474                             GetVal_PrintLine(fp2, token, i);
475                             break;
476                     case 3://3=RETURN
477                             fputs(keywords[i].c_code,fp2); //return yaz ve,
478                             return_type(fp2,token,poss[t].pos);//ilgili fonksiyonu çağır.
479                             break;
480                     case 22:  //22=DEFINE geldiyse, ekrana yazılmayacak,
481                             while(token=strtok(NULL," \n"));//sadece deklarasyon kısmında,
482                             def=TRUE;     //bir farklılık olacak ve bu da def değişkeni,
483                             break;        //aracılığıyla olacak.
484                     default:      //diğer reserved wordler geldiyse de,
485                             fputs(keywords[i].c_code,fp2);//direkt karşılığını yaz.
486                             break;
487             }
488             while(token=strtok(NULL," \t\n"))//Alınan satırının sonuna gelene kadar döner.
```

```
489             {
490                     i=0;//while'a hazırlık
491                     while((i<KEYWORDS_SIZE)&&(_stricmp(keywords[i].token,token)))
492                     {//reserved word structındaki eleman sayısına gelene kadar
493                             i++;//veya tokenlar eşleşene kadar dönecek ve i++ olacak.
494                     }
495                     switch(i)     //en son i ne kalmışsa ona göre ilgili
496                     {                       //case'e girecek.
497                             case KEYWORDS_SIZE: //eğer i, boyuta eşitse, reserved word değildir.
498                                     ptr=head;     //değişken midir diye bakılır.
499                                     while((ptr!=NULL)&&(_stricmp(ptr->token,token)))
500                                     {//struct sonuna gelene yada tokenlar eşleşene kadar ptr'ı next
501                                             ptr=ptr->db_next;
502                                     }
503                                     switch((int)ptr)
504                                     {               //pointer değerine göre;
505                                             case 0:       //Eğer null'sa
506                                                     if(strchr(token,'['))
507                                                     {               //Dizi olabilir.
508                                                             if(!strchr(token,']'))
509                                                             {
510                                                                     printf("Brackets does not match!!\n");
511                                                                     system("PAUSE");
512                                                             }
513                                                             isArray(fp2,token);
514                                                     }
515                                                     else if(strchr(token,']'))
516                                                     {
517                                                             printf("Brackets does not match!!\n");
518                                                             system("PAUSE");
519                                                     }
520                                                     else
521                                                     {//dizi, reserved word değilse, değişken değilse
522                                                             fputs(token,fp2);//buraya girer ve yazılır.
523                                                             if(*token=='\"')
524                                                             {
525                                                                     putc(' ',fp2);
526                                                             }
527                                                             if(strchr(token,'('))
528                                                             {// ( varsa function calldur,PROCEDURE
529                                                                     olmadığı için
530                                                                     iffunccall=TRUE;//yeni fonksiyon
531                                                                     başlangıcı olamaz.
532                                                             }
533                                                             if(iffunc||iffunccall)
534                                                             {       //iki değişkenden biri set ise isFunc
535 çağrılır.
536                                                                     isFunc(fp2,token,Proc);
537                                                             }
538                                                     }
539                                                     break;
540                                             default:
541                                                     if(ptr->ispointer) //ptr, pointersa,
542                                                     {               //buraya girer ve tipi char değilse
543                                                             if(ptr->data_type!="char")
544                                                             {               //* yazılır.
545                                                                     putc('*',fp2);
546                                                             }
547                                                     }
548                                                     fputs(ptr->identifier,fp2); //değişken tipi yazılır.
549                                                     break;
550                                     }
551                                     break;
552                             case 0:       //FONKSIYON MU ? 0=PROCEDURE
553                                     fputs(keywords[i].c_code,fp2); //fonksiyonsa, buraya girer,
```

```
554                                     iffunc=TRUE;  //void yazar.
555                                     iffunccall=FALSE; //iffunc set, iffunccall FALSE olur.
556                                     isFunc(fp2,token,Proc);//ilgili fonksiyon çağrılır.
557                                     break;
558                             case 1:       // 1= BEGIN.
559                                     EOL=0; //put_tabs_new_lines gereği 0 olmalıdır. ; konmaması için
560                                     put_tabs_new_lines(fp2,tab_counter,i,open_bracelet,EOL);
561                                     break;
562                             case 2:// 2= END.
563                                     EOL=0;//put_tabs_new_lines gereği 0 olmalıdır. ; konmaması için
564                                     put_tabs_new_lines(fp2,tab_counter,i,close_bracelet,EOL);
565                                     break;
566                             default://diğer reserved wordler geldiyse de,
567                                     fputs(keywords[i].c_code,fp2);//direkt karşılıklarını yaz
568                                     break;
569                     }
570             }
571             //Satir sonuna gelindiginde yeni satira gecme ve indentetion ayarlamalari.
572             EOL=1;//End of line değişkeni set edilir.
573             if(iffunc)
574             {       //fonksiyon başlangıcıysa,
575                     i=15; //i 15 yapılır. sonuna ; eklenmesin diye.
576                     iffunc=FALSE;
577                     iffunccall=FALSE;
578             }
579             if(!def)//DEFINE gelmediyse buraya gelecek. DEFINE geldiyse birşey yazma.
580                     put_tabs_new_lines(fp2,tab_counter,i,close_bracelet,EOL);//Adı anlatıyor.
581             def=FALSE;//default olarak false olmalı.
582     }
583     if(new_Proc)            //Yeni fonksiyon geldiyse
584     {               //label3'e atlar.
585             goto label3;
586     }
587     //Buraya geldiyse, dosya bitmiştir,
588     if(!_stricmp(Proc,"MAIN"))//son yazılan fonksiyon mainse
589     { //system pause yazılır.
590             fseek(fp2,-3,SEEK_CUR);
591             putc('\t',fp2);
592             fputs("system(\"PAUSE\");",fp2);
593             putc('\n',fp2);
594             putc('}',fp2);
595             putc('\n',fp2);
596     }
597     if(open_bracelet!=close_bracelet)
598     { //ikisi eşit değilse syntax hatası vardır.
599             printf("\nSuslu parantezler eslesmiyor!\n");
600             system("PAUSE");
601     }
602     fclose(fp1); //açılan dosyalar kapatılır.
603     fclose(fp2);
604     return;
605 }

606 void isFunc(FILE *fp2,char *token,char *Proc)
607 {
608     DATABASE *ptr;
609     int control=FALSE;
610     char isARR=FALSE, *token2;
611     while(token=strtok(NULL," \n,]()"))//tokenize et.
612     {
613             isARR=FALSE;
614             if(strstr(token,"["))       //dizi mi?
615             {
616                     isARR=TRUE;
617                     token2=(char*)malloc(strlen(token)+1);
```

```
618                     strcpy(token2,token);
619                     token=strtok_2(token,"[");
620             }
621             ptr=head;
622             while((ptr!=NULL)&&((_stricmp(Proc,ptr->proc))||(_stricmp(ptr->token,token))))
623             { //ptr null olana yada tokenlar eşleşene kadar ptr'ı next yap.
624                     ptr=ptr->db_next;
625             }
626             if(ptr!=NULL)
627             { //ptr null değilse değişkendir.
628                     if(!iffunccall) //function call değilse,
629                     {               //yeni fonksiyon başlangıcıdır.Bu yüzden,
630                             fputs(ptr->data_type,fp2);//değişken tipini yaz,
631                             putc(' ',fp2); //ve boşluk bırak.
632                             if(!_stricmp(Proc,ptr->proc))
633                                     ptr->isPara=TRUE;//Parametreyse, fonksiyon içinde
634                     }                                       //tekrar tanımlanmasın kontrolü.
635                     if(ptr->ispointer)//pointersa
636                     { //* koy.
637                             if(ptr->data_type!="char")
638                                     putc('*',fp2);
639                             else if(!iffunccall)
640                                     putc('*',fp2);
641                     }
642                     fputs(ptr->identifier,fp2);//ve değişken adını yaz.
643                     if(isARR)
644                     {
645                             putc('[',fp2);
646                             putc(']',fp2);
647                     }
648                     putc(',',fp2);  //sonraki elemanla arasına virgül koy
649                     control=TRUE;//buraya girdiyse control set edilir.
650             }
651             else if(!control) //control false ise
652             {       //dizi,değişken değildir.(metin olabilir.)
653                     fputs(token,fp2);//direkt tokenı yaz.
654                     putc(',',fp2); //sonraki elemanla arasına virgül koy.
655             }
656             else if(ptr==NULL)//ptr, nullsa değişken değildir.
657             { //1 karakter geri gel.
658                     fseek(fp2,-1,SEEK_CUR);
659             }
660             control=FALSE;//control default olarak false yapılmalıdır.
661             if(isARR)
662             strcpy(token,token2);
663         }
664         if(!control)//en son token, değişken değilse buraya girer.
665                 fseek(fp2,-1,SEEK_CUR);//1 geri gelerek fazla karakteri siler.
666         putc(')',fp2);//fonksiyon kapatılır.
667     }

668     void put_tabs_new_lines(FILE *fp2, int tab_counter, int i, int close_bracelet, int EOL)
669     {
670             switch(i)//parametre olarak gelen i'ye göre,
671             {               //ilgili case'e girer.
672                     case 1:// 1=BEGIN
673                             if(!EOL)
674                             {       //EOL 0 gelir eğer i=1 ise.
675                                     isOpen_Bracelet(fp2,tab_counter,i,close_bracelet);
676                                     break;//satır sonu ; eklemez.
677                             }
678                     case 2: //2=END
679                             if(!EOL)
680                             {       //EOL 0 gelir eğer i=2 ise.
681                                     isClose_Bracelet(fp2,tab_counter,i,close_bracelet);
```

```
                        break;//satır sonu ; eklemez.
                }
        case 15:// 15=DO   (WHILE olduğunu bildirir.
                isLoop(fp2,tab_counter);//satır sonu ; eklemez.
                break;
        case 16:// 16=THEN (IF olduğunu belirtir.)
                isLoop(fp2,tab_counter);//satır sonu ; eklemez.
                break;
        case 17:// 17=LOOP (FOR olduğunu bildirir.)
                isLoop(fp2,tab_counter);//satır sonu ; eklemez.
                break;
        default:// Döngü değilse BEGIN,END değilse buraya girer.
                isNot_Loop(fp2,tab_counter);//satır sonu ; ekler.
                break;
    }
}

void isLoop(FILE *fp2, int tab_counter)
{           //Döngüyse put_tabs_new_lines'dan buraya dallanır,
    int k;//satır sonuna ; eklemeden gerekli tableri ve,
    fputs("\n",fp2);//yeni satır karakterini yazar.
    for(k=0;k<tab_counter;k++)
    {
            fputs("\t",fp2);
    }
}

void isNot_Loop(FILE *fp2, int tab_counter)
{           //Döngü değilse put_tabs_new_lines'dan buraya dallanır,
    int k;//satır sonuna ; ekler, gerekli tableri ve,
    fputs(";\n",fp2);//yeni satır karakterini yazar.
    for(k=0;k<tab_counter;k++)
    {
            fputs("\t",fp2);
    }
}

void isOpen_Bracelet(FILE *fp2, int tab_counter, int i, int open_bracelet)
{                   //BEGIN'se put_tabs_new_lines'tan buraya dallanır,
    int k;//satır sonuna ; eklemeden gerekli,
    fputs("\n",fp2);//indentasyonları yapar.
    for(k=0;k<tab_counter;k++)//tableri ekler ve
    {                                   //yeni satır karakterini yazar.
            fputs("\t",fp2);
    }
    fputs(keywords[i].c_code,fp2);
    fputs("\n",fp2);
    tab_counter++;
    open_bracelet++;
    for(k=0;k<tab_counter;k++)
    {
            fputs("\t",fp2);
    }
}

void isClose_Bracelet(FILE *fp2, int tab_counter, int i, int close_bracelet)
{                   //END'se put_tabs_new_lines'tan buraya dallanır,
    int k;//satır sonuna ; eklemeden gerekli,
    tab_counter--;//indentasyonları yapar.
    close_bracelet++;
    fputs("\n",fp2);//tableri ekler ve
    for(k=0;k<tab_counter;k++)
    {                                   //yeni satır karakterini yazar.
            fputs("\t",fp2);
    }
```

```
743         fputs(keywords[i].c_code,fp2);
744         fputs("\n",fp2);
745         for(k=0;k<tab_counter;k++)
746         {
747                 fputs("\t",fp2);
748         }
749   }

750   void isArray(FILE *fp2,char *token)
751   {                     //Dizi var ise
752         token=strtok_2(token,"_");//dizi adını alır
753         fputs(token,fp2);//dosyaya yazar
754         putc('[',fp2);//köşeli parantezi yazar
755         token=strtok_2(NULL,"[");//değişken tipini alır ama yazmaz.
756         token=strtok_2(NULL," ]_");//indis adını alır(varsa)
757         fputs(token,fp2);//ve yazar.
758         token=strtok_2(NULL,",");//indis tipini alır ama yazmaz.
759         while(token=strtok_2(NULL," \n,]"))
760         { //yukarıdaki işlemleri, indis bitene kadar yapar,
761                 fputs(token,fp2);//çünkü dizi_c[i_i+j_i] durumu olabilir.
762                 if(token=strtok_2(NULL,"_]"))
763                 {
764                         fputs(token,fp2);
765                         token=strtok_2(NULL," \n,");
766                 }
767                 else
768                 {
769                 }
770         }
771         putc(']',fp2);
772   }

773   int for_loop_exception(FILE *fp2, char *token)
774   {
775         //FOR icin ozel durum olustur.  for (i=0; i < n; i++) seklinde yazdirabilmek icin.
776         DATABASE *ptr;
777         char *hold;//kaybedilmemesi gereken bir token bunda tutulacaktır.
778         int i=12;//reserved words[12]=FOR
779         fputs(keywords[i].c_code,fp2);//for ( yazılır.
780         while(token=strtok(NULL," .\t\n"))
781         {  //satır sonuna kadar tokenize edilir.
782                 i=0;
783                 while((i<KEYWORDS_SIZE)&&(_stricmp(keywords[i].token,token)))
784                 { //FOR'un bittiğini gösteren LOOP gelirse i=17 olduğunda buradan çıkar.
785                         i++;
786                 }
787                 switch(i)
788                 {
789                         case KEYWORDS_SIZE:
790                                 ptr=head;//FOR bitmemişse buraya girer.
791                                 while((ptr!=NULL)&&(_stricmp(ptr->token,token)))
792                                 {
793                                         ptr=ptr->db_next;
794                                 }
795                                 switch((int)ptr)
796                                 {
797                                         case 0://değişken değilse sayı olabilir.buraya girer.
798                                                 fputs(token,fp2);//token neyse o yazılır.
799                                                 fputs("; ",fp2);//for syntaxı gereği ; araya atılır
800                                                 fputs(hold,fp2);//holdda tutulan değişken adı yazılır.
801                                                 putc('<',fp2);//küçük olduğu süre boyunca dönecek.
802                                                 token=strtok(NULL," .\t\n");//maksimum terim alınır.
803                                                 ptr=head;
804                                                 while((ptr!=NULL)&&(_stricmp(ptr->token,token)))
805                                                 {
```

```
                                        ptr=ptr->db_next;
                                }
                                fputs(ptr->identifier,fp2);//maks.terim hangi değişkense,
                                fputs("; ",fp2);//yazılır ve ; koyulur.
                                fputs(hold,fp2);//holddaki indis yazılır.
                                fputs("++",fp2);//for syntax'ı gereği bu konulur.
                                break;
                        default:
                                fputs(ptr->identifier,fp2);//ilk başta buraya girecek ve
                                hold=ptr->identifier;//indisi ekrana basıp, hold'a koyacak.
                                break;
                        }
                        break;
                case 5:/Bosu bosuna ekstradan bir bosluk karakteri,
                        putc('=',fp2);//yazilmasin diye direkt olarak = basiyorum.
                        break; //(database.h'da keywords[5].c_code="= ")
                default://LOOP gelirse ) basılır.
                        fputs(keywords[i].c_code,fp2);
                        break;
                }
        }
        return i;//i değeri kullanılacağı için geri döndürür.
}

void GetVal_PrintLine(FILE *fp2, char *token, int i)
{
        DATABASE *ptr;
        char *tmpcode;//token'ı kaybetmememem lazım. tmpcode'a kopyalanacak.
        fputs(keywords[i].c_code,fp2);//printf(" veya scanf(" yazılır.
        tmpcode=(char*)malloc(LINESIZE);
        strcpy(tmpcode,token);//tmpcode'a token'ı kopyala.
        while(token=strtok(NULL,"“” \"]\n\t"))//satır sonuna kadar,
        {                               //ayarlanmış olan delimiterlerle tokenize et.
                strcat(tmpcode,token);//tmpcode'a token'ı ekle
                if(strchr(token,'$'))//$ karakteri geldiyse,
                {               //ilgili uzantılarla uyuşma durumuna göre,
                        if(strstr(token,"_c"))//ilgili format kodunu yaz.
                        {
                                if(strstr(token,"_cp"))
                                        fputs("%s",fp2);
                                else
                                        fputs("%s",fp2);
                        }
                        else if(strstr(token,"_d"))
                        {       fputs("%f",fp2);     }
                        else if(strstr(token,"_f"))
                        {       fputs("%f",fp2);     }
                        else if(strstr(token,"_ld"))
                        {       fputs("%lf",fp2);    }
                        else if(strstr(token,"_l"))
                        {       fputs("%li",fp2);    }
                        else if(strstr(token,"_i"))
                        {       fputs("%d",fp2);     }
                        if(1)  //her durumda girmesi gerekiyor.
                        {
                                if(strchr(token,','))//virgül varsa
                                { //virgül yaz.
                                        putc(',',fp2);
                                }
                                if(strchr(token,'.'))//nokta varsa,
                                { //nokta yaz.
                                        putc('.',fp2);
                                }
                                putc(' ',fp2);//okunabilirlik için boşluk bırak
                        }
```

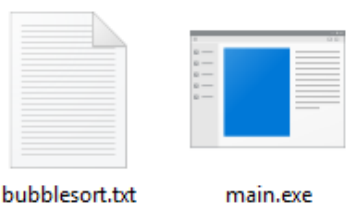
```
870                 }
871                 else//$ karakteri yoksa düz kelimedir.
872                 { //direkt yaz
873                         fputs(token,fp2);
874                         putc(' ',fp2);//boşluk bırak.
875                 }
876         }
877         fseek(fp2,-1,SEEK_CUR);//sona gelince ekstra boşluğu sil
878         if(i==20)
879         { //printf ise \n  " ve ,  yazılmalı.
880                 fputs("\\n\", ",fp2);
881         }
882         else
883         {       //değilse sadece " ve , yazılmalı.
884                 fputs("\", ",fp2);
885         }
886         if(strchr(tmpcode,'$'))//$ karakteri varsa
887         { //değişken kullanılmıştır.yoksa düz metin yazılmıştır.
888                 token=strtok(tmpcode," $\n\t");//$ dan öncesini al ama yazma, düz metin.
889                 while(token=strtok(NULL," $\n\t"))
890                 { //satır sonuna kadar tokenize et.
891                         if(strchr(token,'['))
892                         { //[ varsa dizidir.
893                                 if(i==21)
894                                 { //scanf ise & eklenmeli
895                                         putc('&',fp2);
896                                 }
897                                 isArray(fp2,token);//isArray'i çağır.
898                                 fputs(", ",fp2);//sonraki elemanla ayır.
899                         }
900                         for(ptr=head;ptr!=NULL;ptr=ptr->db_next)
901                         {
902                                 if(strstr(token,ptr->token))
903                                 {
904                                         if(i==21)
905                                         { //scanf ise
906                                                 if(!ptr->ispointer)//pointer değilse
907                                                         putc('&',fp2);//& eklenmeli.
908                                         }
909                                         fputs(ptr->identifier,fp2);//değişken adı yazılır.
910                                         fputs(", ",fp2);//sonraki elemanla ayrılır.
911                                         break;
912                                 }
913                         }
914                 }
915         }
916         fseek(fp2,-2,SEEK_CUR);//fazladan yazılan , ve boşluğu siler.
917         putc(')',fp2);
918         return;
919 }

920 void print_Vars(FILE *fp2, char *Proc,int k)
921 {
922         DATABASE *ptr;
923         int i;
924         char text[4];//dizi boyutu için.(maksimum 999 olabilir)
925         for(i=0;i<USES_VARS_SIZE;i++)
926         {                       //tüm değişken tipleri için dönecek.
927                 if(uses_vars[k][i]==TRUE)
928                 {               //ilgili değişken tipi varsa
929                         fputs(vars[i],fp2); //değişken tipini basar,
930                         putc(' ',fp2);              //boşluk bırakır.
931                         for(ptr=head;(ptr!=NULL)&&(ptr->data_type!=vars[i]);ptr=ptr->db_next);
932                         for(;(ptr!=NULL)&&(ptr->data_type==vars[i]);ptr=ptr->db_next)
933                         {       //ilgili değişken tipinin olduğu kısma gelene kadar null statement döner.
```

```
                        //değişken tipi değişmediği sürece döner.
                        if((!_stricmp(Proc,ptr->proc))&&(!ptr->isPara))//Parametreyse tanımlama.
                        {  //şuanki fonksiyon, ptr'ın bulunduğu fonksiyonsa,
                                if(ptr->ispointer)//buraya girer
                                {               //ptr pointersa * basar
                                        putc('*',fp2);
                                }//değişken adını yazar.
                                else if(i==2)
                                        putc('*',fp2);
                                fputs(ptr->identifier,fp2);
                                if(ptr->isarray)
                                { //değişken bir diziyse
                                        putc('[',fp2);//[ basar.
                                        fputs(itoa(ptr->isarray,text,10),fp2);//int to string
yapılıp
                                        putc(']',fp2);//dizi boyutu basılır ve ] basılır.
                                }
                                putc(',',fp2);//sonraki elemandan ayrılır.
                                putc(' ',fp2);//okunabilirlik için boşluk koyulur.
                        }
                }               //fazladan yazılan , ve boşluk silinir.
                fseek(fp2,-2,SEEK_CUR);
                fputs(";\n\t",fp2);//satır sonuna gelindi, alta geçilir.
        }
    }
}

DATABASE * arrange_bools(char *ident,char *dtype, char *tmptoken,char *ifarray)
{
    DATABASE *ptr;//ptr'a parametrelerle yer açılır.
    ptr=create_list_element(strlen(ident),strlen(dtype)+1,strlen(tmptoken));
    if(*ifarray)
    { //diziyse ilgili kısım set,
        ptr->isarray=TRUE;
    }
    else
    { //dizi değilse ilgili kısım false yapılır.
        ptr->isarray=FALSE;
    }
    ptr->data_type=dtype;//değişken tipi ilgili kısma yazılır.
    return ptr;//ptr değeri döner
}

void return_type(FILE *fp2,char *token,long int pos)
{
    DATABASE *ptr;//RETURN komutu varsa bu fonksiyon çağrılır.
    int space_len,i;//space_len, dönüş tipine göre ayarlanacak.
    token=strtok(NULL," _\n\t");//değişken adı alınır.
    if(!strcmp(token,"VOID"))//VOID'se
    {                                   //void yazılır.
        fputs("void",fp2);
        return;//ve çıkılır.
    } //void değilse alt satıra geçer.
    fputs(token,fp2);//değişken adı yazılır.
    fseek(fp2,pos,SEEK_SET);//ilgili fonksiyonun başlangıcı pos'ta var.oraya gidilir.
    for(ptr=head;(ptr->db_next!=NULL)&&(strcmp(token,ptr->identifier));ptr=ptr->db_next);
    fputs(ptr->data_type,fp2);//İlgili değişkeni bulana kadar null for döner.
    space_len=11-strlen(ptr->data_type);//space len değişken tipine göre ayarlanır.
    for(i=0;i<space_len;i++)//space_len kadar boşluk koy.
        putc(' ',fp2);
    token=strtok(NULL," \t\n");//satır sonuna gelmek için yazılır.
    if(ptr->ispointer)
    {                   //değişken pointersa
        putc('*',fp2);//* koyulur.
    }
```

```
997         else if(!strcmp("int",ptr->data_type))
998         {//pointer değilse, int ise ekstra bir boşluk koyulur,
999               putc(' ',fp2);//çünkü int void'den 1 karakter kısa.
000         }
001         putc(' ',fp2);
002         if(!strcmp(ptr->proc,"MAIN"))//main için özel durum var.
003               fputs("main()",fp2);//
004         else//main değilse, değişkenin ait olduğu,
005               fputs(ptr->proc,fp2);//fonksiyon yazılır.
006         fseek(fp2,0,SEEK_END);//dosya sonuna geri dön.
    }
```

1.6. Ekran Görüntüleri

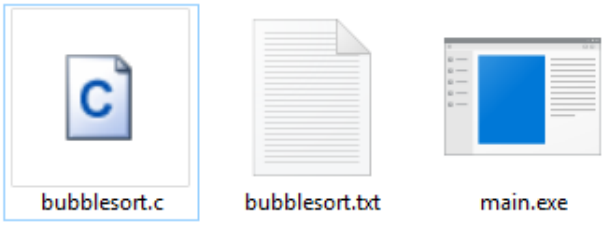
```
PROCEDURE MAIN()
BEGIN
        PRINT_LINE "Dizinin eleman sayisini giriniz"
        GET_VAL "$n_i"
        bubble( n_i)
END
PROCEDURE bubble(n_i)
BEGIN
        PRINT_LINE "Dizinin elemanlarini giriniz"
        FOR k_i <- 0..n_i LOOP
        BEGIN
                GET_VAL "$arr_i[k_i]"
        END
        PRINT_LINE "Sirali Dizi:"
        i_i <- 0
        WHILE ( ( i_i ) < ( n_i - 1 ) ) DO
        BEGIN
                j_i <- 0
                WHILE ( ( j_i ) NOT_EQUAL ( n_i - i_i - 1 ) ) DO
                BEGIN
                        IF ( arr_i[j_i] > arr_i[j_i,+,1] ) THEN
                        BEGIN
                                temp_i <- arr_i[j_i]
                                arr_i[j_i] <- arr_i[j_i,+,1]
                                arr_i[j_i,+,1] <- temp_i
                        END
                        j_i <- j_i + 1
                END
                i_i = i_i + 1
        END
        FOR k_i <- 0..n_i LOOP
        BEGIN
                PRINT_LINE "$arr_i[k_i]"
        END
END
```



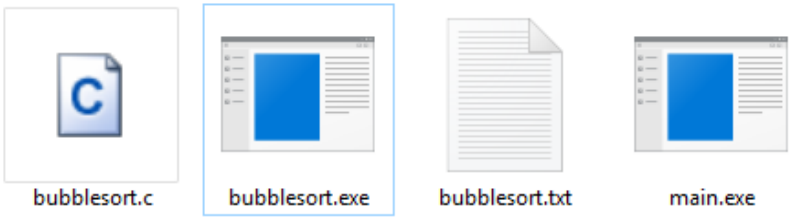

```
>PseudocodeConverter
Pseudo kodun bulundugu dosya ismini giriniz: _
```

```
>PseudocodeConverter
Pseudo kodun bulundugu dosya ismini giriniz: bubblesort.txt
C kodunun yazilacagi dosya ismini giriniz: bubblesort.c
'bubblesort.txt' dosyasi, 'r' modunda acildi.
Pseudocode dosyasi okunuyor..
Pseudocode dosyasi basariyla okundu.
Degiskenlerle ilgili bilgileri tutan symboltable, linkli liste olarak olusturuldu.
'bubblesort.txt' dosyasi, 'r' modunda acildi.
'bubblesort.c' dosyasi, 'w+' modunda acildi.
C kodu dosyaya yazildi.
Olusturulacak exe adini giriniz: _
```

```c
void main()
{
    int n;
    printf("Dizinin eleman sayisini giriniz\n");
    scanf("%d", &n);
    bubble(n);
    system("PAUSE");
}
void        bubble( int n)
{
    int temp, j, i, arr[100], k;
    printf("Dizinin elemanlarini giriniz\n");
    for (k=0; k<n; k++)
    {
        scanf("%d", &arr[k]);
    }
    printf("Sirali Dizi:\n");
    i = 0;
    while ((i) < (n-1))
    {
        j = 0;
        while ((j) != (n-i-1))
        {
            if (arr[j] > arr[j+1])
            {
                temp = arr[j];
                arr[j]= arr[j+1];
                arr[j+1]= temp;
            }
            j = j+1;
        }
        i =i+1;
    }
    for (k=0; k<n; k++)
    {
        printf("%d\n", arr[k]);
    }
}
```



>bubblesort.exe




```
Press any key to continue . . .
Dizinin eleman sayisini giriniz
5
Dizinin elemanlarini giriniz
3
2
4
1
5
Sirali Dizi:
1
2
3
4
5
Press any key to continue . . .
```

1.7. Programın Kısıtları

- İlk fonksiyon, main fonksiyonu değilse çalışmadığı bazı durumlar var. C kodunu %99 doğru oluşturuyor. Fakat parametreyi tekrar fonksiyon içinde deklare etmek gibi bazı ufak hatalar var.
- Syntax olarak bazı kısımlarda bazı karakterlerin kullanılması zorunludur. Operatörlerin içinde, fonksiyon çağrımlarında ilk parametreden önce, vs..
- Sonradan yapılan eklemeler, bazı kısımlarda generic ifadeler yerine istisnalar için olan kodlara sebep oldu, bu da kodun okunabilirliğini azalttı. Kısıt değil fakat beğenmediğim zayıf yönlerinden birisi. Algoritmayı en başta daha doğru kurabilseydim goto kullanmama da gerek kalmayacaktı.
- Daha başka eksiklikler, istisnalar da vardır fakat programın son halinde gözüme çarpanlar bunlar.

1.8. Programın Güçlü Yönleri

- Sembol tablosu olabildiğince düzgün ve açıklayıcı oluşturulduğu için %99 doğru çalıştığını düşünüyorum.
- Projede hata varsa bu sembol tablosu yazdırılarak varsa hatalar kolayca görülebilir.
- Değişkenlere ve fonksiyonlara elimden geldiğince anlamlı isimler vermeye çalıştım. Bu da kodun okunabilirliğini ve debug edilmesini kolaylaştırıyor.
- Tekrar eden kısımlar için fonksiyonlar oluşturdum, sonrasında proje geliştirilmeye devam edilse bile bu generic fonksiyonlar tekrar tekrar kullanılabilir.
- Static yerine dynamic memory allocation kullanarak gereksiz alan kullanımından kaçındım.
- İlk başta düşündüğüm karakter karakter okuma ve karşılaştırma yerine sonrasında çok daha efektif olan strtok fonksiyonunu kullandım ve bu işimi çok kolaylaştırdı. Delimiter'ları duruma göre dilediğim gibi ayarlayarak tek fonksiyonla değişkenleri okumasını sağladım.

1.9. Yararlanılan Kaynaklar ve Bağlantılar

- https://stackoverflow.com/questions/37039786/solve-ÿ-end-of-file-in-c
- https://softwareengineering.stackexchange.com/questions/165543/how-to-write-a-very-basic-compiler
- https://www.geeksforgeeks.org/c-program-find-size-file/
- https://www.tutorialspoint.com/c_standard_library/string_h.htm
- http://www.asciitable.com
- https://stackoverflow.com/questions/18248047/allocate-memory-for-a-struct-with-a-character-pointer-in-c
- https://stackoverflow.com/questions/29638598/strtok-why-you-have-to-pass-the-null-pointer-in-order-to-get-the-next-token
- http://www.cplusplus.com/reference/cstring/strtok/
- https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm
- https://codereview.stackexchange.com/questions/101694/implementation-of-symbol-table-in-c
- https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_71/rtref/stricmp.htm
- https://stackoverflow.com/questions/5820810/case-insensitive-string-comp-in-c
- https://www.quora.com/Can-we-use-logical-operator-in-switch-case-implementation-in-c++
- http://patorjk.com/software/taag/#p=moreopts&c=c%2B%2B&f=Alpha&t=Hayri%20Cakir
- https://www.slideshare.net/Munni28/lexical-analysis-68764636
- https://stackoverflow.com/questions/17992686/visual-studio-2012-syntax-highlighting-is-turning-off-and-on
- https://stackoverflow.com/questions/10319685/new-mingw-gcc-doesnt-do-anything
- http://www.mingw.org
- https://docs.microsoft.com/en-us/cpp/cppcx/crt-functions-not-supported-in-universal-windows-platform-apps?view=vs-2015
- https://www.javatpoint.com/escape-sequence-in-c
- https://www.tutorialspoint.com/c_standard_library/ctype_h.htm
- https://stackoverflow.com/questions/444382/visual-studio-how-can-i-see-the-same-file-in-two-separate-tab-groups/46446829
- http://personal.ee.surrey.ac.uk/Personal/R.Bowden/C/printf.html
- https://www.tutorialspoint.com/c_standard_library/c_function_printf.htm
- https://www.rose-hulman.edu/class/csse/resources/MinGW/installation.htm
- https://stackoverflow.com/questions/7021725/how-to-convert-a-string-to-integer-in-c
- https://stackoverflow.com/questions/8257714/how-to-convert-an-int-to-string-in-c/23840699
- https://www.tutorialspoint.com/compiler_design/compiler_design_pdf_version.htm
- https://stackoverflow.com/questions/5691650/conflicting-types-error-when-compiling-c-program-using-gcc