Цикли

- Случвало ли ви се е да ви накажат да напишете „Няма да говоря в час“ 1000 пъти?

- Колко по-лесно е веднъж да напишете:

```
for(unsigned i =0; i<1000; ++i)
        std::cout<<“Няма да говоря в час\n”;
```

Цикли

- Цикъл е операция, която се повтаря N на брой пъти
- Спестява ръчното писане на един и същ код
- Едно от най-мощните оръжия в програмирането

Оператор while

- Синтаксис:

```
while(<условие>)
    <команда>;
```

- Докато <условие> е истина ще се изпълнява <команда>

- Силно препоръчително е в <команда> да се извършва действие, което да превърне <условие> в лъжа!

Оператор while

```
while(true)
        std::cout<<“I love C++\n”; //безкраен цикъл


unsigned i =0;
while(i<1000)
{
        std::cout<<i; //ще изпише числата от 0 до 999
        ++i;
}
```

Композиция do...while

- Синтаксис:

```
do
<команда>;
 while(<условие>);
```

- Първо ще се изпълни <команда>, а после докато <условие> е истина ще се изпълнява <команда>

- Същата идея като while, но гарантирано цикълът ще се изпълни поне веднъж

Оператор break

- Ако break се извика в цикъл, цикълът се прекъсва.

- unsigned i = 0;
- while(true)
  if(i==99)
    break; //ето как можем да счупим безкраен цикъл след 100 стъпки
  else
    ++i;

Оператор continue

- Оператор continue прекратява командата в цикъла

- За разлика от оператор break, оператор continue не прекратява цикъла, а само командата, като цикълът продължава все едно нищо не се е случило

- Пример: принтиране на всички четни числа от 1 до 1000

```
int counter = 0;
while(counter<1000)
{
++counter;
if(counter%2!=0)
        continue;
std::cout<<counter<<‘ ‘;
}
```

Оператор for

- Синтаксис: for(<израз1>; <условие>; <израз 2>)<тяло>

- Работа:

1. Изпълнява се <израз1>
2. Проверява се <условие>
3. Ако <условие> е истина се изпълнява <тяло>, ако не е излизаме от цикъла
4. Изпълнява се <израз2>
5. Връщаме се в 2.

Оператор for

```
for(unsigned i =0; i<1000; ++i)
        std::cout<<“Няма да говоря в час\n”;

for(unsigned i =0; i<1000;++i)
{
        if(i%2 == 0) //ако трябва да се редуват две изречения
                std::cout<<“Няма да говоря в час\n”;
        else
                std::cout<<“Ще слушам класната\n”;
}
  //можеше просто двете изречения да се обединят в едно
```

Вложени цикли

- Реален пример, който използвах наскоро. Исках да си изведа всички числа от 1 до 100 по хиляда пъти.
- Но как да ги изведем без да ги пишем ръчно?

```
for(unsigned i =1; i<=100; ++i)
        for(unsigned j = 0; j<1000; j++)
                std::cout<<i<<‘ ‘;
```

Работещи, но странни и непрепоръчителни употреби на for

- unsigned i = 0;

```
for(;  i<1000; ++i)
        std::cout<<“Няма да говоря в час\n”;
```

- unsigned i = 0;

```
for(;  i++<1000;)
        std::cout<<“Няма да говоря в час\n”;
```

Работещи, но странни и непрепоръчителни употреби на for

- unsigned i = 0;

```
for(;  ;){
        std::cout<<“Няма да говоря в час\n”;
        if(++i==1000)
                break;
}
```

Работещи, но странни и непрепоръчителни употреби на for

```
for(unsigned i =0; i<1000; std::cout<<“Няма да говоря в час\n”, ++i);
```

- Всичките тези примери имат една и съща функция, НО не е добра практика да ги ползвате!!!
- Ако няма да използвате for така както трябва, то помислете дали while не е по-добра алтернатива
- Ако на контролното имате код като горните, просто си обърнете цикъла от for в while

Примерно преобразуване

- for(unsigned i = 0; i<15; std::cout<<i<<“\n”, i+=3, --i);

- Преобразуване:

```
unsigned i = 0;
while(i<15)
{
        std::cout<<i<<“\n”;
        i += 3;
        --i;
}
```

- След преобразуване четимостта е в пъти по-голяма