Zadanie | Testy

Vytvorte program predstavujúci kalkulačku v dvojkovej sústave.

(a) V premennej `hodnota` si pamätajte aktuálnu hodnotu, na začiatku ju inicializujte na hodnotu 1. Umožnite ju násobiť dvoma a deliť dvoma (celočíselne). Pri násobení dvoma skontrolujte, či číslo „nepretečie"- ak by to malo nastať, vypíšte chybovú hlášku `Operaciu nie je mozne vykonat`

(b) Do kalkulačky pridajte funkciu na pripočítavanie načítaného čísla. Tiež kontrolujte pretečenie. Pripočítavajte po bitoch. Funkcie použite v programe, ktorý vykonáva príkazy od používateľa:

- `L` -- načítanie hodnoty do premennej hodnota. Príkaz je nasledovaný jednou medzerou a hodnotou, ktorá sa má načítať.
- `M` -- pre násobenie dvomi.
- `D` -- pre delenie dvoma.
- `A` -- pre pripočítanie čísla. Tento príkaz je nasledovaný jednou medzerou a hodnotou, ktorá sa má pripočítať.
- `S` -- pre odpočítanie čísla. Tento príkaz je nasledovaný jednou medzerou a hodnotou, ktorá sa má odpočítať.
- `T` -- pre ukončenie program.

Všetky čísla sú načítané zo štandardného vstupu v desiatkovej sústave. Po vykonaní každého príkazu (okrem `T`), program vypíše na štandardný výstup správu `Aktualna hodnota:` nasledovaná jednou medzerou, aktuálnu hodnotu v desiatkovej sústave a znak konca riadku.

Ukážka vstupu:

```
L 2
A 5
S 3
D
T
```

Výstup pre ukážkový vstup:

```
Aktualna hodnota: 2
Aktualna hodnota: 7
Aktualna hodnota: 4
Aktualna hodnota: 2
```

uloha11-1.c

```c

#include <stdio.h>
#include<math.h>
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>


int L(void)
{
    int cislo;
    scanf("%d", &cislo);

    printf("Aktualna hodnota: %d\n", cislo);
    return cislo;
}
//////////////////////////
int M(int cislo)
{
    cislo = cislo *2;
    printf("Aktualna hodnota: %d\n", cislo);
    return cislo;

}
////////////////////////////
int D(int cislo)
{
    cislo = cislo / 2;
    printf("Aktualna hodnota: %d\n", cislo);
    return cislo;

}
//////////////////
int A(int cislo)
{
    int cislo1;
    scanf("%d", &cislo1);
```

Kompilácia

Štandardný vstup

```
L 2
M
M
A 5
S 3
D
M
M
A 5
S 3
D
L 3
M
M
A 5
S 3
D
M
M
A 5
S 3
D
```

Štandardný výstup

Zadanie | Testy

Jakubko sa znovu hral s bitovou reprezentáciou čísel. Nerozumie tomu celkom, a potreboval by program, ktorý by mu spoľahlivo vykonával bitové operácie pre nejaké číslo:

- zistiť (get) i-ty bit,
- nastaviť (set) i-ty bit na 1 (set)
- nastaviť i-ty bit na 0 (clear)
- zmeniť i-ty bit na opačný (flip).

Napíšte pre neho program, ktorý bude tieto operácie spracúvať zo štandardného vstupu podľa ukážky nižšie.

Ukážka vstupu:

```
get 10 2
get 12 2
set 8 2
set 6 2
clear 6 2
clear 3 2
flip 10 3
flip 10 13
```

Ukážka výstupu pre ukážkový vstup:

```
0 <- get(10,2)
1 <- get(12,2)
12 <- set(8,2)
6 <- set(6,2)
2 <- clear(6,2)
3 <- clear(3,2)
2 <- flip(10,3)
8202 <- flip(10,13)
```

uloha11-2.c

```c
// uloha11-2.c -- Peter Plevko, 11.12.2019 09:58

#include <stdio.h>
#include <string.h>

int get(int x, int n)
{
  return     (x&(1<<n))>>n     ;
}

int set(int x, int n)
{
  return    x|(1<<n)    ;
}

int clear(int x, int n)
{
  return    x&(~(1<<n))    ;
}

int flip(int x, int n)
{
  return    x^(1<<n)    ;
}

int main()
{
  char op[20];
  int x, i;

  while(scanf("%s %d %d", op, &x, &i) == 3)
  {
    if (!strcmp(op, "get"))
      printf("%d <- get(%d,%d)\n", get(x, i), x, i);
    if (!strcmp(op, "set"))
      printf("%d <- set(%d,%d)\n", set(x, i), x, i);
    if (!strcmp(op, "clear"))
      printf("%d <- clear(%d,%d)\n", clear(x, i), x, i);
    if (!strcmp(op, "flip"))
      printf("%d <- flip(%d,%d)\n", flip(x, i), x, i);
  }
  return 0;
}
```

Kompilácia

Štandardný vstup

```
get 10 3
get 12 3
set 8 3
set 6 3
clear 6 3
clear 3 3
flip 10 2
flip 10 5
get 110 6
get 112 5
set 18 6
set 16 5
clear 56 5
clear 13 6
flip 110 5
flip 110 6
```

Štandardný výstup

Zadanie | Testy

Jakubko skúmal bitový zápis čísel. Napísal si číslo po bitoch a hľadal, ktorý bit je najnižší, ktorý je najvyšší a koľko ich vlastne v čísle je jednotkových. Čísel je veľa a Jakubkovi sa už nechce, pomôžte mu a napíšte mu program.

Na vstupe je postupnosť kladných celých čísel, napíšte program, ktorý pre každé z nich na výstup podľa ukážky nižšie vypíše počet jednotkových bitov (count), rád najvyššieho bitu(max) a rád najnižšieho bitu (min).

Ukážka vstupu:

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

Výstup pre ukážkový vstup:

```
1:  count:1  max:0  min:0
2:  count:1  max:1  min:1
3:  count:2  max:1  min:0
4:  count:1  max:2  min:2
5:  count:2  max:2  min:0
6:  count:2  max:2  min:1
7:  count:3  max:2  min:0
8:  count:1  max:3  min:3
9:  count:2  max:3  min:0
```

uloha11-3.c

```c
// uloha11-3.c -- Peter Plevko, 11.12.2019 09:59

#include<stdio.h>

int count(int i)
{
  int c = 0;
  while (i > 0)
  {
    c += i & 1;
    i >>= 1;
  }
  return c;
}

int min(int i)
{
  int c = 0;
  while (i > 0)
  {
    if (i & 1)
      break;
    i >>= 1;
    c = c+1;
  }
  return c;
}

int max(int i)
{
  int j = 0, c = 0;
  while (i > 0)
  {
    if (i & 1)
      c = j;
    i >>= 1;
    j = j+1;
  }
  return c;
}

int main(void)
{
  int i;
  while (scanf("%d", &i) > 0)
  {
    printf("%d:", i);
    printf("  count:%d", count(i));
    printf("  max:%d", max(i));
    printf("  min:%d", min(i));
    printf("\n");
  }
  return 0;
}
```

Kompilácia

Štandardný vstup

```
9433
936
2827
6965
7843
8385
5436
542
6699
1471
2412
77
8740
109
7813
3976
590
3476
9222
5786
5261
5418
2617
6479
5832
1580
2912
5173
4117
3185
3979
9852
4072
3108
3119
```

Štandardný výstup

Zadanie | Testy

Jakubko sa začal venovať vesmírnym javom a zistil, že váš predchádzajúci program na výpočet počtu bitov je pomalý. Potreboval by rýchlejší. Pokúste sa napísať rýchlejší program, ktorý by počet bitov počítal po blokoch, pre ktoré si počty bitov vopred vypočíta.

Na vstupe je postupnosť nezáporných celých čísel, napíšte program, ktorý pre každé na výstup podľa ukážky nižšie vypíše počet jednotkových bitov.

Ukážka vstupu:

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

Výstup pre ukážkový vstup:

```
count(1): 1
count(2): 1
count(3): 2
count(4): 1
count(5): 2
count(6): 2
count(7): 3
count(8): 1
count(9): 2
```

uloha11-4.c

```c
#include<stdio.h>

int slowcount(int i)
{
  int c = 0;
  while (i > 0)
  {
    c += i & 1 ;
    i >>= 1 ;
  }
  return c;
}

int t[256];

int fastcount(int i)
{
  int c = 0;
  while (i > 0)
  {
    c += t[i& 255 ];
    i >>= 8 ;
  }
  return c;
}

int main(void)
{
  int i;
  for (i = 0; i < 256 ; i++)
    t[i] = slowcount(i) ;

  while (scanf("%d", &i) > 0)
    printf("count(%d): %d\n", i, fastcount(i));
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

Štandardný výstup

```
1
```

Zadanie | Návod | Testy

Hľadanie zatúlaného čísla. Jakubko skúmal dlhú postupnosť čísel, ktorá bola zaujímavá tým, že každé číslo v nej bolo dva krát, okrem možno jedného, zatúlaného, ktoré tam bolo len raz. Postupnosť bola také dlhá, že si ju nemohol zapamätať, ale napriek tome sa snažil zistiť, či obsahuje zatúlané číslo, a ak áno, ktoré to je.

Na vstupe je postupnosť nezáporných celých čísel, napíšte program, ktorý v nej nájde zatúlané číslo a vypíše ho na výstup. Ak postupnosť zatúlané číslo neobsahuje, program vypíše `Ziadne`

Ukážka vstupu:

```
3 5 3 6 4 5 4
```

Výstup pre ukážkový vstup:

```
6
```

uloha11-5.c

```c
// uloha11-5.c -- Peter Plevko, 11.12.2019 10:03

#include <stdio.h>

int main(void)
{
  int i, j = 0, k = 0;
  while (scanf("%d", &i) > 0)
  {
    if ( i>=0 )
      k++;
    j = j^i ;
  }
  if ( j==0 && k%2==0 )
    printf("Ziadne\n");
  else
    printf("%d\n", j );
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1 2 3 4 5 6 7 8 9 10 0
10 9 8 7 6 5 4 3 2 1 0
```

Štandardný výstup

Zadanie | Testy

Jakubko sa hral s písmenkami v slovách. Čítal postupne písmenká a hľadal-rozmýšľal, či sa v slove nachádza rovnaké písmenko viackrát. Pri prvom opakujúcom sa písmenku sa potešil. Napíšte pre neho program, ktorý pre každé slovo na vstupe zistí prvé opakujúce sa písmenko, ktoré vypíše na výstup. Ak slovo neobsahuje opakujúce sa písmená, tak vypíše -

Ukážka vstupu:

```
abeceda hello world cool jakubko
```

Ukážka výstupu pre ukážkový vstup:

```
el-ok
```

uloha11-6.c

```c
// uloha11-6.c -- Peter Plevko, 11.12.2019 10:05

#include<stdio.h>

char rovnake_pismena( char* slovo)
{
  int i = 0 ;
  while ( *(slovo) > 64 )
  {
    if (i & 1<<(*slovo-97) )
      return *(slovo) ;
    i |= 1<<(*slovo-97) ;
    slovo++;
  }
  return 0;
}

int main()
{
  char buf[1000], c;
  while(scanf("%s", buf) > 0)
    if ( (c=rovnake_pismena(buf))!=0 )
      printf("%c\n", c);
    else
      printf("-\n");
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1  large businesses and financial institutions ha
```

Štandardný výstup

```
1
```

Zadanie | Testy

Jakubko sa hral so sitom. A to nielen takým obyčajným, ale prvočíselným a veľmi úsporným :) Chcel by vedieť ako také úsporné sito vyrobiť. Vyrobte pre Jakubka jednoduché prvočíselné sito, ktoré by mohol ukázať svojim kamarátom.

Na vstupe je kladné celé číslo N (do 2000) veľkosť sita, do ktorej chce nájsť všetky prvočísla. Na výstup vypíšte všetky prvočísla od 1 do N, každé na samostatnom riadku.

Ukážka vstupu:

```
20
```

Ukážka výstupu pre ukážkový vstup:

```
2
3
5
7
11
13
17
19
```

uloha11-7.c

```c
// uloha11-7.c -- Peter Plevko, 11.12.2019 10:06

#include <stdio.h>

char pole[250];

void sito(int k)
{
  int i, j;
  for (i =  2 ;  i<250 ; i++)
  {
    for (j = i+i; j <= k; j += i)
      pole[   j   ] |=      'N'      ;
  }
}

int main()
{
  int i, n;
  scanf("%d", &n);
  sito(n);
  for (i = 2; i <= n; i++)
    if (          pole[i]!='N'          )
      printf("%d\n", i);
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1  2000
```

Štandardný výstup

```
1
```

Zadanie | Testy

Napíšte funkciu `unsigned int invert(unsigned x, int i, int n)`, ktorá invertuje (zmení 0 na 1 a naopak) v čísle `x` od pozície `i`, `n` bitov. Ostatné bity zostanú nezmenené.

Túto funkciu si môžete otestovať v programe, ktorý číta riadok zo štandardného vstupu obsahujúci 3 celé čísla. Prvé číslo reprezentuje číslo, ktoré sa má invertovať, zapísané v desiatkovej sústave. Druhé číslo reprezentuje pozíciu, od ktorej sa majú bity invertovať (počítame od 0 od najvyššej pozície) a posledné číslo reprezentuje počet bitov, ktoré sa budú invertovať.

Príklad:

```
x = 123 (1111011)
invert(x, 1, 2)
x: 75 (1001011)
```

uloha11-8.c

```c
// uloha11-8.c -- Peter Plevko, 4.12.2019 08:35

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

unsigned invert(unsigned x,int i, int n)
{

  int counter=0,copy=x;
  while(copy>0)
  {
    copy>>=1;
    counter++;
  }
  for(int j=counter-i-n;j<counter-i;j++)
  {
      x^=1<<j;
  }
  return x;
}



void print_bin(unsigned int x)
{
  int i, j;
  for (j = 0, i = 31; i >= 0; i--)
  {
    if (x & (1<<i))
      j = 1;
    if (j)
    {
      if (x & (1<<i))
        printf("1");
      else
        printf("0");
    }
```

Kompilácia

Štandardný vstup

```
123 1 2
```

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 zisťovanie prvočísel.

Na vstupe je jedno kladné celé číslo N, napíšte rekurzívny program, ktorý určí všetky prvočísla do N a vypíše ich na výstup.

Ukážka vstupu:

```
10
```

Výstup pre ukážkový vstup:

```
2
3
5
7
```

uloha11-9.c

```c
// uloha11-9.c -- Peter Plevko, 15.12.2019 18:11

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

int deli( int n,int k )
{
  if ( n%k == 0)
    return 1;
  if (k <= ((k==2)? && deli(n, k-1 ))
    return 1;
  return 0 ;
}

int prvocislo(int n)
{
  return n == 2 || !deli( n,n-1 );
}

int main(void)
{
  int i, n;
  scanf("%d", &n );
  for ( i=2;i<n;i++ )
    if (prvocislo( i ))
      printf("%d\n", i);
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1  10
```

Štandardný výstup

```
1
```