Prowadzący zajęcia  
dr inż. Andrzej Urbański

**Laboratoria nr 4**

**Programowanie Niskopoziomowe**

**Pliki**

Wojciech Regulski  
Informatyka(WI) I1  
nr 132312

Zadanie nr 1.

Zliczyć ilość wystąpień zdefiniowanego znaku w pliku wejściowym.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void strupp(char\* beg)

{

while (\*beg = toupper(\*beg))beg++;

}

int main()

{

FILE \*f1;

int l[26],i;

for(i=0; i<26; i++)

l[i]=0;

char temp[100];

if ((f1=fopen("test.txt", "r"))==NULL)

{

printf ("Nie moge otworzyc pliku test.txt do odczytu!\n");

exit(1);

}

while ( !feof(f1) )

{

fgets(&temp,100,f1);

strupp(temp);

for(i=0; i<100; i++)

{

if(temp[i]>=65 && temp[i]<=90)

l[temp[i]-65]++;

}

printf("%s",temp);

}

printf("\n\n");

for(i=0;i<26;i++)

{

printf("%c wystapilo: %d razy\n",i+65,l[i]);

}

return 0;

}

Program przechodzi po każdym znaku, po czym jeśli jest literą, zwiększa ilość ilość wystąpień danej litery o 1 w odpowiadającym indeksie tablicy.Zadanie nr 2.

Napisać kalkulator pozwalający na dodawanie, odejmowanie, mnożenie liczb szesnastkowych.

#include <stdio.h>

#include <stdlib.h>

long hexaToDecimal(char hexadecimal[])

{

long decimalNumber=0;

char hexDigits[] = "0123456789ABCDEF";

int i, j, power=0, digit;

for(i=strlen(hexadecimal)-1; i >= 0; i--)

{

for(j=0; j<16; j++)

{

if(hexadecimal[i] == hexDigits[j])

{

decimalNumber += j\*pow(16, power);

}

}

power++;

}

return decimalNumber;

}

void decimalToHexa(long n)

{

char hexDigits[] = "0123456789ABCDEF";

char hexadecimalNumber[40];

int index=0, remaindar;

while(n != 0)

{

remaindar = n % 16;

hexadecimalNumber[index] = hexDigits[remaindar];

n /= 16;

index++;

}

hexadecimalNumber[index] = '\0';

strrev(hexadecimalNumber);

printf("%s\n",hexadecimalNumber);

}

int main()

{

printf("Prosze uzywac wielkich liter\n");

char a[20];

printf("a(hexadecimal)= ");

fflush(stdin);

gets(a);

char b[20];

printf("b(hexadecimal)= ");

fflush(stdin);

gets(b);

long la=hexaToDecimal(a);

long lb=hexaToDecimal(b);

long wynik = la+lb;

printf("a+b= ");

decimalToHexa(wynik);

wynik = la-lb;

printf("a-b= ");

decimalToHexa(wynik);

wynik = la\*lb;

printf("a\*b= ");

decimalToHexa(wynik);

return 0;

}

Podane liczby hexadecymalne są konwertowane na decymalne, na decymalnych prowadzone są operacje matematyczne, po czym wynik jest z powrotem konwertowany na liczbe hexadecymalną. Zadanie nr 3.

Sortowanie stogowe.

#include <stdio.h>

int max (int \*a, int n, int i, int j, int k) {

int m = i;

if (j < n && a[j] > a[m]) {

m = j;

}

if (k < n && a[k] > a[m]) {

m = k;

}

return m;

}

void downheap (int \*a, int n, int i) {

while (1) {

int j = max(a, n, i, 2 \* i + 1, 2 \* i + 2);

if (j == i) {

break;

}

int t = a[i];

a[i] = a[j];

a[j] = t;

i = j;

}}

void heapsort (int \*a, int n) {

int i;

for (i = (n - 2) / 2; i >= 0; i--) {

downheap(a, n, i);

}

for (i = 0; i < n; i++) {

int t = a[n - i - 1];

a[n - i - 1] = a[0];

a[0] = t;

downheap(a, n - i - 1, 0);

}

}

int main () {

int a[] = {54,13,7,-4,67,333,67};

int n = sizeof a / sizeof a[0];

int i;

for (i = 0; i < n; i++)

printf("%d%s", a[i], i == n - 1 ? "\n" : " ");

heapsort(a, n);

for (i = 0; i < n; i++)

printf("%d%s", a[i], i == n - 1 ? "\n" : " ");

return 0;

}

Prosta implementacja algorytmu.Zadanie nr 4.

Wypisanie najdłuższego i najkrótszego wiersza pliku wejściowego.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

FILE \*f1;

char max[100],min[100];

int iMin=100,iMax=0,i;

if ((f1=fopen("input.txt", "r"))==NULL)

{

printf ("Nie moge otworzyc pliku input.txt do odczytu!\n");

exit(1);

}

while ( !feof(f1) )

{

char temp[100];

//fscanf (f1, "%s", temp);

fgets(temp,100,f1);

if(iMax<strlen(temp))

{

iMax=strlen(temp);

for(i=0;i<100;i++)

max[i]=temp[i];

}

if(iMin>strlen(temp))

{

iMin=strlen(temp);

for(i=0;i<100;i++)

min[i]=temp[i];

}

}

printf("Najdluzszy wiersz: %s \nNajkrotszy wiersz: %s",max,min);

fclose (f1);

return 0;

}

Plik jest odczytywany wiersz po wierszu; gdy jest znaleziony wiersz dłuższy niż aktualnie najdłuższy, to jest przepisywane w jego miejsce. Analogicznie w przypadku najkrótszego.Zadanie nr 5.

Zaimplementuj następujące funkcje występujące w języku AWK: gsub(r,s,t), gensub(r,s,h,t), index(s,t), split(s,a,fs), sub(r,s,t).

void gsub(char \*tekst, char \*wynik, char \*tablica)

{

char \*p = strstr(tablica, tekst);

do

{

if(p)

{

char b[1024];

memset(b,'\0',strlen(b));

if(tablica == p)

{

strcpy(b,wynik);

strcat(b,p+strlen(tekst));

}

else

{

strncpy(b,tablica,strlen(tablica) - strlen(p));

strcat(b,wynik);

strcat(b,p+strlen(tekst));

}

memset(tablica,'\0',strlen(tablica));

strcpy(tablica,b);

}

}while(p && (p = strstr(tablica, tekst)));

}

char \*gensub(char \*tekst, char \*wynik, char \*tablica)

{

char \*finalny = malloc(sizeof(char)\*1024);

char \*p = strstr(tablica, tekst);

do

{

if(p)

{

char b[1024];

memset(b,'\0',strlen(b));

if(tablica == p)

{

strcpy(b,wynik);

strcat(b,p+strlen(tekst));

}

else

{

strncpy(b,tablica,strlen(tablica) - strlen(p));

strcat(b,wynik);

strcat(b,p+strlen(tekst));

}

memset(tablica,'\0',strlen(tablica));

strcpy(finalny,b);

}

}while(p && (p = strstr(tablica, tekst)));

return finalny;

}

int index(char \*tekst, char \*wynik, char \*tablica)

{

char \*p = strstr(tablica, tekst);

int finalny = 0;

int found = 0;

do

{

if(p)

{

char b[1024];

memset(b,'\0',strlen(b));

if(tablica == p)

{

strcpy(b,wynik);

strcat(b,p+strlen(tekst));

}

else

{

strncpy(b,tablica,strlen(tablica) - strlen(p));

strcat(b,wynik);

strcat(b,p+strlen(tekst));

}

memset(tablica,'\0',strlen(tablica));

strcpy(tablica,b);

found = 1;

return finalny;

}

finalny++;

} while(!found && p && (p = strstr(tablica, tekst)));

}

void sub(char \*tekst, char \*wynik, char \*tablica)

{

char \*p = strstr(tablica, tekst);

int found = 0;

do

{

if(p)

{

char b[1024];

memset(b,'\0',strlen(b));

if(tablica == p)

{

strcpy(b,wynik);

strcat(b,p+strlen(tekst));

}

else

{

strncpy(b,tablica,strlen(tablica) - strlen(p));

strcat(b,wynik);

strcat(b,p+strlen(tekst));

}

memset(tablica,'\0',strlen(tablica));

strcpy(tablica,b);

found = 1;

}

} while(!found && p && (p = strstr(tablica, tekst)));

}

int main()

{

return 0;

}Zadanie nr 6.

Załóżmy, że w pliku wejściowym znajdują się wiersze zawierające tylko pola numeryczne. Napisz program, którego zadaniem będzie wypisanie, wartości bezwzględnych dla wszystkich pól znajdujących się w pliku wejściowym.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

FILE \*plik;

if ((plik=fopen("plik.txt", "r"))==NULL)

{

printf ("Blad\n");

return 2;

}

char linia[100];

while(fgets(linia, 100, plik))

{

printf("%d\n", abs(atoi(linia)));

}

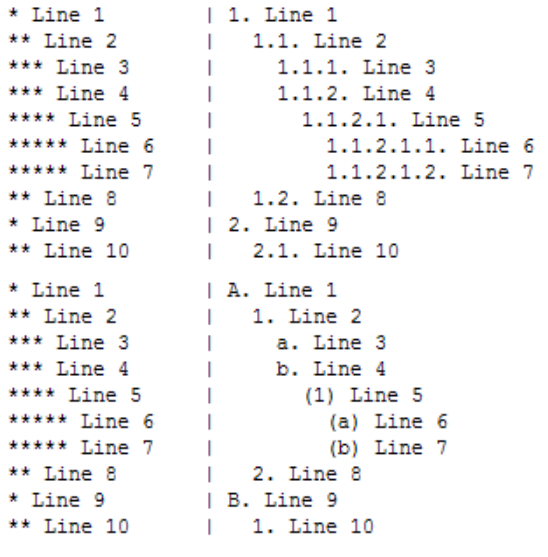
fclose (plik);

return 0;

}

Plik analizowany linia po linii, wypisana zostaje wartość bezwzględna obliczana przez abs(atoi(linia)).Zadanie nr 7.

Napisać programy, które przeprowadzą konwersję plików według poniższych formatów:



#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main()

{

FILE \*f1, \*f2;

int l[5],i,wybor;

for(i=0; i<5; i++)

l[i]=0;

char temp[100];

if ((f1=fopen("test2.txt", "r"))==NULL)

{

printf ("Nie moge otworzyc pliku test2.txt do odczytu!\n");

exit(1);

}

if ((f2=fopen("nowy.txt", "w"))==NULL)

{

printf ("Nie moge otworzyc pliku nowy.txt do zapisu!\n");

exit(1);

}

printf("Wybierz kodowanie (1 lub 2): ");

scanf("%d",&wybor);

switch(wybor)

{

case 1:

{

while ( !feof(f1) )

{

fscanf (f1, "%s", temp);

switch (strlen(temp))

{

case 1:

{

l[1]=l[2]=l[3]=l[4]=0;

//printf("%d. ",++l[0]);

fprintf(f2,"%d. ",++l[0]);

break;

}

case 2:

{

l[2]=l[3]=l[4]=0;

//printf(" %d.%d. ",l[0],++l[1]);

fprintf(f2," %d.%d. ",l[0],++l[1]);

break;

}

case 3:

{

l[3]=l[4]=0;

//printf(" %d.%d.%d. ",l[0],l[1],++l[2]);

fprintf(f2," %d.%d.%d. ",l[0],l[1],++l[2]);

break;

}

case 4:

{

l[4]=0;

//printf(" %d.%d.%d.%d. ",l[0],l[1],l[2],++l[3]);

fprintf(f2," %d.%d.%d.%d. ",l[0],l[1],l[2],++l[3]);

break;

}

case 5:

{

//printf(" %d.%d.%d.%d.%d. ",l[0],l[1],l[2],l[3],++l[4]);

fprintf(f2," %d.%d.%d.%d.%d. ",l[0],l[1],l[2],l[3],++l[4]);

break;

}

}

fgets(&temp,100,f1);

//printf("%s",temp);

fprintf(f2,"%s",temp);

}

break;

}

case 2:

{

while ( !feof(f1) )

{

fscanf (f1, "%s", temp);

switch (strlen(temp))

{

case 1:

{

l[1]=l[2]=l[3]=l[4]=0;

fprintf(f2,"%c. ",65+(l[0]++));

break;

}

case 2:

{

l[2]=l[3]=l[4]=0;

fprintf(f2," %d. ",++l[1]);

break;

}

case 3:

{

l[3]=l[4]=0;

fprintf(f2," %c. ",97+(l[2]++));

break;

}

case 4:

{

l[4]=0;

fprintf(f2," (%d) ",++l[3]);

break;

}

case 5:

{

fprintf(f2," (%c) ",97+(l[4]++));

break;

}

}

fgets(&temp,100,f1);

fprintf(f2,"%s",temp);

}

break;

}

}

fclose (f1);

fclose (f2);

return 0;

}

Program analizuje ilość gwiazdek na początku linii, zarządza numeracją w tablicy pomocniczej, po czym wypisuje odpowiednią numeracje i przepisuje resztę linii.Zadanie nr 8.

Napisz program, którego zadaniem będzie wypisanie statystyk związanych z wartościami numerycznymi znajdującymi się w kolumnach. W skład statystyk dla każdej kolumny ma zostać wypisana suma wszystkich pól tej kolumny oraz średnia związana z daną kolumną.

Plik wejściowy

1 2 3 5 2 0 10 12 2 3 1

Plik wyjściowy

Suma: 1 12 15 7 5 1 Srednia: 0,5 6 7,5 3,5 2,5 1

#include<stdio.h>

#include<math.h>

int main()

{

//Zadanie 8

FILE \*file;

char line, c [10];

int pl[100][2];

int n=0,i,j;

file = fopen("file.txt", "r");

if(file==NULL)

return -1;

while(!feof(file))

{

int k = 0;

line = fgetc(file);

for(i=0;line!='\n' && !feof(file); i++)

{

while(line!='\n' && line!=' ' && !feof(file))

{

c[k] = line;

k++;

line = fgetc(file);

}

for(j=0;j<k;j++)

pl[i][0]+=((c[j]-48)\*pow(10,k-j-1));

k=0;

pl[i][1]++;

if(line==32)

line = fgetc(file);

}

}

printf("Suma: ");

for(i=0;i<n;i++)

printf("%d ", pl[i][0]);

printf("\nSrednia: ");

for(i=0;i<n;i++)

printf("%lf ", (pl[i][0]\*1.0)/(pl[i][1]\*1.0));

fclose(file);

return 0;

}Zadanie nr 9.

Przyjmijmy, że słowo jest to niepusty ciąg znaków różnych od spacji, znaku tabulacji i nowej linii, (zatem ciąg a+b\*c jest jednym słowem). Napisać program obliczania liczby słów w pliku. Wiersze, w których pierwszym znakiem jest średnik nie są brane pod uwagę.

Plik wejściowy

jeden 2 !!! ;

on on on on on on on on

;on on on on on on on on

four ;; six

Plik wyjściowy 6

#include <stdio.h>

#include <stdlib.h>

int main()

{

FILE \*file;

char ln;

int k=0, l=0;

file = fopen("file.txt", "r");

if(file==NULL)

return -1;

while(!feof(file))

{

ln = fgetc(file);

if(ln=='\n')

{

ln = fgetc(file);

if(ln==';')

l=1;

else

{

l=0;

k++;

}

}

else if(l==0 && (ln==' ' || ln=='\t'))

k++;

}

printf("%d", k+1);

fclose(file);

return 0;

}Zadanie nr 10.

Plik wejściowy zawiera ciąg wierszy. W każdym wierszu znajdują się trzy liczby całkowite (o małych wartościach bezwzględnych), oddzielone od siebie dowolną liczbą spacji i znaków tabulacji. Napisać program obliczający dla każdego wiersza sumę liczb w nim zawartych i drukujący zestawienie tabelaryczne uzyskanych wyników w postaci: nagłówek 'SUMA' dla ostatniej kolumny, w kolejnych wierszach, wyrównane w kolumnach do lewej składowe sumy i wartość sumy, oddzielone od siebie znakami '|'.

Plik wejściowy

1 5 18

13 2 -5

Plik wyjściowy

SUMA

| 1 | 5 | 18 | 24

| 13 | 2 | -5 | 10

#include <stdio.h>

#include <stdlib.h>

int main()

{

FILE \*file;

char zn[10];

char ln;

int k=0, l=0;

int i=0, x=0,j,n=0;

file = fopen("file.txt", "r");

if(file!=NULL)

{

while(!feof(file))

{

ln = fgetc(file);

while(ln!='\n' && !feof(file))

{

if(ln=='-')

{

x=1;

ln = fgetc(file);

}

while(ln!='\n' && ln!=' ' && !feof(file))

{

zn[k] = ln;

k++;

ln = fgetc(file);

}

for(j=0;j<k;j++)

{

if(x)

i-=((zn[j]-48)\*pow(10,k-j-1));

else

i+=((zn[j]-48)\*pow(10,k-j-1));

}

printf("|%d", i);

n+=i;

i=k=0;

if(ln==' ') ln = fgetc(file);

}

printf("|%d\n", n);

n=0;

}

}

return 0;

}Zadanie nr 11.

Napisać program kompresujący dane za pomocą algorytmu ByteRun.

void ByteRunCompress(signed char a[], int length)

{

int i = 0;

while (i < length)

{

if ((i < length-1) &&

(a[i] == a[i+1]))

{

int j = 0;

while ((i+j < length-1) &&

(a[i+j] == a[i+j+1]) &&

(j < 127))

{

j++;

}

cout << -j << ", " << (int)a[i+j] << ", ";

i += (j+1);

}

else

{

int j=0;

while ((i+j < length-1) &&

(a[i+j] != a[j+i+1]) &&

(j < 128))

{

j++;

}

if ((i+j == length-1) &&

(j < 128))

{

j++;

}

cout << (j-1) << ", ";

for (int k=0; k<j; k++)

{

cout << (int)a[i+k] << ", ";

}

i += j;

}

}

}

ByteRun kompresuje dane, które występują kilka razy z rzędu, np. sześć ósemek z rzędu zostaje zapisane jako -5, 8. Prosta implementacja tych założeń.