Son Teslim tarihi: 11.12.2024

Ödev #3

OĞUZHAN TOPALOĞLU
Ç19052025 – Grup 1

Bilgisayar Mühendisliği Bölümü,
Elektrik-Elektronik Fakültesi,
Yıldız Teknik Üniversitesi

Istanbul, 2024

Video Linki: https://youtu.be/fLAY6pMXlgw

İÇERİK

1. Problemin Çözümü

Problemi iki kısma ayırarak çözdüm: parsing ve hashing kısımları.

Parsing kısmında okunan dosya bir kez tur atılarak parse ediliyor ve değişken belirlendiğinde bir counter arttırılarak hashmap boyutu belirleniyor. Ardından ikinci bir tur daha atılarak oluşturulmuş hashmap'e bilgi girişi yapılıyor. Parsing yapılırken öncelikle "int main() {" ifadesi bulununcaya dek boşluk gibi geçilecek karakterler geçilerek bir buffer'da string içinde okunan karakterler toplanıyor. Int main bulunduktan sonra da noktalı virgüllere göre satırlar ayrılıp iki kategoriye ayrılıyor. Birisi değişken tanımı yapılan "int x" ya da "char a,b,c" gibi satırlar iken öbürü ise herhangi bir aritmetik operasyonu içeren ve değişken tanımlamak yerine değişken erişimi yapan ve değişkenleri kullanan satırlar. Değişken tanımı yapılan satırlarda önceden deklere edilen bir değişken yeniden deklere edilmeye çalışıldıysa bir hata alınabilir, değişken kullanan satırlarda ise deklere edilmemiş bir değişken kullanıldıysa hata alınabilir.

Hashing kısmında ise double hashing algoritması ile sırayla tanımlamalar yapılıyor ve lookup ile insert fonksiyonları ile değişken tanımlama satırları ile değişken kullanma satırları çalıştırılıyor.

2. Karşılaşılan Sorunlar

Ödevde en çok zorlandığım kısım parsing oldu. Çoğu kez “int main() {“ ifadesi tam olarak bulunmadan değişik parsing işlemleri yapılarak null pointer exception’ı alıyordum. Ayrıca parsing’de bir sürü iç içe parsing olduğundan dolayı buffer’lar kafamı sıkça karıştırdı.

Örneğin tüm dosya içeriği bir buffer’da iken okunmuş kısım ayrı bir buffer’da ve bu okunmuş kısmın içinden değişken türü ve ismi çekiliyorsa bunlar da ayrı bir buffer’da…

Hashing kısmında pek sorun yaşamadım.

3. Karmaşıklık Analizi

Kodda bulunan fonksiyonların çoğu farklı karmaşıklığa sahiptir. Teker teker önemli fonksiyonları analiz etmek gerekirse,

1. Dosya içinde (okunan kodda) kaç karakter olduğunu bulan “long getFileLength(const char*)” fonksiyonu O(1)dir. Bunun nedeni hesaplama için kullanılan fseek ve ftell fonksiyonlarının O(1) olmasından kaynaklanmaktadır.

2. Dosya içeriğini okuyan "char* getFileContents(const char*, long) fonksiyonu" O(n)'dir. Bunun nedeni, dosyanın tamamını okumak için kullanılan fread fonksiyonunun dosyadaki tüm karakterlere sırayla erişim yapmasıdır. Eğer dosya uzunluğu n ise, fread işlemi doğrudan dosyanın büyüklüğüne bağlıdır.

3. Hashmap boyutunu belirleyen "int determineHashmapSize(char*, long, int)" fonksiyonu O(n)'dir. Bunun nedeni, kod içeriğinde tüm karakterlerin taranarak değişkenlerin belirlenmesi ve sayılmasıdır. Koddaki karakter sayısı n ise, her karakter sırasıyla kontrol edilir. Bununla birlikte, sonrasında asal sayıyı belirlemek için kullanılan isPrime fonksiyonu ek bir hesaplama gerektirir ancak n her zaman daha baskın olur.

4. Hashing fonksiyonu olan "int hash(Hashmap*, char*, int*)" O(m)'dir. Bunun nedeni, verilen anahtarın uzunluğu m (string'in uzunluğu) ise, her bir karakterin sırayla işlenmesi gerektiğidir. Bu işlem sırasında hornerKey hesaplanarak bir indeks oluşturulur. Ancak bu, sadece anahtar uzunluğuna bağlı olduğu için sabit hash tablosu boyutuyla değişmez.

5. Hashmap'e değişken ekleyen "void hashmapInsert(Hashmap*, char*, char*)" fonksiyonu worst case'de O(n) best case'de O(1) olarak çalışır. Bunun nedeni, double hashing kullanılarak indeks çakışmalarının etkili bir şekilde çözülmesidir. Ancak, en kötü durumda yani çok fazla çakışma varsa karmaşıklık O(n) olabilir.

4. Ekran Görüntüleri

code3.txt

```
1
2
3   int main () {
4     int _x1;
5     float _x2;
6     char _x3;
7     char _x4;
8     float _x5;
9     int _x6;
10    _x5 = _x6 + _x1 + _x2;
11    _x1 = 20;
12    _x7 = 5;
13    _y2 = 20;
14    int _y2;
15    _y2 = 20;
16  }
17
```

code1.txt

```
1
2   int main (    )
3   {
4     int _aa,_bb,     _cc;
5     char     _aa;
6     char _x;
7     _aa =5;
8
9
10
11
12
13    _xx = 9;
14    _bb= _aa +   _dd;
15  }
16
```

code2.txt

```
1
2   int main () {
3       int _x, _y, _z;
4       float _a;
5       float _x;
6       _x2 = 10;
7   }
8
```

run.bat

```
1
2
3   @echo off
4
5   IF EXIST "C19052025.exe" (del "C19052025.exe")
6
7   "C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\x86_64-w64-mingw32-gcc-4.9.2" C19052025.c -o C19052025.exe
8
9   @echo on
10
11  C19052025.exe code1.txt DEBUG
12
13
```

```
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3> .\run.bat

C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>C19052025.exe code1.txt NORMAL
HATA: _aa degiskeni daha once deklere edilmistir.
HATA: _xx degiskeni deklere edilmemistir.
HATA: _dd degiskeni deklere edilmemistir.
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>
```

```
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3> .\run.bat

C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>C19052025.exe code2.txt NORMAL
HATA: _x degiskeni daha once deklere edilmistir.
HATA: _x2 degiskeni deklere edilmemistir.
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>
```

```
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3> .\run.bat

C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>C19052025.exe code3.txt NORMAL
HATA: _x7 degiskeni deklere edilmemistir.
HATA: _y2 degiskeni deklere edilmemistir.
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>
```

```
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3> .\run.bat

C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>C19052025.exe code1.txt DEBUG
DEBUG: Deklere edilmis degisken sayisi: 5
DEBUG: Sembol tablosunun uzunlugu: 11
DEBUG: _aa degiskeni icin ilk adres degeri: 8, son adres: 8
DEBUG: _bb degiskeni icin ilk adres degeri: 7, son adres: 7
DEBUG: _cc degiskeni icin ilk adres degeri: 6, son adres: 6
HATA: _aa degiskeni daha once deklere edilmistir.
DEBUG: _x degiskeni icin ilk adres degeri: 7, son adres: 4
HATA: _xx degiskeni deklere edilmemistir.
HATA: _dd degiskeni deklere edilmemistir.
DEBUG: Sembol tablosunun son hali:
  Hashmap (Size: 11):
    - 0: ..
    - 1: ..
    - 2: ..
    - 3: ..
    - 4: char _x;
    - 5: ..
    - 6: int _cc;
    - 7: int _bb;
    - 8: int _aa;
    - 9: ..
    - 10: ..
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>
```

```
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3> .\run.bat

C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>C19052025.exe code2.txt DEBUG
DEBUG: Deklere edilmis degisken sayisi: 5
DEBUG: Sembol tablosunun uzunlugu: 11
DEBUG: _x degiskeni icin ilk adres degeri: 7, son adres: 7
DEBUG: _y degiskeni icin ilk adres degeri: 8, son adres: 8
DEBUG: _z degiskeni icin ilk adres degeri: 9, son adres: 9
DEBUG: _a degiskeni icin ilk adres degeri: 6, son adres: 6
HATA: _x degiskeni daha once deklere edilmistir.
HATA: _x2 degiskeni deklere edilmemistir.
DEBUG: Sembol tablosunun son hali:
  Hashmap (Size: 11):
    - 0: ..
    - 1: ..
    - 2: ..
    - 3: ..
    - 4: ..
    - 5: ..
    - 6: float _a;
    - 7: int _x;
    - 8: int _y;
    - 9: int _z;
    - 10: ..
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>
```

```
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3> .\run.bat

C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>C19052025.exe code3.txt DEBUG
DEBUG: Deklere edilmis degisken sayisi: 7
DEBUG: Sembol tablosunun uzunlugu: 17
DEBUG: _x1 degiskeni icin ilk adres degeri: 0, son adres: 0
DEBUG: _x2 degiskeni icin ilk adres degeri: 1, son adres: 1
DEBUG: _x3 degiskeni icin ilk adres degeri: 2, son adres: 2
DEBUG: _x4 degiskeni icin ilk adres degeri: 3, son adres: 3
DEBUG: _x5 degiskeni icin ilk adres degeri: 4, son adres: 4
DEBUG: _x6 degiskeni icin ilk adres degeri: 5, son adres: 5
HATA: _x7 degiskeni deklere edilmemistir.
HATA: _y2 degiskeni deklere edilmemistir.
DEBUG: _y2 degiskeni icin ilk adres degeri: 15, son adres: 15
DEBUG: Sembol tablosunun son hali:
  Hashmap (Size: 17):
    - 0: int _x1;
    - 1: float _x2;
    - 2: char _x3;
    - 3: char _x4;
    - 4: float _x5;
    - 5: int _x6;
    - 6: ..
    - 7: ..
    - 8: ..
    - 9: ..
    - 10: ..
    - 11: ..
    - 12: ..
    - 13: ..
    - 14: ..
    - 15: int _y2;
    - 16: ..
PS C:\Users\oguzh\Desktop\Kodlama\zSpecial\university-stuff\algoritma analizi\hw3>
```