TEST 1 (Daca reusiti sa rezolvati, postati si rezolvarea aici <3 )

1. Se dau doua numere pe 8 biti, a si b. Sa se construiasca un numar pe 32 de biti astfel: primii 16 reprezinta suma a si b, iar urmatorii 16 modulul diferentei.

Rezolvare:

void p1(uint8\_t a, uint8\_t b)

{

uint32\_t mask1 = a+b;

uint32\_t mask2;

if(a>b)

mask2 = a-b;

else

mask2 = b-a;

mask2 = mask2<<16;

mask2 |= mask1;

show\_32bits(mask2);

}

1. Se da un numar pe 32 de biti. Se cere sa se implementeze o functie inverse\_nibble care sa inverseze nibble-urile intre ele, in fiecare portiune de cate 8 biti.

Nr8biti = nibble1nibble2 -> nibble2 nibble1

void inverse\_nibble8(uint8\_t n)

{

uint8\_t mask1 = 0x0F;

uint8\_t mask2 = 0xF0;

uint8\_t n1=0, n2=0;

n1 = n & mask1;

n2 = n & mask2;

n = 0;

n |= (n1<<4) | (n2>>4);

show\_8bits(n);

}

// o varianta

uint32\_t inversare(uint32\_t n, uint32\_t a,uint32\_t b){

n = (((n & a) << 4) | ((n&b) >> 4));

return n;

}

void ex2(){

uint32\_t n = 0,i=0;

uint32\_t aux = 0,a = 0, b = 0;

if(scanf("%d",&n) != 1 ){

perror("Eroare la citire\n");

exit(-1);

}

print\_32(n);

printf("\n");

a = 0xF;

b = 0xF0;

while(i < 4){

aux = (aux | inversare(n,a,b));

a = a << 8;

b = b << 8;

i ++;

}

print\_32(aux);

}

//O alta varianta

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

void afisare\_pe\_32\_biti(uint32\_t n)

{

for(int i=31;i>=0;i--)

{

if(n&(1<<i))

{

printf("1");

}

else

{

printf("0");

}

if((i)%4==0)

{

printf(" ");

}

}

}

uint32\_t inverse(uint32\_t n)

{

uint32\_t mask1=0xF0F0F0F0;

uint32\_t mask2=0x0F0F0F0F;

uint32\_t rezultat;

rezultat=((n&mask1)>>4)|((n&mask2)<<4);

return rezultat;

}

int main()

{

uint32\_t nr,rez;

printf("Introduceti un numar: ");

if(scanf("%d",&nr)!=1)

{

perror("Eroare citire");

exit(-1);

}

rez=inverse(nr);

printf("Numarul initial: ");

afisare\_pe\_32\_biti(nr);

printf("\nNumarul modificat: ");

afisare\_pe\_32\_biti(rez);

return 0;

}

1. Se da un număr intreg. Sa se implementeze o funcție care returnează un număr pe 8 biti fără semn astfel încât pe cel mai nesemnificativ nibble se stochează numărul cifrelor pare din numărul dat ca parametru, iar pe cel semnificativ se stochează numărul total de cifre din număr. Se va afișa numărul obtinut în hexadecimal cu prefixul “0x”.

Exemplu: Input: 1234567

Output: 0x73

int main(){

int x,cifre=0,pare=0;

uint8\_t y;

scanf("%d",&x);

while(x int0){

cifre++;

if(x%2==0) pare++;

x/=10;

}

y= (cifre<<4) | pare;

printf("0x%X",y);

return 0;

}

1. Se citesc numere naturale de la tastatura pana la intalnirea lui 0.Sa se afiseze minimul si media aritmetica a acestor numere.Apoi sa se formeze un numar intreg fara semn pe 32 biti astfel incat pe cei mai semnificativi 16 biti sa se afle media aritmetica ,iar pe cei mai putini semnificativi 16 biti sa se afle minimul. Pentru a calcula minimul dintre 2 numere se foloseste o functie ce primeste 2 parametrii si returneaza minimul.

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <string.h>

dif

int mini (int minim, int a){int

if (minim > a)

minim = a;

return minim;

}

void print\_by\_32(int32\_t n){

printf("%u\n", n);

uint32\_t mask = 0xrea 80000000;

for (int i = 1; i <=32; i++){

if ((n & mask) == 0)

printf("0");

else

printf("1");

mask >>= 1;

if (i % 4 == 0)

printf(" ");

}

printf("\n");

}

void formator(int16\_t minim, int16\_t media){

uint32\_t copy = media;

copy = copy << 16;

copy = copy | minim;

print\_by\_32(copy);

printf("n = %u\n", copy);

}

int main (){

int16\_t a = 0, suma = 0, nr = 0, minim = 0, media;

scanf("%hd", &a);

minim = a;

while (a != 0){

nr++;

suma+=a;

minim = mini(minim, a);

scanf("%hd", &a);

}

media = suma / nr;

print\_by\_32(minim);

print\_by\_32(media);

printf("Minimul: %d\nMedia aritmetica: %d\n", minim, media);

formator(minim, media);

return 0;

}

1. Sa se scrie o functie decode care primeste 2 parametrii: un nr n pe 32 de biti si un nr natural p. Nr n reprezinta data unui an scrisa in felul urmator: primii 4 biti de la dreapta (LSB) reprezinta ziua, urmatorii 4 reprezinta luna, iar restul pana la 32 reprezinta anul. Functia returneaza (in decimal):

* ziua, daca p=0
* luna, daca p=1
* anul, daca p=2
* 0, daca p>2

#include<stdio.h>

#include<stdint.h>

int decode(uint32\_t n,int p)

{

uint8\_t ziua=n&0x1F,luna=(n>>5)&0xF;

uint32\_t anul=n>>9;

switch(p)

{

case 0:

{

return ziua;

bre ak;

}

case 1:

{

return luna;

break;

}

case 2:

{

return anul;

break;

}

default:

{

return 0;

break;

}

}

}

int main(void)

{

uint32\_t n;

int p;

scanf("%d %d",&n,&p);

printf("%d\n",decode(n,p));

return 0;

}

1. Sa se scrie o functie care primeste un nr pe 16 biti fara semn, un n si un p si ii inverseaza numarului n biti incepand de pe pozitia p.

int main(){

uint16\_t nr,mask=0;

int n,p;

scanf("%hd %d %d",&nr,&n,&p);

if((n+p>15) || (n<0) || (p<0)){

printf("Date gresite");

exit(-1);

}

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

mask = mask<<1;

mask = mask | 1;

}

nr=nr ^ (mask<<p);

printare\_16biti(nr);

return 0;

}

//varianta 2

void showbits16(uint16\_t n)

{

uint16\_t m=1;

m<<=15;

for(int i=0; i<=15; i++)

{

if((n&m)!=0)

printf("1");

else

printf("0");

m>>=1;

if(i%4==3)

printf(" ");

}

printf("\n");

}

void inversare(uint16\_t n, int p)

{

uint16\_t mask=0;

for(int i=0; i<=p; i++)

{

mask=mask|(((n>>i)&1)<<(p-i));

}

uint16\_t masque=n;

for(int i=0; i<=p; i++)

{

masque=masque&((n>>i)<<i);

}

masque=masque|mask;

showbits16(masque);

}

int main(void)

{

uint16\_t n;

printf("n= "); scanf("%hd", &n);

showbits16(n);

int p;

printf("p= "); scanf("%d", &p);

inversare(n, p);

return 0;

}

1. Sa se scrie o functie care primeste un numar n pe 32 de biti si returneaza un numar pe 64 de biti format din nibble-urile fiecarei cifre din numarul n. Exemplu : 123 -> 0001 0010 0011

uint32\_t n = 0,i = 0;

uint64\_t new = 0;

scanf(“%ld”,&new);

while(n > 0){

new = new | ((n%10) << i);

i += 4;

n = n/10;

}

print\_64(new);

TEST 2

1. Se citesc 2 vectori A si B dintr-un fisier pana la EOF. Sa se realizeze o functie de citire din fisier cu 3 parametri(vectorul A, vectorul B, dimensiunea comuna). Apoi, toate numerele din vectorul A care au paritate para pe biti (adica daca au un nr par de biti de 1) sa fie adaugate la vectorul B, avand grija sa nu fie depasita memoria, iar la final sa se afiseze vectorul B.

emacs

void citesteDinFisier(int v1[], int v2[], int \*count) {

FILE \*fisier;

fisier = fopen("test.txt", "r");

if (fisier == NULL) {

printf("fisierul nu s-a deschis\n");

return;

}

\*count = 0;

while (fscanf(fisier, "%d", &v1[\*count]) != EOF) {

printf("Am citit: %d\n", v1[\*count]);

(\*count)++;

}

fclose(fisier);

}

int returneazaNrBiti(int a){

int count=0;

while(a!=0){

int musk=1;

if( (a&musk) ==1 ){

count++;

}

a>>=1;

}

return count;

}

void problema1(int v1[] , int v2[] , int n ){

int nou=0;

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

if(returneazaNrBiti(v1[i]) % 2 == 0){

v2[nou] = v1[i];

nou++;

}

}

for(int i = 0 ; i < nou ; i++){

printf("%d--\n" , v2[i]);

}

}

int main(void){

int v1[100];

int v2[100];

int count;

citesteDinFisier(v1,v2,&count);

problema1(v1,v2,count);

return 0;

}

1. Se dau 2 tablouri pe 32 de biti. Se adauga in tabloul B numerele din tabloul A care sunt pare pe biti. Programul se va testa prin redirectare de fisier si nu prin numere citite de la tastatura.

#include <stdio.h>

#include <stdint.h>

#define MAX 10

int check\_parity(int n){

uint32\_t mask = 0x80000000;

uint8\_t nr = 0;

for (int i = 1; i <= 32; i++){

if ((n & mask) != 0)

nr++;

mask = mask >> 1;

}

printf("nr = %hhu \n\n", nr);

if (nr % 2 == 0 && nr != 0)

return 1;

return 0;

}

void print\_by\_32(int32\_t n){

uint32\_t mask = 0x80000000;

for (int i = 1; i <=32; i++){

if ((n & mask) == 0)

printf("0");

else

printf("1");

mask >>= 1;

if (i % 4 == 0)

printf(" ");

}

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

}

int main (void){

int A[MAX], B[MAX], i = -1, j = -1;

while (scanf("%d", &A[++i]) == 1){

print\_by\_32(A[i]);

if (check\_parity(A[i]) == 1)

B[++j] = A[i];

}

printf("B = ");

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

printf("%d ", B[k]);

}

printf("\n");

return 0;

}

1. Se cere sa se scrie o funcie care primeste un vector si dimensiunea sa ca parametri si numara cate numere impare se afla pe pozitii pare. Vectorul se populeaza cu numere redirectate dintr-un fisier si se salveaza pana la EOF sau pana ce nu mai este loc in vector (se hardcodeaza o dimenziune cu #define).

#define SIZE 10

int p3(int v[], int n){

int k=0,i=0;

for(i=0;i<n;i+=2){

if(v[i]%2==1)

k++;

}

return k;

}

int main(){

int v[SIZE],i=0;

while(v[i] != EOF && i<SIZE){

scanf("%d", (v+i));

i++;

}

printf("%d\n",p3(v,i-1));

}

varianta 2:

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#define MAX 1000

int functie(int vector[],int size)

{

int k=0;

for(int i=0;i<size;i++)

{

if(i%2==0 && vector[i]%2==1)

k++;

}

return k;

}

void populeazavector(int vector[],int \*size,char \* caleFisier)

{

FILE \*f=NULL;

int x=0;

//1.deschidem fisierul

if((f=fopen(caleFisier,"r")) == NULL)

{

perror(NULL);

exit(-1);

}

while(fscanf(f,"%d",&x) == 1 && \*size<MAX)

{

vector[\*size]=x;

(\*size)++;

}

//2.inchidem fisierul

if(fclose(f) !=0 )

{

perror(NULL);

exit(-1);

}

}

int main()

{

int v[MAX];

int size=0;

populeazavector(v,&size,"numere.txt");

int rezultat = functie(v,size);

printf("%d",rezultat);

return 0;

}

1. Se citeste un vector de numere intregi fara semn pe 16 biti de dimensiune nedefinita,si se formeaza un numar intreg fara semn pe 32 biti care are pe cei mai semnificativi 16 biti cel mai mare numar impar din vector,iar pe restul 16,pozitia din vector la care se gaseste cel mai mare numar impar.Rezolvarea se face folosind doar pointeri(fara indecsi).Se trateaza situatiile in care dimensiunea vectorului este prea mare.(se stabilieste o limita).

#include <stdio.h>

#include <stdint.h>

#define MAX 100

void print\_by\_32(uint32\_t n){

printf("n = %u\n", n);

uint32\_t mask = 0x80000000;

for (int i = 1; i <=32; i++){

if ((n & mask) == 0)

printf("0");

else

printf("1");

mask >>= 1;

if (i % 4 == 0)

printf(" ");

}

printf("\n");

}

uint32\_t formator (uint16\_t n, uint16\_t pos){

uint32\_t res = 0;

res = n;

res = res << 16;

res = res | pos;

return res;

}

int main (void){

uint16\_t v[MAX], minim = 0;

uint32\_t new = 0, pos = 0;

uint8\_t i = 1;

scanf("%hu", v);

minim = \*v;

while (scanf("%hu", v + i) == 1){

if (\*(v + i) < minim){

minim = \*(v + i);

pos = i;

}

i++;

}

new = formator(minim,pos);

print\_by\_32(pos);

print\_by\_32(minim);

print\_by\_32(new);

return 0;

}

//varianta 2

#define MAX 30

void showbits32(uint32\_t n)

{

uint32\_t m=1;

m<<=31;

for(int i=0; i<=31; i++)

{

if((n&m)!=0)

printf("1");

else

printf("0");

m>>=1;

if(i%4==3)

printf(" ");

}

printf("\n");

}

void showbits16(uint16\_t n)

{

uint16\_t m=1;

m<<=15;

for(int i=0; i<=15; i++)

{

if((n&m)!=0)

printf("1");

else

printf("0");

m>>=1;

if(i%4==3)

printf(" ");

}

printf("\n");

}

int main(void)

{

uint16\_t v[MAX], maxim=0, poz;

int i=1;

while(scanf("%hu", v+i)==1)

{

printf("v[%d]: %d - ", i, \*(v+i));

showbits16(\*(v+i));

if(\*(v+i)>maxim && \*(v+i)%2!=0)

{

maxim=\*(v+i);

poz=i;

}

i++;

}

if(maxim==0)

printf("nu exista numere impare in vector\n");

else

{

printf("maximul este: %d - ", \*(v+poz));

showbits16(maxim);

printf("pozitie maxim: %d - ", poz);

showbits16(poz);

}

uint32\_t new=0;

new|=maxim;

new<<=16;

new|=poz;

printf("numarul nou format: ");

showbits32(new);

return 0;

}

1. Se da un fișier. Scrieți o functie care sa afle numărul maxim impar și poziția sa și sa se afișeze. Funcția trebuie sa returneze un număr pe 32 de biți în care pe cei 16 cei mai semnificativi biți este numărul maxim impar, iar pe cei 16 cei mai puțin semnificativi biți este poziția acestuia.

//rezolvarea, mai sus#

1. Se citește un vector de numere intregi dintr-un fisier. Se cere sa se elimine din vector numerele divizibile cu 3 fără a modifica ordinea elementelor. Se va utiliza doar aritmetica cu pointeri (fara indecși). Dimensiunea maxima a vectorului va fi data printr-un macro(#define SIZE), iar pentru testare se va folosi redirectarea standard. Se vor folosi funcții pentru citire, prelucrare și afisare.

#define SIZE 10

int citire(int \*v){

int i=0,n=0;

while((scanf("%d",v+i))==1 && i<SIZE){

i++;

n++;

}

return n;

}

int eliminare(int \*v,int n){

int i=0,j=0;

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

if(\*(v+i)%3==0){

n--;

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

\*(v+j)=\*(v+j+1);

i--;

}

}

return n;

}

void afisare(int \*v,int n){

int i=0;

printf("\n");

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

printf("%d ",\*(v+i));

printf("\n");

}

int main(){

int v[SIZE+1],n;

n=citire(v);

n=eliminare(v,n);

afisare(v,n);

return 0;

}

1. Să se implementeze o funcție care primeşte ca parametru un tablou de întregi pe 16 biti A, un tablou de numere întregi pe 16 biti B şi un întreg N reprezentând numărul de elemente. Cele două tablouri vor avea acelaşi număr de elemente N. Funcţia va adauga în tabloul B numerele din tabloul A ce au o paritate para pe biți. Funcţia va returna dimensiune utilá (nouă) a tabloului B. Tabloul A nu va fi modificat si parametrul va fi adaptat în acest sens.

Se va testa funcţia printr-un program ce va citi datele unui tablou de la stdin (intrarea standard).

Testarea programului se va realiza prin introducere de date de la tastatură până la EOF şi prin

redirectarea stdin dintr-un fisier.

NU se va citi numarul de elemente din tablou de la intrarea standard (stdin, tastatura)

Pentru adresarea elementelor din tablou se va folosi doar operatorul de indexare.

Fisierul de test poate fi descărcat folosind urmatoarea comanda.

wget http://staff.cs.upt.ro/~valy/pt/nr.txt

Dimensiunea maximă a tabloului va fi constantă, configurabilă printr-un macro (define). Se vor considera și trata situațiile în care dimensiunea tabloului este mai mare sau mai mică decât numărul de elemente citite de la stdin.

TEST 3

1. Sa se implementeze cifrul lui cezar. Se va citi de la tastatura sau dintr-un fisier un sir de caractere care va fi alocat dinamic si un numar care va fi "cheia" cifrului. Functia va inlocui fiecare litera a sirului initial cu litera din alfabet care se afla la o distanta egala cu cheia. Daca litera curenta adunata cu cheia trece de litera z atunci numaratoarea va continua de la inceputul alfabetului.

Ex: Pentru sirul "Ana are mere" si cheia 3 se obtine "Dqd duh phuh"

Pentru sirul "Zebra" si cheia 2 se obtine "Bgdtc"

char \*cifrulcezar(char \*string,int cheie)

{

char \*result=malloc(sizeof(char)\*(strlen(string)+1));

if(result==NULL)

{

perror("eroare");

exit(-1);

}

for(int i=0;i<strlen(string);i++)

{

result[i]=string[i]+cheie;

}

result[strlen(string)] = '\0';

return result;

}

//rezolvarea 2

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#define chunk 128

char \*cifrul\_lui\_cezar(char \*string,int nr)

{

char \*result=malloc(sizeof(char)\*(strlen(string)+1));

if(result==NULL)

{

perror("Eroare alocare memorie");

exit(-1);

}

for(int i=0;i<strlen(string);i++)

{

if(isalpha(string[i])){

result[i]=(string[i]-97+nr)%26+97;

}

else

{

result[i]=string[i];

}

}

result[strlen(string)]='\0';

return result;

}

int main()

{

FILE \*fin=fopen("fisier.txt","r");

if(fin==NULL)

{

perror("error");

exit(-1);

}

char \*string= malloc(sizeof(char)\*(chunk));

int i=0,current\_size=chunk,n=0;

char c;

while(((c=fgetc(fin))!=EOF) && (c!='\n'))

{

string[i]=c;

i++;

if(i==(current\_size))

{

current\_size+=chunk;

if((string=realloc(string,sizeof(char)\*(current\_size)+1))==NULL)

{

printf("eroare");

exit(-1);

}

}

}

string[i]='\0';

printf(“Introduceti n:”);

scanf(“%d”,&n);

char \*rezultat=cifrul\_lui\_cezar(string,n);

printf("\nString-ul rezultat este: %s\n",rezultat);

free(rezultat);

free(string);

fclose(fin);

return 0; }

1. Sa se scrie o functie care primeste ca argumente 2 fisiere: source si destination. sa se scrie in fisierul destination suma numerelor gasite in textul din fisierul source si apoi continutul fisierului fara acele numere. antentul functiei: void addNumbers(char \*source, char \*dest). Sa se scrie folosind minimul de memorie, intr-un mod eficient.

input source: Ana7 a4re mere9.

output dest: 20

Ana are mere.

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

void addNumbers(char \*source, char \*dest)

{

FILE \*fin = NULL;

FILE \*fout = NULL;

if((fin = fopen(source, "r")) == NULL)

{

perror("eroare deschidere");

exit(-1);

}

if((fout = fopen(dest, "w")) == NULL)

{

perror("eroare deschidere");

exit(-1);

}

char c;

int suma = 0;

int nr = 0;

// formam doar suma

while((c = fgetc(fin)) != EOF)

{

if(isdigit(c))

{

nr = c - '0' + (nr \* 10); // ducem val c la valoarea numarului 0-9

}

else

{

// nr s-a terminat si adaugam nr la suma

suma +=nr;

nr = 0; // luam alt nr

}

}

// am terminat de facut suma

if(fclose(fin) != 0)

{

perror("eroare inchidere");

exit(-1);

}

fprintf(fout, "%d\n", suma);

// redeschidem ca sa luam textul fara nr

if((fin = fopen(source, "r") == NULL)

{

perror("eroare deschidere");

exit(-1);

}

while((c = fgetc(fin)) != EOF)

{

if(!isdigit(c))

{

fputc(c, fout);

}

}

if(fclose(fin) != 0)

{

perror("eroare inchidere");

exit(-1);

}

if(fclose(fout) != 0)

{

perror("eroare inchidere");

exit(-1);

}

}

int main(int argc, char \*argv[])

{

if(argc != 3)

{

perror("eroare argumente");

exit(-1);

}

addNumbers(argv[1], argv[2]);

return 0;

}

3.Se cere sa se scrie o functie care prelucreaza un sir de caractere astfel: fiecare caracter care nu e litera se inlocuieste cu valoarea sa din tabelul ASCII in format hexazecimal intre paranteze patrate si precedat de 0x. Sirul se aloca dinamic in funcia de prelucrare.

EX: “ana are mere, pere!” => “ana[0x20]are[0x20]mere[0x2C][0x20]pere[0x21]”

Rezolvare 3:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

char \*modificare(char \*s)

{

char \*s2=NULL,aux[7]="";

int cnt=0,len=0,adder=0;

for(int i=0;i<strlen(s);i++)

{

if (isalpha(s[i])==0)

{

cnt++;

}

}

len=strlen(s);

if((s2=malloc((len+5\*cnt+1)\*sizeof(char)))==NULL)

{

perror(NULL);

exit(-1);

}

for(int i=0;i<len;i++)

{

if (isalpha(s[i]))

{

s2[i+adder]=s[i];

}

else

{

s2[i+adder]='\0';

sprintf(aux,"[0x%X]",s[i]);

strcat(s2,aux);

adder+=5;

}

}

return s2;

}

int main(void)

{

char s[200],\*s2=NULL;

if(fgets(s,200,stdin)!=NULL)

{

if(s[strlen(s)-1]=='\n')

{

s[strlen(s)-1]='\0';

}

s2=modificare(s);

}

printf("%s\n",s2);

free(s2);

return 0;

}

4. Sa se scrie o implementare proprie a functiei strcat (denumita mystrcat) din biblioteca standard string.h cu diferenta ca rezultatul concatenarii este returnat ca un string alocat dinamic. Pentru implementare, se interzice utilizarea oricarei functii standard de prelucrare de string-uri din biblioteca string.h. Pentru simplificarea acestei cerinte se recomanda nici sa nu se includa deloc in program biblioteca string.h. Functia se va testa prin string-uri citite de la tastatura. Programul se va termina fara memory leaks. Se vor verifica si testa corespunzator toate cazurile de eroare posibile.

Rezolvare 4:

#include<stdio.h>

#include<stdlib.h>

char \*mystrcat(char \*s1,char \*s2)

{

char \*s=NULL;

int len1=0,len2=0,i=0;

while(s1[len1]!='\0')

{

len1++;

}

while(s2[len2]!='\0')

{

len2++;

}

if(s1[len1-1]=='\n')

{

s1[len1-1]='\0';

len1=len1-1;

}

if(s2[len2-1]=='\n')

{

s2[len2-1]='\0';

len2=len2-1;

}

if((s=malloc((len1+len2+1)\*sizeof(char)))==NULL)

{

perror(NULL);

exit(-1);

}

for(;i<len1+len2;i++) //i-ul a fost initializat mai sus

{

if (i<len1)

{

s[i]=s1[i];

}

else

{

s[i]=s2[i-len1];

}

}

s[i]='\0';

return s;

}

int main()

{

char s1[100],s2[100],\*s=NULL;

if((fgets(s1,100,stdin)!=NULL) && (fgets(s2,100,stdin)!=NULL))

{

s=mystrcat(s1,s2);

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

free(s);

}

return 0;

}

5. Se da fisierul “random\_text\_numbers.txt” si se cere sa se scrie intr-un alt fisier pe primul rand suma tuturor numerelor intregi din fisier, urmat de textul ramas prin eliminarea numerelor.Se puncteaza citirea eficienta din fisier.

#define CHUNK 100

char\* citire(char \*s,FILE \*f){

char c[CHUNK];

int i=2;

char \*aux;

while((fgets(c,CHUNK,f))!=NULL){

s=strcat(s,c);

if(strlen(c)<CHUNK-1){

break;

}

if((aux=(char\*)realloc(s,i\*CHUNK\*sizeof(char)))==NULL){

printf("Memorie insuficienta");

free(s);

exit(-1);

}

s=aux;

i++;

}

return s;

}

int eliminare(char \*s){

int i,j,n=strlen(s),k=0;

char aux;

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

if(strchr("1234567890",s[i])){

aux=s[i+1];

s[i+1]='a';

k+=atoi(&s[i]);

s[i+1]=aux;

n--;

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

s[j]=s[j+1];

i--;

}

}

return k;

}

int main(){

char \*s;

FILE \*f,\*g;

int k;

if((f=fopen("random\_text\_numbers.txt","r"))==NULL){

printf("Eroare la deschiderea fisierului\n");

exit(-1);

}

if((g=fopen("output\_file.txt","w"))==NULL){

printf("Eroare la deschiderea fisierului\n");

exit(-1);

}

if((s=(char\*)malloc(CHUNK\*sizeof(char)))==NULL){

printf("Memorie insuficienta\n");

exit(-1);

}

s[0]=0;

s=citire(s,f);

fclose(f);

k=eliminare(s);

fprintf(g,"%d\n%s",k,s);

free(s);

fclose(g);

return 0;

}

6. Fie un oras de extraterestri. Un oras are: nume si un vector alocat dinamic cu extraterestri. Fiecare extraterestru are: nume (maxim 20 de caractere), culoare (rosu, verde sau albastru), data nasterii (zi, luna si an) si daca este sau nu, membru in Senatul Extraterestrilor. Sa se implementeze cat mai abstract toate structurile de date din problema si sa se scrie o functie care primeste ca parametru un oras de extraterestrii si returneaza un pointer la un alt oras de extraterestrii care contine aceeasi extraterestrii ca cel primit ca parametru, sortati in ordine alfabetica.

Se va testa programul cu date hard-codate (setate in codul sursa), sau prin citire din fisier/tastatura.

7. Intr-un fișier se scriu datele când concurenții termina o cursa. În alt fișier se afișează pe prima linie timpul cel mai rapid în care cineva termina cursa, iar pe restul liniilor diferența de timp. (nu mai știu sigur dacă în exmplul dat la test componentele din timp se luau cu 0 în față dacă erau mai mici decat 10 sau nu)

Ex :

fișier.in

04:05:31

03:02:01

10:18:43

fișier.out

03:02:01

+ 01:03:30

+ 07:16:42

8. Se da o functie care primeste ca parametrii: calea catre un fisier de intrare, calea catre un fisier de iesire si un nr n. Sa se afiseze in fisierul de iesire caracterele de pe pozitiile 0, 1•n, 2•n, 3•n, … din fisierul de intrare.

void functie(char \*input,char \*output,int n)

{

FILE \*fin=fopen(input,"rt");

if(fin==NULL)

{

perror("eroare");

exit(-1);

}

FILE \*fout=fopen(output,"w");

if(fout==NULL)

{

perror("eroare");

exit(-1);

}

int poz=0;

int c=0;

while((c=fgetc(fin))!=EOF)

{

if(poz%n==0)

{

fprintf(fout,"%c",c);

}

poz++;

}

if(fclose(fin)!=0)

{

perror("eroare");

exit(-1);

}

if(fclose(fout)!=0)

{

perror("eroare");

exit(-1);

}

}

9. Să se implementeze o funcție transform(const char \*srcfile, const char \*destfile) care primeste ca parametrii cai către un fișier de intrare și un fisier de ieșire. Programul citește din fisierul de intrare numere pe 32 de biti fără semn în format hexazecimal și afișează în fisierul de ieșire numerele transformate în binar, cu prefixul “0b”. Se vor folosi cât mai multe funcții în implementarea programului. Se va testa programul cu fisierul de intrare “nr\_hex.txt”.

Exemplu: 1A2B3546-> 0b 0001 1010 0010 1011 0011 0101 0100 0110

void transform(char \*input,char \*output)

{

FILE \*fin=fopen(input,"rt");

if(fin==NULL)

{

perror("eroare");

exit(-1);

}

FILE \*fout=fopen(output,"w");

if(fout==NULL)

{

perror("eroare");

exit(-1);

}

int buf=0;

while(fscanf(fin,"%X",&buf)>0)

{

fprintf(fout,"%b\n",buf);

}

if(fclose(fin)!=0)

{

perror("eroare");

exit(-1);

}

if(fclose(fout)!=0)

{

perror("eroare");

exit(-1);

}

}

ex: la ^masa sta Laura La ^masa Sta Laura

in Oras #ste O cASa In Oras #ste O CASa

#define CHUNK 1000

void transform(char \*input,char \*output)

{

FILE \*fin=fopen(input,"rt");

if(fin==NULL)

{

perror("eroare");

exit(-1);

}

FILE \*fout=fopen(output,"w");

if(fout==NULL)

{

perror("eroare");

exit(-1);

}

char linie[CHUNK];

int prim=0;

while(fgets(linie,CHUNK,fin)!=NULL)

{

prim=0;

for(int i=0;i<strlen(linie);i++)

{

if(((linie[i-1]==' ') || prim==0) && (isalpha(linie[i])!=0))

{

fprintf(fout,"%c",toupper(linie[i]));

prim=1;

}

else

{

fprintf(fout,"%c",linie[i]);

}

}

}

if(fclose(fin)!=0)

{

perror("eroare");

exit(-1);

}

if(fclose(fout)!=0)

{

perror("eroare");

exit(-1);

}

}

11. Sa se încerce problema de mai sus și pentru transformarea tuturor literelor mici în litere mari, indiferent de poziția lor în șir (nu mai este necesar să transformăm doar prima literă)

12. Se citesc de la tastatura 2 stringuri. Sa se scrie o functie care returneaza un string alocat dinamic format din caracterele comune din cele 2 stringuri luate o singura data alfabetic ordonate. Ex: s1 = Andrei s2 = anais -> s = ain

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void alphabetical (char \*s){

char aux = '\0';

for (int i = 0; i < strlen(s) - 1; i++){

for (int j = i + 1; j < strlen(s); j++)

if (s[i] > s[j]){

aux = s[i];

s[i] = s[j];

s[j] = aux;

}

}

}

const char\* mutual (const char \*s, const char \*d){

int app = 0, j = 0;

char \*rez = NULL;

int is\_double = 0;

char poz[strlen(d)];

strcpy(poz, d);

if ((rez = (char \*) malloc (sizeof(char))) == NULL){

fprintf(stderr, "Eroare la alocarea memoriei pentru string-ul cu caractere comune.\n");

exit(EXIT\_FAILURE);

}

for (int i = 0; i < strlen(s); i++){

app = 0;

strcpy(poz, d);

is\_double = 0;

if (strchr(poz, s[i]) != NULL)

app = 1;

if (app == 1){

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

if (rez[k] == s[i])

is\_double = 1;

}

if (is\_double == 0){

rez[j++] = s[i];

if ((rez = (char \*) realloc (rez, j+1)) == NULL){

fprintf(stderr, "Eroare la adaugarea de memorie pentru string-ul cu caractere comune.");

exit(EXIT\_FAILURE);

}

}

}

}

rez[j] = '\0';

return rez;

}

int main (void){

char s[256], d[256];

fgets(s, 256, stdin);

fgets(d, 256, stdin);

char \*result = NULL;

if ((result = (char \*) malloc (strlen(mutual(s, d)))) == NULL){

fprintf(stderr, "Eroare la alocarea memoriei.\n");

exit(EXIT\_FAILURE);

}

strcpy(result, mutual(s, d));

result[strlen(result) - 1] = '\0';

if (result[0] != '\0'){

alphabetical(result);

printf("Caracterele comune o singura data in ordine alfabetica: %s\n", result);

}

else {

printf("Nu exista caractere comune.\n");

}

free(result);

return 0;

}

13. Un astronaut este caracterizat de urmatoarele:

* nume
* data nasterii (zi, luna, an)
* are sau nu rang de ofiter
* culoare costum spatial (poate fi una din alb, negru, gri)

Nava spatiala este caracterizata de urmatoarele:

* nume
* cod unic de identificare
* nume planeta de origine
* nume galaxie de origine
* vector de pointeri catre astronauti alocat dinamic

Sa se scrie o functie ce primeste o nava spatiala si returneaza o nava spatiala ce are astronautii sortati in ordine alfabetica dupa nume.

Atentie la alegerea tipurilor de date!

Se va testa prin date introduse de la tastatură sau cu date hard-codate.

Un fisier contine pe fiecare linie un numar. Sa se aloce dinamic un vector ce contine toate numerele din fisier si sa se afiseze, sa se ordoneze descrescator si sa se afiseze, apoi sa se afiseze, tot descrescator, fiecare numar scris pe 4 biti. sa se foloseasca argumente in linie de comanda

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define CHUNK 8

#define MAX 100

void showBits (uint8\_t nr)

{

uint8\_t mask = 0x80; // 0b1000000000000000

uint8\_t i = 0;

for (i = 0; i < 8; i++)

{

if ((nr & mask) == 0)

{

printf ("0");

}

else

{

printf ("1");

}

mask = mask >> 1; // mask >>= 1;

}

printf ("\n");

}

int \*functie(int \*size)

{

FILE \*f=NULL;

if((f=fopen("txt.in","r")) == NULL)

{

perror(NULL);

exit(-1);

}

int ch=0;

int \*array;

int index=0;

int current\_size=0;

current\_size=CHUNK;

if((array=malloc(current\_size\*sizeof(int))) == NULL)

{

perror(NULL);

exit(-1);

}

while(fscanf(f,"%d",&ch) != EOF)

{

array[index++]=ch;

if(index == current\_size)

{

current\_size=current\_size + CHUNK;

if((array=realloc(array,current\_size\*sizeof(int))) == NULL)

{

perror(NULL);

exit(-1);

}

}

}

if((array=realloc(array,current\_size\*sizeof(int))) == NULL)

{

perror(NULL);

exit(-1);

}

if(fclose(f)!=0)

{

perror(NULL);

exit(-1);

}

\*size=index;

return array;

}

void modificare(int \*array, int size)

{

int i=0;

int aux;

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

{

for(int j=i+1; j<size; j++)

{

if(array[i]<array[j])

{

aux=array[i];

array[i]=array[j];

array[j]=aux;

}

}

}

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

{

printf("%d ",array[i]);

}

printf("\n");

}

int main(void)

{

int \*array=NULL;

int size=0, i;

array=functie(&size);

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

{

printf("%d ",array[i]);

}

printf("\n");

modificare(array,size);

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

{

showBits(array[i]);

}

printf("\n");

free(array);

return 0;

}

Sa se aloce dinamic un vector ce contine numere random. Sa se transforme vectorul de tip int intr-un string cu elemente de tip char.

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define CHUNK 8

#define MAX 100

int \*functie(int \*size)

{

int aux;

int \*array;

int index=0;

int current\_size=0;

current\_size=CHUNK;

srand(time(NULL));

if((array=malloc(current\_size\*sizeof(int))) == NULL)

{

perror(NULL);

exit(-1);

}

while(1)

{

aux=rand();

array[index++]=aux;

if(index == current\_size)

{

current\_size=current\_size + CHUNK;

if((array=realloc(array,current\_size\*sizeof(int))) == NULL)

{

perror(NULL);

exit(-1);

}

}

if(index>=MAX)

{

break;

}

}

if((array=realloc(array,current\_size\*sizeof(int))) == NULL)

{

perror(NULL);

exit(-1);

}

\*size=index;

return array;

}

char \*array\_to\_string(int \*array, int size)

{

char \*string;

int i;

int offset=0;

if((string=malloc((size\*13+1)\*sizeof(char))) == NULL)

{

return NULL;

}

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

{

offset=offset + sprintf(string+offset,"%d ",array[i]);

}

string[offset]='\0';

return string;

}

int main(void)

{

int \*array=NULL;

int size=0, i;

array=functie(&size);

char \*string;

printf("array:\n");

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

{

printf("%d ", array[i]);

}

printf("\n");

string=array\_to\_string(array,size);

printf("string:\n");

printf("%s ",string);

free(string);

free(array);

return 0;

}