Može da koristi, a ne može da škodi

1. Primer poziva funkcije pow iz zaglavlja math.h

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

int main ()
{
  printf ("5.0 ^ 3 = %lf\n", pow (5.0,3));
  printf ("5.0 ^ (-3) = %lf\n", pow (5.0,-3));
  printf ("27.0 ^ (1/3) = %lf\n", pow (27.0,1/3));
  printf ("27.0 ^ (1/3.0) = %lf\n", pow (27.0,1/3.0));
  printf ("27.0 ^ (-1/3.0) = %lf\n", pow (27.0,-1/3.0));
  printf ("3.14 ^ 12 = %lf\n", pow (3.14,12));
  printf ("32.01 ^ 1.54 = %lf\n", pow (32.01,1.54));
  return 0;
}
```

IZLAZ

```
5.0 ^ 3 = 125.000000
5.0 ^ (-3) = 0.008000
27.0 ^ (1/3) = 1.000000
27.0 ^ (1/3.0) = 3.000000
27.0 ^ (-1/3.0) = 0.333333
3.14 ^ 12 = 918662.051843
32.01 ^ 1.54 = 208.036691
```

2.

```
/* konverzija iz osnove 2-16 u dekadnu */
#include <stdio.h>
#include <stdlib.h>

int main ()
{
  char szInput [256];
  unsigned long ul;
  printf ("Unesite neoznacen broj u osnovi 2-16: ");
  fgets (szInput,256,stdin);
  ul = strtoul (szInput,NULL,2);
// ul = strtoul ("11110011",NULL,2);
  printf ("Uneli ste: %lu. Duplirano: %lu\n",ul,ul*2);
  return 0;
}
```

ULAZ

```
11110011
```

IZLAZ

```
Uneli ste: 243. Duplirano: 486
```

scanf, printf, ... parametar format

- %c char
- %d long, int
- %u unsigned int
- %lld long long
- %f float
- %lf double
- %s char*

cin, cout sinhronizacija

Kada radimo sa fajl stream-ovima, oni su povezani sa internim baferom tipa streambuf. Ovaj bafer je memorijski blok koji se ponaša kao posrednik između toka i fizičkog fajla. Na primer, svaki put kada se pozove funkcija fputc ili putchar ili put (koja upisuje jedan karakter), karakter se ne upisuje

direktno u fizički fajl, već se najprije upisuje u taj bafer. Kada se bafer isprazni, svi podaci se upisuju u fizički medijum (ako je u pitanju izlazni tok) ili se prosto oslobađa (ako je u pitanju ulazni tok). Ovaj proces se označava kao sinhronizacija i vrši se u sledećim situacijama:
1. Prilikom zatvaranja fajla: pre zatvaranja fajla svi baferi koji još uvek nisu prazni su sinhronizovani i svi podaci iz njih se upisuju ili iščitavaju.
2. Kada je bafer pun: bafer ima ograničenu veličinu. Kada je bafer pun, automatski se sinhronizuje.
3. Eksplicitno, sa manipulatorima: kada se određeni manipulatori koriste nad tokovima, vrši se eksplicitna sinhronizacija. Ti manipulatori su: flush i endl.
4. Eksplicitno, sa funkcijom članicom sync(): pozivom funkcije sync, koja nema parametara, vrši se sinhronizacija. Ova funkcija vraća celi broj -1 ukoliko tok nema odgovarajući bafer ili u slučaju neuspeha. U drugom slučaju (ako je sinhronizacija uspela) funkcija sync vraća 0.

ios::sync_with_stdio(true);//omogucuje mixed I/O

Nije preporucljivo kombinovati C++ IO sa C IO, ali …

```
#include <iostream>
#include <cstdio>

using namespace std;

int main()

{
   ios::sync_with_stdio(true); //kombinovanje I/O
   float x;
   cout <<"Unesite realni broj "<<endl;
   cin >> x;
   printf("Realni broj na 2 decimale je %.2f\n", x);
   return 0;

}
```

Forsiranje ispisa na 5 decimala I isključvanje IO sinhronizacije

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()

{
   // iskljucujemo sinhronizaciju da bi ucitavanje teklo brze
   ios::sync_with_stdio(false);
   double x;
   cout <<"Unesite realni broj "<<endl;
   cin >> x;
   cout << fixed << showpoint << setprecision(5)<< x << endl;
   return 0;

}
```

Gaus (okruzno 2007)

Profesor Đurić u poslednje vreme ima velikih problema sa svojim nestašnim đacima. Da bi ih smirio, odlučio je da im da, na prvi pogled, mnogo težak zadatak. U prošlosti su se često primenjivale takve mere smirivanja, i omiljen zadatak je bio sabiranje prvih 1000 brojeva. *A*li, otkako je mali Gaus našao način da brzo izračuna taj zbir, profesori su morali da promene zadatak. I tako je prof. Đurić smislio sledeće: daće đacima nenegativne cele brojeve *A* i *B*, i tražiće od njih da mu kažu koliko brojeva iz intervala [*A*,*B*] (tj. svi brojevi veći ili jednaki od *A* i manji ili jednaki od *B*) ima paran zbir cifara (npr, zbir cifara broja 1234 je 1 + 2 + 3 + 4 = 10, dakle paran broj). Međutim, među đacima se nalazi i mali Draganče koji, poput malog Gausa, želi da što pre reši taj zadatak i nastavi da pravi probleme prof. Đuriću. Kako Draganče nije uspeo da nađe rešenje zadatka, pomoć je potražio od njegovih drugova, mladih programera.

Ulaz:

(Ulazni podaci se nalaze u datoteci **gaus.in**) U prvom redu tekstualne datoteke zapisani su brojevi *A* i *B*, odvojeni jednim razmakom.

Izlaz:

(Izlazne podatke upisati u datoteku **gaus.out**) U izlaznu datoteku ispisati samo jedan broj - koliko ima brojeva iz intervala [*A*,*B*] takvih da im je zbir cifara paran broj.

Ograničenja:

- $0 \leq A \leq B \leq 2^{30}$
- vremensko ograničenje za izvršavanje programa je 1 s.

Primer 1:

gaus.in	**gaus.out**
5 15	5

Primer 2:

gaus.in	**gaus.out**
16 20	3

/zvanicno c++ resenje

```
#include <cstdio>
#include <vector>
using namespace std;

vector<int> cifra;

long count(long n, bool paran, bool manje){
  if (n==-1){
    if (paran) return 1;
    else return 0;
  }
  else{
    long ret=0;

    if (manje){
      ret += 5*count(n-1,paran,true) + 5*count(n-1,!paran,true);
    }
    else{
      if (cifra[n]%2==0) ret += count(n-1, paran, false);
      else ret += count(n-1,!paran,false);

      if (cifra[n]>0){
        int p=1 + (cifra[n]-1)/2;
        int np=(1 + cifra[n]-1)/2;

        ret += p * count(n-1,paran,true);
        ret += np * count(n-1,!paran,true);
      }
    }

    return ret;
  }
}
```

```
int main(){
  long A,B;

  FILE *f;
  f=fopen("gaus.10.in","r");
  fscanf(f,"%ld %ld",&A,&B);
  fclose(f);
```

```
  long res=0;

  while (B>0){
    cifra.push_back(B%10);
    B/=10;
  }
  res+=count(cifra.size()-1,true,false);

  cifra.clear();
  A--;
  while (A>0){
    cifra.push_back(A%10);
    A/=10;
  }
  res-=count(cifra.size()-1,true,false);


  f=fopen("gaus.10.out","w");
  fprintf(f,"%ld\n",res);
  fclose(f);

  return 0;
}
```

RESENJE (bez nizova)

```
#include <cstdio>

long long A,B;

long long res;
int zbir_cifara(long long A)
{
   int tmp=0;
   while (A!=0)
   {
      tmp+=A%10;
      A/=10;
   }
   return tmp;
}
int main()
{
   scanf("%lld %lld",&A,&B);
   if (A+30>B)
   {
      while (A<=B)
      {
         if (zbir_cifara(A)%2==0) res++;
         A++;
      }
   }
   else
   {
      while (B%10!=0)
      {
         if (zbir_cifara(B)%2==0)
         res++;
         B--;
```

```
        }
        if (zbir_cifara(B)%2==0) res++;
        while (A%10!=0)
        {
            if (zbir_cifara(A)%2==0)
            res++;
            A++;
        }

        res+= (B-A)/2;
    }
    printf("%lld",res);
    return 0;
}
```

Kreiranje šablona (Code::Blocks project)

1. Kreirajte novi projekat i prilagodite (uredite) kod koji želite. Na primer:

```
#include <iostream>

#define ffor(_a,_f,_t) for(int _a=(_f),__t=(_t);_a<__t;_a++)

#define FOR(__i, __n) ffor (__i, 0, __n)

using namespace std;

int main()

{

   cout << "Bonjour!" << endl;

   return 0;

}
```

2. Kliknite na File> Save project as template

3. Kad god želite da koristite taj template kao početnu strukturu za svoj projekat, izborom

File > New > From template... možete koristiti već kreiran šablon.