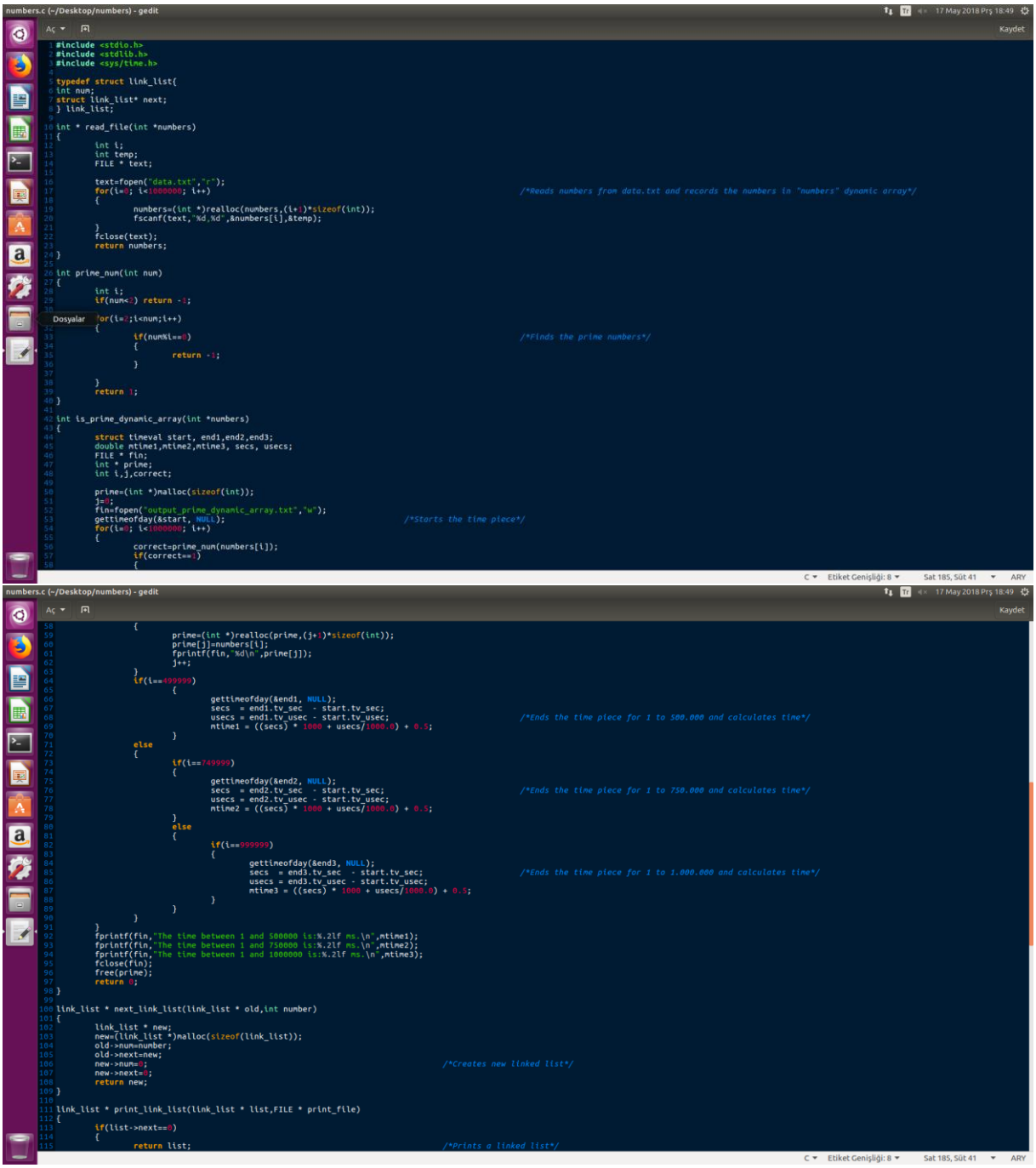
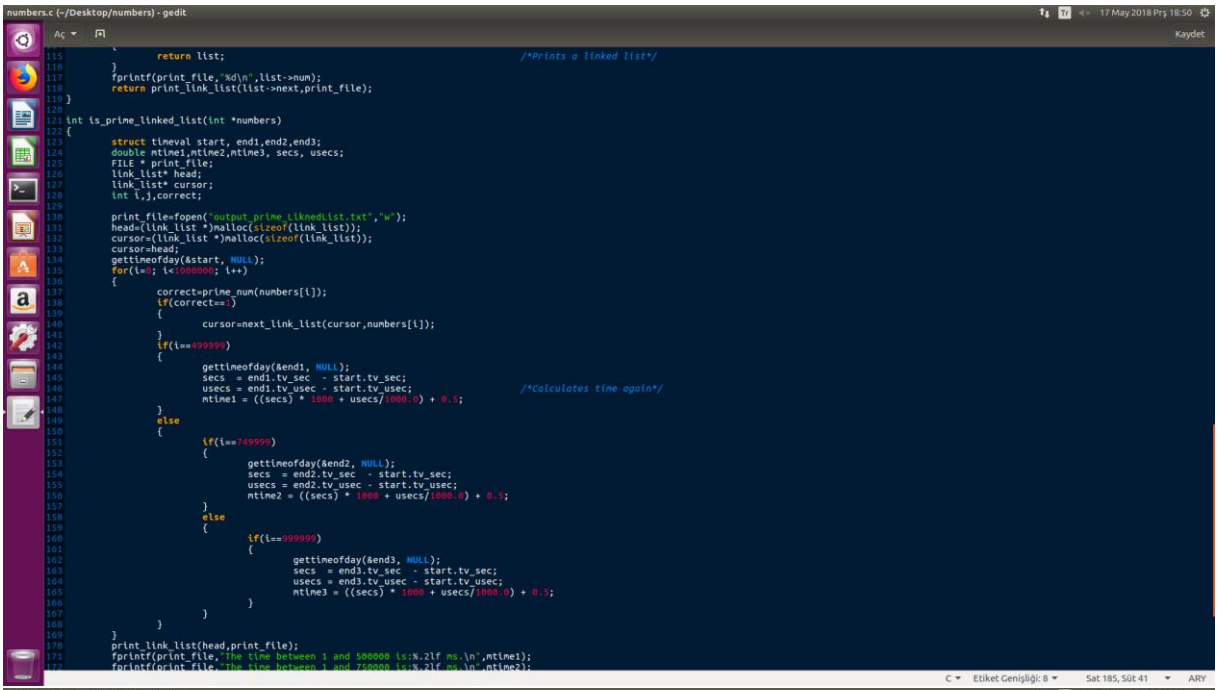
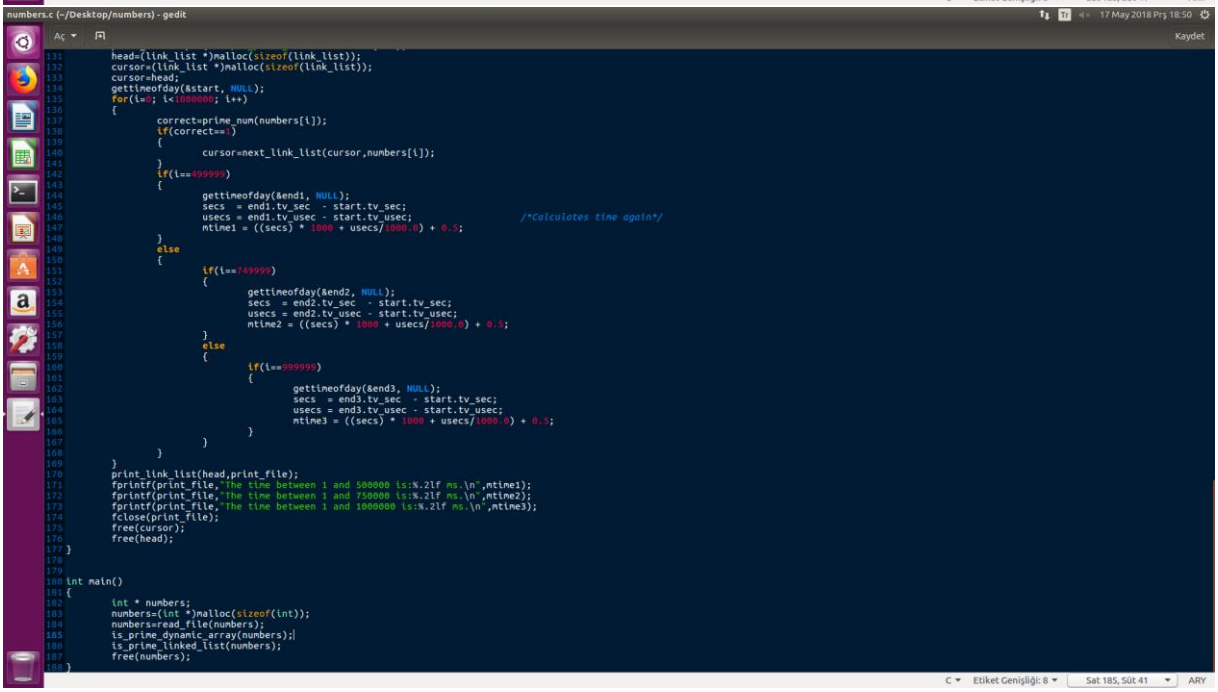


```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

typedef struct link_list{
int num;
struct link_list* next;
} link_list;

int * read_file(int *numbers)
{
        int i;
        int temp;
        FILE * text;

        text=fopen("data.txt","r");
        for(i=0; i<1000000; i++)                                        /*Reads numbers from data.txt and records the numbers in "numbers" dynamic array*/
        {
                numbers=(int *)realloc(numbers,(i+1)*sizeof(int));
                fscanf(text,"%d,%d",&numbers[i],&temp);
        }
        fclose(text);
        return numbers;
}

int prime_num(int num)
{
        int i;
        if(num<2) return -1;

        for(i=2;i<num;i++)
        {
                if(num%i==0)                                            /*Finds the prime numbers*/
                {
                        return -1;
                }

        }
        return 1;
}

int is_prime_dynamic_array(int *numbers)
{
        struct timeval start, end1,end2,end3;
        double mtime1,mtime2,mtime3, secs, usecs;
        FILE * fin;
        int * prime;
        int i,j,correct;

        prime=(int *)malloc(sizeof(int));
        j=0;
        fin=fopen("output_prime_dynamic_array.txt","w");
        gettimeofday(&start, NULL);                             /*Starts the time piece*/
        for(i=0; i<1000000; i++)
        {
                correct=prime_num(numbers[i]);
                if(correct==1)
                {
```



```c
                {
                        prime=(int *)realloc(prime,(j+1)*sizeof(int));
                        prime[j]=numbers[i];
                        fprintf(fin,"%d\n",prime[j]);
                        j++;
                }
                if(i==499999)
                        {
                                gettimeofday(&end1, NULL);
                                secs  = end1.tv_sec  - start.tv_sec;
                                usecs = end1.tv_usec - start.tv_usec;                   /*Ends the time piece for 1 to 500.000 and calculates time*/
                                mtime1 = ((secs) * 1000 + usecs/1000.0) + 0.5;
                        }
                else
                {
                        if(i==749999)
                        {
                                gettimeofday(&end2, NULL);
                                secs  = end2.tv_sec  - start.tv_sec;                    /*Ends the time piece for 1 to 750.000 and calculates time*/
                                usecs = end2.tv_usec - start.tv_usec;
                                mtime2 = ((secs) * 1000 + usecs/1000.0) + 0.5;
                        }
                        else
                        {
                                if(i==999999)
                                {
                                        gettimeofday(&end3, NULL);
                                        secs  = end3.tv_sec  - start.tv_sec;            /*Ends the time piece for 1 to 1.000.000 and calculates time*/
                                        usecs = end3.tv_usec - start.tv_usec;
                                        mtime3 = ((secs) * 1000 + usecs/1000.0) + 0.5;
                                }
                        }
                }
        }
        fprintf(fin,"The time between 1 and 500000 is:%.2lf ms.\n",mtime1);
        fprintf(fin,"The time between 1 and 750000 is:%.2lf ms.\n",mtime2);
        fprintf(fin,"The time between 1 and 1000000 is:%.2lf ms.\n",mtime3);
        fclose(fin);
        free(prime);
        return 0;
}

link_list * next_link_list(link_list * old,int number)
{
        link_list * new;
        new=(link_list *)malloc(sizeof(link_list));
        old->num=number;
        old->next=new;
        new->num=0;                                             /*Creates new linked list*/
        new->next=0;
        return new;
}

link_list * print_link_list(link_list * list,FILE * print_file)
{
        if(list->next==0)
        {
                return list;                                    /*Prints a linked list*/
```


numbers.c (~/Desktop/numbers) - gedit
17 May 2018 Prş 18:50
Aç
Kaydet
return list;
/*Prints a linked list*/
}
fprintf(print_file,"%d\n",list->num);
return print_link_list(list->next,print_file);
}
int is_prime_linked_list(int *numbers)
{
struct timeval start, end1,end2,end3;
double mtime1,mtime2,mtime3, secs, usecs;
FILE * print_file;
link_list* head;
link_list* cursor;
int i,j,correct;
print_file=fopen("output_prime_Liknedlist.txt","w");
head=(link_list *)malloc(sizeof(link_list));
cursor=(link_list *)malloc(sizeof(link_list));
cursor=head;
gettimeofday(&start, NULL);
for(i=0; i<1000000; i++)
{
correct=prime_num(numbers[i]);
if(correct==1)
{
cursor=next_link_list(cursor,numbers[i]);
}
if(i==499999)
{
gettimeofday(&end1, NULL);
secs = end1.tv_sec - start.tv_sec;
usecs = end1.tv_usec - start.tv_usec;
/*Calculates time again*/
mtime1 = ((secs) * 1000 + usecs/1000.0) + 0.5;
}
else
{
if(i==749999)
{
gettimeofday(&end2, NULL);
secs = end2.tv_sec - start.tv_sec;
usecs = end2.tv_usec - start.tv_usec;
mtime2 = ((secs) * 1000 + usecs/1000.0) + 0.5;
}
else
{
if(i==999999)
{
gettimeofday(&end3, NULL);
secs = end3.tv_sec - start.tv_sec;
usecs = end3.tv_usec - start.tv_usec;
mtime3 = ((secs) * 1000 + usecs/1000.0) + 0.5;
}
}
}
}
print_link_list(head,print_file);
fprintf(print_file,"The time between 1 and 500000 is:%.2lf ms.\n",mtime1);
C
Etiket Genişliği: 8
Sat 185, Süt 41
ARY


numbers.c (~/Desktop/numbers) - gedit
17 May 2018 Prş 18:50
Aç
Kaydet
head=(link_list *)malloc(sizeof(link_list));
cursor=(link_list *)malloc(sizeof(link_list));
cursor=head;
gettimeofday(&start, NULL);
for(i=0; i<1000000; i++)
{
correct=prime_num(numbers[i]);
if(correct==1)
{
cursor=next_link_list(cursor,numbers[i]);
}
if(i==499999)
{
gettimeofday(&end1, NULL);
secs = end1.tv_sec - start.tv_sec;
usecs = end1.tv_usec - start.tv_usec;
/*Calculates time again*/
mtime1 = ((secs) * 1000 + usecs/1000.0) + 0.5;
}
else
{
if(i==749999)
{
gettimeofday(&end2, NULL);
secs = end2.tv_sec - start.tv_sec;
usecs = end2.tv_usec - start.tv_usec;
mtime2 = ((secs) * 1000 + usecs/1000.0) + 0.5;
}
else
{
if(i==999999)
{
gettimeofday(&end3, NULL);
secs = end3.tv_sec - start.tv_sec;
usecs = end3.tv_usec - start.tv_usec;
mtime3 = ((secs) * 1000 + usecs/1000.0) + 0.5;
}
}
}
}
print_link_list(head,print_file);
fprintf(print_file,"The time between 1 and 500000 is:%.2lf ms.\n",mtime1);
fprintf(print_file,"The time between 1 and 750000 is:%.2lf ms.\n",mtime2);
fprintf(print_file,"The time between 1 and 1000000 is:%.2lf ms.\n",mtime3);
fclose(print_file);
free(cursor);
free(head);
}
int main()
{
int * numbers;
numbers=(int *)malloc(sizeof(int));
numbers=read_file(numbers);
is_prime_dynamic_array(numbers);
is_prime_linked_list(numbers);
free(numbers);
}
C
Etiket Genişliği: 8
Sat 185, Süt 41
ARY

SATIR 1-24: Dosyadan okumak için bir file pointer ı oluşturdum.

Arrayin dinamik olabilmesi için realloc ile sürekli yeri i+1 arttırdım.

Sonra sayıları oluşturduğum dinamik arraye attım. Ayrıca bu array i maine gönderdim.

SATIR 26-40: Bir sayının asal olup olmadığına bakan fonksiyon yazdım.

SATIR 42-98: Asal sayıları kaydetmek için dinamik array oluşturdum. Bu sırada zaman bölmelerini tanımladım. Döngü başlamadan süreyi başlattım. Sayıları dinamik arraye yazarken 500.000,750.000 ve 1.000.000 da süreyi ayırdım ve başlangıç süresinden çıkardım.

Sonra sayıları ve süreleri text dosyasına bastım.

SATIR 100-109: Bağlı liste yapmak için yeni yer ayırdım ve bunu geri gönderdim.

SATIR 111-119: Bağlı listeyi yazdırmak için recursive döngü tanımladım.

SATIR 121-177: Yine zaman bölmelerini tanımladım. Yeni bir bağlı liste oluşturdum. Listenin başını kaybetmemek için hiç düzenlemediğim ikinci bir liste oluşturdum. (head isimli) Asal sayıları tek tek next_link_list fonksiyonuna gönderdim. Yine süreleri ayırdım. En sonda print_link_list fonksiyonunu çağırarak text dosyasına bastım.

SATIR 180-188: Mainde yeni bir dinamik array başlattım ve bunu bütün fonksiyonlara gönderdim.