Subject:
Date

```
#include <iostream>
using namespace std;
int main()
{
  int counter = 99;

  while (counter > 0)
{
  cout << counter << endl;
  counter -= 11;
}
  return 0;
}
```

- توی این برنامه عدد 99 تولید می‌شود و 11 واحد از 99 کم می‌شود تا زمانیکه به صفر برسیم و حلقه پایان یابد

* برنامه‌ای که 10 عدد صحیح را با هم جمع می‌کند این برنامه تغییر یافته برنامه دومی جمع است که مثالی از حلقه while است.

* نوشتن برنامه‌ای که 10 عدد را با هم جمع می‌کند

✗ عدد اولیه است ، تعداد عدد های واردی است و sum جمع متغیرهای اولیه از صفر

توی این برنامه میشه درخواست جمع 50 عدد را داشت فقط کافی است شرط while تغییر کند.

Subject:
Date

جمع ۱۰ عدد
حلقه while

```
#include <iostream>
using namespace std;
int main()
{
 int i=0, X, sum=0;
cout << "Enter 10 integers;
while (i<10)
{
 cin >> X;
 sum += X;
 i += ;
}
 cout << "sum of the numbers is:" << sum << endl;
return 0;
}
```

- در برنامه نویسی ما همیشه به دنبال کوتاه کردن دستورات هستیم برای مثال؛ $i = i+1 \Rightarrow i+=1 \Rightarrow ++1$

* با استفاده از عملگرهای پیش و پس کاهشی (i++ ، i--) ،

پس افزایشی و پس کاهشی (i++ و --i) می توان دستورات کوتاه تری نوشت.

این دو عملگرها در عبارت های جبری کاربرد دارد و مشخص می کند چه زمانی یک واحد اضافه شود.

مثلاً اگر قبل از عبارت جبری باشد اول یک واحد اضافه می شود و سپس محاسبات جبری انجام می شود

P4PCO

Subject:
Date

اما اگر بعد از عبارت جبری باشد ابتدا محاسبات جبری انجام می‌شود سپس یک واحد اضافه می‌شود.

* این برنامه را می‌توان برای اعداد اعشاری نیز نوشت. فقط کافی است اعداد را به صورت اعشاری اعلان کنیم (int => double).

- شمارنده‌ها معمولاً اعداد صحیح هستند، int اعلان می‌شود تا حافظه کمتری اشغال شود (در ابتدا)

* برنامه‌ای می‌نویسیم که 50 عدد صحیح را دریافت کرد، و میانگین آن را چاپ کند.

برنامه 50 عدد و میانگین
حلقه while و عملگر پس افزایشی

```
#include <iostream>
using namespace std;
int main()
{
int i=0
double X, sum=0
cout << "Enter 50 numbers: ";
while (i<50)
{
cin >> X;
sum += X;
i++;
}
cout << " sum of the numbers is" << sum << endl
   << "average of the numbers is" << sum/50 << endl;
return 0;
}
```

PqPCO

Subject:
Date

* برنامه‌ای بنویسیم که تعدادی عدد را از کاربر دریافت و میانگین آن را مشخص و چاپ کند

```
#include <iostream>                    گرفتن مقداری عدد
using namespace std;                   مقدار نگهبان
int main()
{
  int i=0, grade, sum=0;
  double average;
  cout << "Enter a grade or -1 to exit: ";
  cin >> grade;
  while (grade != -1)
  {
    sum += grade  →  جمع و حاصل‌گیری
    i++  →  یکی اضافه شود
    cout << "Enter agrad or -1 to exit: ";  → بررسی حلقه اگر -1 بود تمام
    cin >> grade;                          اما اگر عدد دیگری بود انقدر ادامه پیدا
  }                                        می‌کند تا -1 وارد شود
  average = sum / i;  → محاسبه میانگین
  return 0;
}
```

توضیح: در این برنامه از (i برای شمارش تعداد نمره‌های وارد شده استفاده می‌شود که مقدار اولیه آن ۰)

(grade برای ذخیره نمره دریافتی است) (sum برای جمع بستن نمرات است)

Subject:
Date

- در دستور cout ما از کاربر می خواهیم یا یک عدد وارد کند یا ۱- را.

- ۱- به معنی پایان حلقه است است. در این جا ۱- مقدار نگهبان حلقه است. هر زمان خواستیم از حلقه خارج شود این دستور وارد می شود. می گوییم از grade مخالف ۱- باشد حلقه کار خود را انجام می دهد.

* اگر بخواهیم بعد از محاسبه مقدار average (میانگین) چاپ شود لازم است دستور cout آن را اضافه کنیم. در غیر این صورت فقط محاسبه می شود و خروجی نداریم.

* در این برنامه sum ، i هردو در int اعلان شده است و ممکن است دچار اشکال در پاسخ میانگین شویم برای اینکه جواب تقسیم دقیق تر محاسبه شود از عملگر تبدیل استفاده می کنیم

* عملگر تبدیل نوع = static_cast < نوع > (اسم متغیر)

آپدرلاین است

داخل () پارامتر رابه < > تبدیل می کند.

* با یک شرط if_else کنترل می کنیم که آیا عددی وارد شده یا خیر. امکان دارد کاربر به اشتباه همان اول ۱- را وارد کند که در این صورت با صفر صفرم روبرو می شویم $\frac{0}{0}$. در این حالت با پاسخ (not an number)(nan) چاپ می شود. اگر i=0 باشد پیغام عدد وارد

Subject:
Date

شده است و به رو می شویم و می دانیم همان ابتدا ۱- وارد نشده است.

برنامه محاسبه میانگین حسابی چند عدد با استفاده از حلقه تکرار نامعین و دستور شرطی

```
#include<iostream>
using namespace std;
int main()
{
 int i=0, grade, sum=0;
 double average;
 cout<<" Enter a grade or -1 to exit:";
 cin>>grade;
 while(grade!=-1)
 {
  sum+=grade;
  i++;
 cout<<"Enter agrade or -1 to exit:";
 cin>>grade;
 }
 if(i==0)
 cout<<" No grade is entered!"<<endl;
 else{
 average=sum/static_cast<double>(i);
 cout<<"Average is"<<average<<endl;
 }
   return 0;
}
```

Subject:
Date

* برنامه ای بنویسید که چند عدد صحیح را دریافت کند و میانگین هندسی آنها را حساب کند.

- میانگین هندسی = همه‌ی اعداد وارد شده در هم ضرب شود سپس رادیکال (جذر) با فرجه تعداد اعداد گرفته شده

Geometric average => میانگین هندسی

- برای ایجاد تغییر در برنامه قبل کافی است به جای جمع حاکیزنی از ضرب حاکیزنی استفاده شود. += → *=

- چون هر عددی ضرب در صفر برابر با صفر می‌شود برای جلوگیری از این اشکال sum=0 را به sum=1 تغییر می‌دهیم، چون یک عدد خنثی در ضرب است.

- sum در دستور به معنای جمع است و ما برای اینکه دستور خواناتر شود به جای آن از prod استفاده می‌کنیم که prod مخفف product (ضرب) است

- رادیکال به معنی عددی به توان $\frac{1}{i}$ است که i فرجه است.

- در تقسیم یک مشکل داریم چون a و b هر دو عدد صحیح هستند جواب تقسیم عدد صحیح است اگر a عددی بزرگتر از یک باشد پاسخ به صورت صفر است چون قسمت اعشاری آن خوانده نمی‌شود پس باید یکی از آنها را به عدد اعشاری تبدیل کنیم، به راحتی می‌توان برای یک ممیز قرار داد ۱,۰ → ۱.۰ یا می‌شد a را با عملگر تبدیل نوع تغییر دهیم

PAPCO

Subject:
Date

محاسبه میانگین هندسی
حلقه تکرار while ، if ،
سرفایل cmath

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
 int i=0, grade, prod=1;
 double average;
 cout <<" Enter a grade or -1 to exit:";
 cin >> grade;
 while (grade!=-1)
 {
  prod *= grade;
  i++
  cout <<" Enter a grade or -1 to exit:";
  cin >> grade
 }
 if(i==0)
  cout <<" No grade is entered!" << endl;
 else{
 average = pow(prod, 1.0/i);
  cout <<" Geometric average is" << average << endl;
 }
  return 0;
}
```

Subject:
Date

- حلقه دیگری برای انجام دستور های تکراری وجود دارد به نام حلقه for .

- حلقه for شامل ۳ بخش می باشد ۱- شروع حلقه (مقداردهی اولیه)

۲- شرط پایان حلقه ۳- نو (رشد)

- هر برنامه ای که با حلقه while نوشته شود با حلقه for هم قابل نوشتن است و برعکس .

* چگونگی کاربرد حلقه ی for به جای حلقه while .

حلقه while

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
 int i=1;
 while(i<=10)
 {
  cout<<i<<endl;
  i++;
 }
  return 0;
}
```

حلقه for =>

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int i;
 for(i=1; i<=10; i++)
 {
  cout<<i<<endl;
 }
 return 0;
}
```

- اگر متغیر i را داخل حلقه for اعلان کنیم فقط در حلقه در دسترس است یعنی

Subject:
Date

اگر بخواهیم که در جای دیگر از این متغیر استفاده کنیم باید مجدد اعلان شود. پس اگر متغیری در بیرون حلقه for اعلان شود در همه جای تابع main در دسترس است.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  for(int i=99; i>0; i-=11)
{
  cout << i << endl;
}
  return 0;
}
```

* دریافت عدد مثبت صحیح n و محاسبه مقدار فاکتوریل آن.

- فاکتوریل به معنی ضرب اعداد 1 تا n است. پس باید برنامه به گونه ای باشد که تکرار ضرب اعداد را بیان دهد

- در این برنامه n عدد وارد شده توسط کاربر است و f حاصل ضرب اعداد.

Subject:
Date

```
#include <iostream>          محاسبه n!
# include <cmath>            حلقه for
using name space std;
int main ()
{
 int n, f=1;
cout <<" Enter a positive integer:";
 cin >> n;
 for ( int i=1; i<=n; i++)
{
 f *= i;
}
  cout <<n <<"!= " <<f <<endl;
 return 0;
}
```

* برنامه‌ای بنویسم که ترکیب $\binom{n}{k}$ را حساب کند.

$\binom{n}{k} = \frac{n!}{(n-k)!k!} \Rightarrow \binom{10}{3} = \frac{10!}{7!3!}$

- n, k برای ذخیره n, k

- چون ۳ تا فاکتوریل حساب می‌کنیم: $f_1\ n!$, $f_2\ k!$, $f_3\ (n-k)!$

- سعی از سه حلقه جدا استفاده کرد یکبار $n!$ یکبار $k!$ یکبار $(n-k)!$ و سعی از یک حلقه استفاده کرد

Subject:
Date

ترکیب $\binom{n}{k}$
if, for حلقه‌ی

```
#include <iostream>
#include <cmath>
using name space std;
int main()
{
  int k, n, f1=1, f2=1, f3=1;
cout << "Enter k and n (k <= n): ";
cin >> k >> n
for (int i=1; i <= n; i++)
{
  f1 *= i;
  if (i == k)
  f2 = f1;
if (i == n-k)
  f3 = f1;
}
  cout << "combination of " << k << " of " << n
       << "is" << f1/(f2 * f3) << endl;
return 0;
}
```

- بنابر تقدم عملگرها باید در محاسبه ترکیب دقت شود و هم جای درست ( ) استفاده شود.

- = این عملگر به معنی جایگذاری است و سمت راست را در چپ قرار می‌دهد.

PAPCO