Zadanie | Testy

Napíšte funkciu `int parne(int x[], int pocetx, int y[])`, ktorá skopíruje všetky párne čísla z poľa `x` do poľa `y` v poradí v akom sa nachádzajú v poli `x` a vráti počet prvkov poľa `y`. Argument `pocetx` určuje počet prvkov poľa `x`.

Môžete predpokladať, že argument `y`, bude mať dostatočnú veľkosť pre všetky párne prvky.

Ukážka volania:

```
x = {4, 7, 1, 3, 2, 5, 6}
pocetx = 7
pocety = parne(x, pocetx, y);  // volanie funkcie
pocety: 3                      // vypis vysledku
y: {4, 2, 6}
```

uloha6-1.c

```c
// uloha6-1.c -- Peter Plevko, 30.10.2019 08:00

#include <stdio.h>

int parne(int x[], int pocetx, int y[])
{
  int j=0,i=0;
for (i=0;i<pocetx;i++)
if (x[i]%2==0)
{
y[j]=x[i];
j=j+1;
}

return j;
}

int main()
{
  int i, x[10], pocetx;
  scanf("%d", &pocetx);
  for (i = 0; i < pocetx; i++)
    scanf("%d", &x[i]);

  int y[10];
  int pocety = parne(x, pocetx, y);
  printf("pocety: %d\ny: {", pocety);
  for (i = 0; i < pocety; i++)
  {
    if (i > 0)
      printf(", ");
    printf("%d", y[i]);
  }
  printf("}\n");
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Napíšte funkciu `int nasobky(int x[], int pocetx, int y[], int k)`, ktorá z poľa `x` prekopíruje do poľa `y` všetky násobky čísla `k` (`k >= 0`), v poradí v akom sa nachádzajú v poli `x` a vráti počet prvkov poľa `y`. Argument `pocetx` určuje počet prvkov poľa `x`.

Môžete predpokladať, že argument `y`, bude mať dostatočnú veľkosť pre všetky násobky čísla `k`, ktoré sú v poli `x`.

Ukážka volania:

```
x = {4, 7, 10, 1, 3, 9, 2, 5, 8, 6}
pocetx = 10
pocety = nasobky(x, pocetx, y, 2);   // volanie funkcie
pocety: 5                            // vypis vysledku
y: {4, 10, 2, 8, 6}
```

uloha6-2.c

```c
// uloha6-2.c -- Peter Plevko, 30.10.2019 08:01

#include <stdio.h>

int nasobky(int x[], int pocetx, int y[], int k)
{
int i=0,j=0,s=0;

    for (i=0;i<pocetx;i++)
    {
        if (k != 0){
          if (x[i]%k==0)

            {
            y[j]=x[i];
            j++;
            s++;
            }
        }
        else
        {
            if (x[i] == 0) {
                y[j] = x[i];
                j++;
                s++;
                }
        }
    }

return s;
}


int main()
{
    int i, x[10], pocetx;
    scanf("%d", &pocetx);
```

Kompilácia

Štandardný vstup

```
1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Napíšte funkciu `int delitele(int x[], int pocetx, int y[], int k)`, ktorá z poľa `x` prekopíruje do poľa `y` všetky delitele čísla `k`, v poradí v ako sa nachádzajú v poli `x` a vráti počet prvkov poľa `y`. Argument `pocetx` určuje počet prvkov poľa `x`.

Môžete predpokladať, že argument `y`, bude mať dostatočnú veľkosť pre všetky delitele čísla `k`, ktoré sú v poli `x`.

Ukážka volania:

```
x = {4, 7, 10, 2, 3, 9, 6, 5, 8, 12}
pocetx = 10
pocety = delitele(x, pocetx, y, 24); // volanie funkcie
pocety: 6                           // vypis vysledku
y: {4, 2, 3, 6, 8, 12}
```

uloha6-3.c

```c
// uloha6-3.c -- Peter Plevko, 30.10.2019 09:31

#include <stdio.h>

int delitele(int x[], int pocetx, int y[], int k)

{
int i=0,j=0,s=0;


    for (i=0;i<pocetx;i++)
    {
        if (x[i] != 0){
          if (k%x[i]==0)

            {
            y[j]=x[i];
            j++;
            s++;
            }
        }

    }

return s;



}

int main()
{
    int i, x[10], pocetx;
    scanf("%d", &pocetx);
    for (i = 0; i < pocetx; i++)
        scanf("%d", &x[i]);
```

Kompilácia

Štandardný vstup

```
1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Napíšte program, ktorý z prvého riadku vstupu načíta celé číslo `n` a alokuje v pamäti blok `n` položiek pre znaky. Potom zo štandardného vstupu načíta `n` znakov a vypíše ich odzadu. (Využite ukazovateľovú aritmetiku.)

Ukážka vstupu:

```
4
ahojky
```

Výstup pre ukážkový vstup:

```
joha
```

uloha6-4.c

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{

    char* pole;
    int i,n;
    //nacita cele cislo n
    scanf("%d ", &n);

    pole = (char*)malloc(n * sizeof(char));
    fgets(pole, n+2, stdin);


    pole[n] = 0;

    for (i = n-1;i>=0;i--)
    {
        printf("%c", pole[i]);
    }



return 0;
}
```

Kompilácia

Štandardný vstup

```
7
abecedaxxx
```

Štandardný výstup

Zadanie | Testy

Napíšte program, ktorý slová zo súborov `prvy.txt` a `druhy.txt` zapíše do súboru `treti.txt` striedavo tak, že každé nepárne slovo v súbore `treti.txt` bude zo súboru `prvy.txt` a každé párne zo súboru `druhy.txt` v poradí, ako boli v pôvodných súboroch. Každé (aj posledné) slovo v súbore `treti.txt` bude nasledované medzerou. Naviac, pred každým slovom bude značka vyjadrujúca, z ktorého súboru slovo pochádza. Ak zo súboru `prvy.txt`, značkou je znak `+`, ak zo súboru `druhy.txt`, značkou je znak `-`. Ak niektorý zo súborov obsahuje viac slov ako druhý, potom tieto budú zapísané za sebou na konci súboru `treti.txt`. Predpokladajte, že slová obsahujú len písmená a oddelené môžu byť len jednou medzerou alebo jedným znakom konca riadku.

Ukážka súboru `prvy.txt`:

```
Ahojte
nasi studenti
ktori maju radi programovanie
```

Ukážka súboru `druhy.txt`:

```
vsetci mili
```

Ukážka súboru `treti.txt`:

```
+Ahojte -vsetci +nasi -mili +studenti +ktori +maju +radi +program
```

uloha6-5.c | prvy.txt | treti.txt | druhy.txt

```c
#include <stdio.h>

int main()
{

  char a[100], b[100];
  int c = 1, d = 1;

  FILE *fa = fopen("prvy.txt", "rt");
  FILE *fb = fopen("druhy.txt", "rt");
  FILE *fc = fopen("treti.txt", "wt");

  while (c==1||d==1)
  {
    if (c > 0)
      c = fscanf(fa, "%s", a);
    if (d > 0)
      d = fscanf(fb, "%s", b);
    if (c <= 0 && d <= 0)
      break;

    if (c > 0)
      fprintf(fc, "+%s ", a);
    if (d > 0)
      fprintf(fc, "-%s ", b);
  }

  fclose(fa);
  fclose(fb);
  fclose(fc);

  return 0;
}
```

Kompilácia

Štandardný vstup

Štandardný výstup

Zadanie | Testy

Napíšte program, ktorý zistí počet jednotlivých písmen v každom riadku súboru. Vstupom programu je jeden riadok obsahujúci meno súboru. Výstupom je histogram výskytu písmen zapísaný v prehľadnej tabuľke, kde prvý riadok bude obsahovať všetky písmená abecedy prehľadne oddelené. Každý ďalší riadok bude obsahovať číslo riadku a vždy pod písmenami budú zarovnané počty výskytov tohto písmena v jednotlivých riadkoch súboru (nerozlišujte medzi veľkými a malými písmenami). Počty výskytov uveďte ako najviac dvojciferné celé číslo predchádzané jednou medzerou. Všetky riadky výstupu budú ukončené znakom konca riadku.

Ukážka vstupu:

```
subor.txt
```

Ukážka obsahu súboru `subor.txt`:

```
Toto je ukazkovy subor.
V subore su pismena.
```

Výstup pre ukážkový vstup:

```
   A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U
 1  1  1  0  0  1  0  0  0  0  1  2  0  0  0  4  0  0  1  1  2  2
 2  1  1  0  0  2  0  0  0  1  0  0  0  1  1  1  1  0  1  3  0  2
```

uloha6-6.c | subor.txt

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#define N ('Z' - 'A' + 1)

    int main() {
        int x=0,i;
        char hist[N], slovo[1000];
        char nazovsuboru[1000];
        gets(nazovsuboru);

        FILE* Fread;

        Fread = fopen(nazovsuboru, "r");

        for (i = 0; i < N; i++)
            hist[i] = 0;

        i = 0;

        printf("  ");
        for (i = 0; i < N; i++)
        {
            printf(" %2c", i + 'A');
        }
        printf("\n");

        while (fgets(slovo,1000, Fread) != NULL)
        {

            for (i = 0; i < N; i++)
            {
                hist[i] = 0;
            }
```

Kompilácia

Štandardný vstup

```
1 abc123.txt
```

Štandardný výstup

```
1
```

Zadanie | Testy

V súboroch `cisla1.txt` a `cisla2.txt` sa nachádzajú usporiadané postupnosti celých čísel oddelených medzerami. Napíšte program, ktorý spojí tieto dve postupnosti do jednej spoločnej postupnosti do súboru `vysledok.txt` tak, aby výsledná postupnosť bola usporiadaná, a aby obsahovala každé z čísel zo súborov `cisla1.txt` a `cisla2.txt`. Postupnosti môžu byť ľubovoľne dlhé.

Ukážka súboru `cisla1.txt`:

```
2 4 6 8 10 12 14 16
```

Ukážka súboru `cisla2.txt`:

```
-10 -5 0 5 10
```

Ukážka súboru `vysledok.txt`:

```
-10 -5 0 2 4 5 6 8 10 10 12 14 16
```

uloha6-7.c | vysledok.txt | cisla2.txt | cisla1.txt

```c
// uloha6-7.c -- Peter Plevko, 13.11.2019 22:18

#include <stdio.h>

int main()
{

int c;
int pole[100],pocitadlo=0;
FILE *cisla,*cisla_2,*vysledok;
cisla=fopen("cisla1.txt","r");
cisla_2=fopen("cisla2.txt","r");
vysledok=fopen("vysledok.txt","a");

while(fscanf(cisla,"%d",&c)==1)
{
pole[pocitadlo]=c;
pocitadlo++;
}
while(fscanf(cisla_2,"%d",&c)==1)
{
pole[pocitadlo]=c;
pocitadlo++;
}
//printf("%d",pocitadlo);
int pomocna;
for(int i=0;i<pocitadlo-1;i++)
{
  for(int j=0;j<pocitadlo-i-1;j++)
  {
    if(pole[j]>pole[j+1])
    {
      pomocna=pole[j];
      pole[j]=pole[j+1];
      pole[j+1]=pomocna;
    }
  }
```

Kompilácia

Štandardný vstup

```
1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Napíšte rekurzívnu funkciu `dlzka()`, ktorá vráti dĺžku reťazca. Funkciu použite v programe, ktorý pre každý reťazec na štandardnom vstupu vypíše jeho dĺžku, oddelenú medzerou. Reťazce na vstupe sú oddelené medzerou alebo novým riadkom. Na posledný riadok program vypíše správu `Najdlhsie slovo ma X znakov`, kde X je počet znakov v najdlhšom slove.

Ukážka vstupu

```
Smolkovia su najlepsi!
```

Výstup pre ukážkový vstup

```
9 2 9
Najdlhsie slovo ma 9 znakov.
```

uloha6-8.c

```c
// uloha6-8.c -- Peter Plevko, 13.11.2019 22:33

#include <stdio.h>

int dlzka( char *a )
{
  if ( *a !='\0' )
    return 1+dlzka(a+1);
  return 0;
}

int main()
{
  char buf [500] ;
  int max = 0, i;

  while (scanf("%s", buf) > 0 )
  {
    printf("%d ", i=dlzka(&buf[0]) );
    if ( max<i )
      max = i;
  }
  printf("\nNajdlhsie slovo ma %d znakov\n", max);
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1 Smolkovia su najlepsi!
```

Štandardný výstup

```
1
```

Zadanie | Testy

Napíšte rekurzívnu funkciu `samohlasky()`, ktorá vráti počet jednoduchých samohlások v reťazci. Znaky a, e, i, o, u, y považujte za jednoduché samohlásky, dvojhlásky neuvažujte (počítajte ako dve jednoduché samohlásky). Funkciu použite v programe, ktorý pre každý reťazec na štandardnom vstupu vypíše počet jednoduchých samohlások na samostatný riadok. Reťazce na vstupe sú oddelené medzerou alebo novým riadkom.

Ukážka vstupu

```
Smolkovia su najlepsi!
```

Výstup pre ukážkový vstup

```
4
1
3
```

uloha6-9.c

```c
// uloha6-9.c -- Peter Plevko, 13.11.2019 23:40

#include <stdio.h>

int samohlasky(char *s)
{
  char *z = "aeiouy";

  if ( *s=='\0' )
    return 0;

  int pocet = samohlasky(s+1) ;

  while (*z !='\0')
    pocet += s[0]==(z++)[0] ;

  return pocet;
}

int main()
{
  char buf [1000] ;
  while(scanf("%s", buf) > 0)
    printf("%d\n", samohlasky(buf));
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1 Smolkovia su najlepsi!
```

Štandardný výstup

```
1
```