Zadanie | Testy

Napíšte program, ktorý pre postupnosť čísel z intervalu <0, 99>> na vstupe vytvorí histogram/grafikon po desiatkach, podobne ako pre mestskú hromadnú dopravu sa po hodinách vypisujú minúty príchodu autobusov.

Tento histogram pre každú číslicu, ktorá sa vyskytuje na pozícií desiatok, vypíše všetky číslice predstavujúce jednotky v týchto číslach, usporiadané vzostupne.

Štandardný vstup obsahuje až do konca čísla z intervalu <0, 99>. Nie je vopred zadaný počet čísel na vstupe, čítajte do konca vstupu. Na štandardný výstup vypíšte histogram/grafikon podľa požiadaviek.

Ukážka vstupu:

```
1 2 5 2 25 27 93 4 93 93 58 51
```

Výstup pre ukážkový vstup:

```
0 | 12245
2 | 57
5 | 18
9 | 333
```

uloha12-1.c

```c
// uloha12-1.c -- Peter Plevko, 11.12.2019 08:31

#include <stdio.h>

int main()
{
  int sucetpola=0;
   int arr[1000];
  int temp,prva_cifra=0,posledna_cifra=0;
int x=0,y=0,i=0,j=0,pole[1000];
int s=0;
int cislo;
int dvojrozmernepole[1000][1000];


while(scanf("%d",&cislo)==1)
{
pole[i]=cislo;
i++;
s++;
}


for (i=0;i<s;i++)
{
posledna_cifra=pole[i]%10;
prva_cifra=(pole[i]/10)%10;

 dvojrozmernepole[prva_cifra][i]=posledna_cifra;

}
// sort
x=0;

for (x=0;x<s;x++)
{
    for(i=0; i<s; i++)
```

Kompilácia

Štandardný vstup

```
4 66 33 73 95 32 52 55 11 66 6 9 8 23 55 1 2 5
74 66 33 73 95 32 52 55 11 66 6 9 8 23 55 1 2
64 66 33 73 95 32 52 55 11 66 6 9 8 23 55 1 2
```

Štandardný výstup

Zadanie | Testy

Štvorec je magický vtedy, keď súčet prvkov vo všetkých riadkoch, stĺpcoch a uhlopriečkach je rovnaký. Napíšte funkciu `int is_magic(int **a, int n)`, ktorá zistí, či je daný štvorec magický. Argument `a` obsahuje štvorec veľkosti `n`. Rozmer štvorca `n` je do 20. Funkcia vráti 1 ak je štvorec `a` magický, inak vráti 0.

Príklad magického štvorca:

```
8 1 6
3 5 7
4 9 2
```

uloha12-2.c

```c
// uloha12-2.c -- Peter Plevko, 11.12.2019 08:32

#include <stdio.h>
#include <stdlib.h>

int is_magic(int **a, int n)
{
  int x=0,y=0,z=0,i,j;
  for (i=0; i<n-1; i++){
    for (j=0; j<n; j++){
      x+=a[i][j];
      y+=a[i+1][j];
    }
    z=x;
    if (x!=y) return 0;
      x=0;
      y=0;
  }
  for (j=0; j<n; j++){
    for (i=0; i<n; i++){
      x+=a[i][j];
    }
    if (x!=z) return 0;
      x=0;
  }

  for (i=0; i<n; i++){
    for (j=0; j<n; j++){
      if (i==j)
        x+=a[i][j];
      if ((n-i-1)==j)y+=a[i][j];
    }
  }
if (x!=z || y!=z) return 0;
return 1;
}
```

Kompilácia

Štandardný vstup

1

Štandardný výstup

1

Zadanie | Testy

Vytvorte program pozostávajúci z troch modulov: `retazec`, `subor` a hlavného programu.

Modul *retazec* bude obsahovať: premennú pre reťazec znakov a funkciu na pekný výpis reťazca. Pekne vypísať reťazec znamená orámikovať ho hviezdičkami, napr. pekný výpis slova `ahoj` je:

```
********
* ahoj *
********
```

Modul *subor* bude obsahovať: premennú ukazovateľ na súbor a funkciu na pekný výpis súbora. Pekne vypísať súbor znamená vypísať ho do rámčeka z hviezdičiek, pričom šírka rámčeka je parametrom funkcie. Text, ktorý sa nezmestí do rámčeka je potrebné dať do ďalšieho riadku. Napr. pekný výpis súbora s obsahom (vypisujú sa po slovách, medzeri a nové riadky nie sú podstatné): `Janko Hrasko, Popoluska, Dlhy Siroky a Bystrozraky Snehulienka` a šírkou rámika `20` je:

```
********************
* Janko Hrasko,    *
* Popoluska, Dlhy  *
* Siroky a         *
* Bystrozraky      *
* Snehulienka      *
********************
```

Hlavný modul (program) bude používať premenné z ostatných modulov. Na štandardnom vstupe načíta názov súbora a šírku rámika, a následne pekne vypíše názov súbora a po ňom pekne vypíše obsah súbora.

Ukážka výstupu pre súbor `vstup.txt` a šírku `20`:

```
*************
* vstup.txt *
*************
********************
* Janko Hrasko,    *
* Popoluska, Dlhy  *
* Siroky a         *
* Bystrozraky      *
* Snehulienka      *
********************
```

uloha12-3.c | subor.c | retazec.c | vstup.txt

```c
#include <stdio.h>


extern char string[100];
extern FILE* file;

extern void vypis(char* str);
extern void vypis_suboru(FILE* file, int sirka);

int main() {
  scanf("%s\n", string);

  int sirka;
  scanf("%d\n", &sirka);

  vypis(string);
  vypis_suboru(file, sirka);

  return 0;
}
```

Kompilácia

Štandardný vstup

```
vstup.txt
20
```

Štandardný výstup

```

```

Zadanie | Testy

Pavúk Emil si potrebuje skontrolovať pavučinu. Jeho pavučinu si môžeme predstaviť v rovine ako N uchýtných bodov, označených celým číslom od 1 do N (N je do 100), medzi ktorými sú naťahané pavučie vlákna. Každý bod je zadaný X-ovou a Y-ovou súradnicou v rovine.

Emil si teraz plánuje obchôdzku svojej pavučiny prechodom po týchto bodoch v nejakej postupnosti a zaujíma ho koľko sa nachodí a či je to efektívne -- akú vzdialenosť pri takejto obchôdzke prejde, a priemernú vzdialenosť medzi bodmi. Priemerná vzdialenosť sa určí ako celková prejdená dĺžka podelená počtom prejdených pavučích vlániek.

Napíšte program, ktorý načíta zo vstupu pavučinu -- v prvom riadku číslo N (počet úchytných bodov), a na nasledujúcich N riadkoch dve desatinné čísla X-ovú a Y-súradnicu i-teho úchytného bodu. Zvyšok čísel do konca vstupu budú tvoriť poradové čísla úchytných bodov, v poradí v akom ich Emil plánuje prejsť. Na výstup napíšte celkovú prejdenú vzdialenosť a priemernú prejdenú vzdialenosť, obe čísla vypíšte na štyri desatinné miesta.

Ukážka vstupu:

```
5
0.0 0.0
10.0 0.0
10.0 10.0
0.0 10.0
5.0 5.0
1 2 3 4 5
```

Výstup pre ukážkový vstup:

```
37.0711 9.2678
```

Vysvetlenie:

```
1-2 = 10.000000
2-3 = 10.000000
3-4 = 10.000000
4-5 = 7.071068
```

uloha12-4.c

```c
#include <stdio.h>
#include <math.h>

// makro na vypocet stvorca cisla
#define SQ(x) ((x)*(x))

typedef struct s_point {

  double x;
  double y;

} S_POINT;

// globalna premenna -- max. 100 uchytnych bodov
S_POINT points[100];

// globalna premenna -- pocet uchytnych bodov
int numPoints = 0;

// pomocna funkcia pre vypocet vzdialenosti medzi pomocnymi bodmi i a j
double vzdialenost(int i, int j)
{
  // dopis telo funkcie
  return sqrt(SQ(points[i - 1].x - points[j - 1].x) + SQ(points[i - 1].y - points[j - 1].y));
}

int main()
{
  // sem napis svoje riesenie
  int n;
  scanf("%d", &n);

  // nacitanie uchytnych bodov
  for(int i = 0; i < n; i++) {
    scanf("%lf %lf", &points[i].x, &points[i].y);
  }
```

Kompilácia

Štandardný vstup

```
10
0.0 0.0
10.0 0.0
10.0 10.0
0.0 10.0
5.0 5.0
20.0 20.0
210.0 220.0
210.0 210.0
220.0 210.0
25.0 25.0
1 2 3 4 5 9 1 9 8 10 9 8 7 6 4
```

Štandardný výstup

Zadanie | Testy

Jakubko sa pripravuje na programátorský test z rekurzie. Prechádza si staršie úlohy a snaží sa ich vyriešiť rekurzívne. Našiel úlohu na nájdenie maximálneho prvku v poli celých čísel.

Na vstupe je jedno kladné celé číslo N a potom N celých čísel, napíšte rekurzívny program, ktorý určí maximum spomedzi týchto čísel a výsledok vypíše na výstup.

Ukážka vstupu:

```
5
2 4 3 8 1
```

Výstup pre ukážkový vstup:

```
8
```

uloha12-5.c

```c
// uloha12-5.c -- Peter Plevko, 12.12.2019 18:37

#include <stdio.h>
#include <stdlib.h>

int najdi_max( int * x,int n )
{
  if (n == 1 )
    return x[0];
  int i = najdi_max(  ++x,--n  );
  if ( *(--x) > i )
    return x[0];
  return i ;
}

int main(void)
{
  int i, n, *x;
  scanf("%d", &n);
  x = (int*)malloc(  n*sizeof(int)  );
  for (i = 0;  i<n  ; i++)
    scanf("%d", &x[i]);

  printf("%d\n", najdi_max(x, n));
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1  20
2  62 4 3 8 1 22 24 23 28 21 52 54 53 58 51 32 34
```

Štandardný výstup

```
1
```

Zadanie | Testy

Jakubko si číta v knižke o číslach. Pre každé číslo si hlave počíta počet cifier, niečo mu nesedí, nejako sa zamotal. Pomôžte mu a napíšte program, v ktorom rekurzívna funkcia určí počet cifier čísla na vstupe.

Na vstupe je viacero kladných celých čísel, napíšte rekurzívnu funkciu, ktorá určí pre každé číslo na vstupe počet jeho cifier. Výsledok vypíšte podľa ukážky nižšie.

Ukážka vstupu:

```
2
5
25
522
1231231
```

Výstup pre ukážkový vstup:

```
pocet_cifier(2) -> 1
pocet_cifier(5) -> 1
pocet_cifier(25) -> 2
pocet_cifier(522) -> 3
pocet_cifier(1231231) -> 7
```

uloha12-6.c

```c
// uloha12-6.c -- Peter Plevko, 14.12.2019 15:30

#include <stdio.h>
#include <stdlib.h>

int pocet_cifier(int n)
{
  if ( n <= 9 )
    return 1 ;
  return 1 + pocet_cifier( n/10 );
}

int main(void)
{
  int k;
  while(scanf("%d", &k) > 0)
    printf("pocet_cifier(%d) -> %d\n", k, pocet_cifier(k));
  return 0;
}
```

Kompilácia

Štandardný vstup

```
63632
75636367
373725
393522
1231231
2115235643
54154124
0
1
2
3
4
5
7
8
9
10
11
12
13
14
15
```

Štandardný výstup

```
```

Zadanie | Testy

Jakubko sa pripravuje na programátorský test z rekurzie. Prechádza si staršie úlohy a snaží sa ich vyriešiť rekurzívne. Našiel úlohu na výpočet súčtu všetkých párnych čísel v poli.

Na vstupe sú celé čísla (najviac 100 čísel), napíšte rekurzívny program, ktorý na výstup vypíše súčet tých čísel zo vstupu, ktoré sú párne.

Ukážka vstupu:

```
1 2 3 4 5 6 7 8
```

Výstup pre ukážkový vstup:

```
20
```

uloha12-7.c

```c
// uloha12-7.c -- Peter Plevko, 12.12.2019 18:38

#include <stdio.h>
#include <stdlib.h>

int sucet_parnych(int *a, int n)
{
  if (n == 0 )
    return 0;
  if ( *a%2 == 0)
    return *a + sucet_parnych( ++a , n-1 );
  return sucet_parnych( ++a , n-1 );
}

int main(void)
{
  int pole[100], i;
  while(scanf("%d", &pole[i++]) > 0);
  printf("%d\n", sucet_parnych(&pole[0], i-1));
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1  -12 1 2 3 4 5 6 7 8 9 10 11 12 -10 20 30 40 -1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Jakubko sa hrá so štvorcami čísel. Štvorec čísla x je číslo x*x, napr. pre 3 to je 3*3 = 9. Zaujíma ho, koľko štvorcov čísel postupne od 1 môže spočítať tak, aby nepresiahol nejakú hodnotu. Napr. pre hodnotu K=15 môže spočítať štvorce 1*1 + 2*2 + 3*3, čím dostane hodnotu 14. Ak by k tomu pripočítal ešte štvorec 4*4 dostal by už hodnotu presahujúcu K=15.

Na vstupe je niekoľko čísiel K, napíšte rekurzívnu funkciu, ktorá zistí, koľko najviac štvorcov čísel postupne od 1, 2, ... môže Jakubko spočítať, aby nepresiahol túto hodnotu. Výsledok vypíšte vo forme rovnice, ktorá ukazuje príslušný súčet, podľa ukážky nižšie.

Ukážka vstupu:

```
4
5
6
13
14
15
```

Výstup pre ukážkový vstup:

```
1^2 = 1 <= 4
1^2 + 2^2 = 5 <= 5
1^2 + 2^2 = 5 <= 6
1^2 + 2^2 = 5 <= 13
1^2 + 2^2 + 3^2 = 14 <= 14
1^2 + 2^2 + 3^2 = 14 <= 15
```

uloha12-8.c

```c
// uloha12-8.c -- Peter Plevko, 14.12.2019 21:11

#include <stdio.h>
#include <stdlib.h>

int zisti(int k, int n)
{
  if ( n*n <= k )
    return zisti( k-n*n , n+1 );
  return n-1 ;
}

int main(void)
{
  int i, j, k, sucet;
  while(scanf("%d", &k) > 0)
  {
    j = zisti(k, 0);
    printf("1^2");
    sucet = 1;
    for (i = 2; i <= j; i++)
    {
      printf(" + %d^2", i);
      sucet += i*i;
    }
    printf(" = %d <= %d\n", sucet, k);
  }
  return 0;
}
```

Kompilácia

Štandardný vstup

```
505
504
506
507
553
25235
423452
523
234242
23539245
72452
2547425
274458
```

Štandardný výstup

```
```

Zadanie | Testy

Jakubko sa pripravuje na programátorský test z rekurzie. Prechádza si staršie úlohy a snaží sa ich vyriešiť rekurzívne. Našiel úlohu na určenie všetkých N prvkových variácií s opakovaním z K-prvkovej množiny.

Na vstupe sú prirodzené čísla N a K, napíšte rekurzívny program, ktorý na výstup vypíše všetky N prvkové variácie s opakovaním z K-prvkovej množiny čísel 1, 2, ..., K.

Ukážka vstupu:

```
3 2
```

Výstup pre ukážkový vstup:

```
1 1 1
1 1 2
1 2 1
1 2 2
2 1 1
2 1 2
2 2 1
2 2 2
```

uloha12-9.c

```c
// uloha12-9.c -- Peter Plevko, 14.12.2019 21:18

#include <stdio.h>
#include <stdlib.h>

int v[10];

void var(int n, int k, int x)
{
  int i;
  if (x == n )
  {
    for (i = 0; i<n ; i++)
      printf("%d ", v[i] );
    printf("\n");
    return ;
  }

  for (i = 1 ; i<=k ; i++)
  {
    v[ x ] = i;
    var( n , k , x + 1 );
  }
}

int main(void)
{
  int n, k;
  scanf("%d %d", &n, &k);
  var(n, k, 0 );
  return 0;
}
```

Kompilácia

Štandardný vstup

```
3 5
```

Štandardný výstup

```
1
```