# Przydatne strony

*http://www.algorytm.edu.pl/wstp-do-c/typy-zmiennych.html*

*http://www.algorytm.edu.pl/instrukcja-iteracyjna-ptla/break-continue.html*

*http://www.algorytm.edu.pl/cwiczenia-iteracje.html*

*http://www.algorytm.edu.pl/instrukcja-iteracyjna-ptle.html*

# Instrukcje break i continue -sterujące działaniam pętli

## instrukcja continue

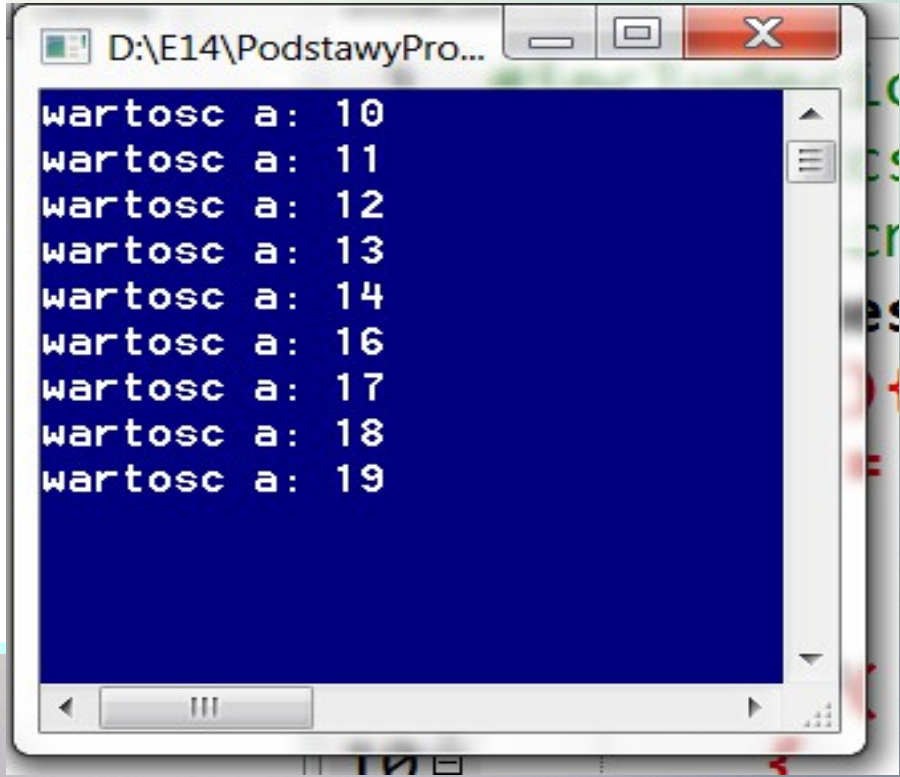
Czasami zdarza się, że w danej iteracji trzeba pominąć wszystkie pozostałe instrukcje pętli, ale nadal w niej pozostać. Umożliwia nam to instrukcja **continue**.

- W pętli **while** i **do-while** powoduje przejście do instrukcji sprawdzającej warunek.
- W pętli **for** przekazuje sterowanie do jej części modyfikacyjnej do trzeciego wyrażenia w nawiasach.

**Instrukcja break** nakazuje kompilatorowi przejście do pierwszej instrukcji znajdującej się za instrukcją złożoną o podanej nazwie.

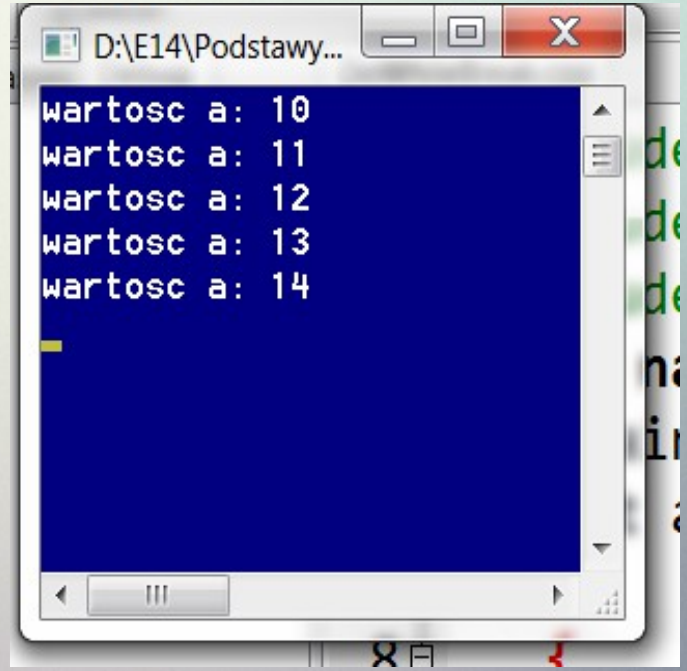
# Przykład

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
int main(){
    int a = 10;
    do
    {
        if( a == 15)
        {
            a = a + 1;
            continue;
        }
        cout << "wartosc a: " << a << endl;
        a = a + 1;
    }while( a < 20 );
cin.ignore();
getchar();
return 0;
}
```



# Przykład

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
int main(){
    int a = 10;
    do
    {
        if( a == 15)
        {
            a = a + 1;
            break;
        }
        cout << "wartosc a: " << a << endl;
        a = a + 1;
    }while( a < 20 );
cin.ignore();
getchar();
return 0;
}
```

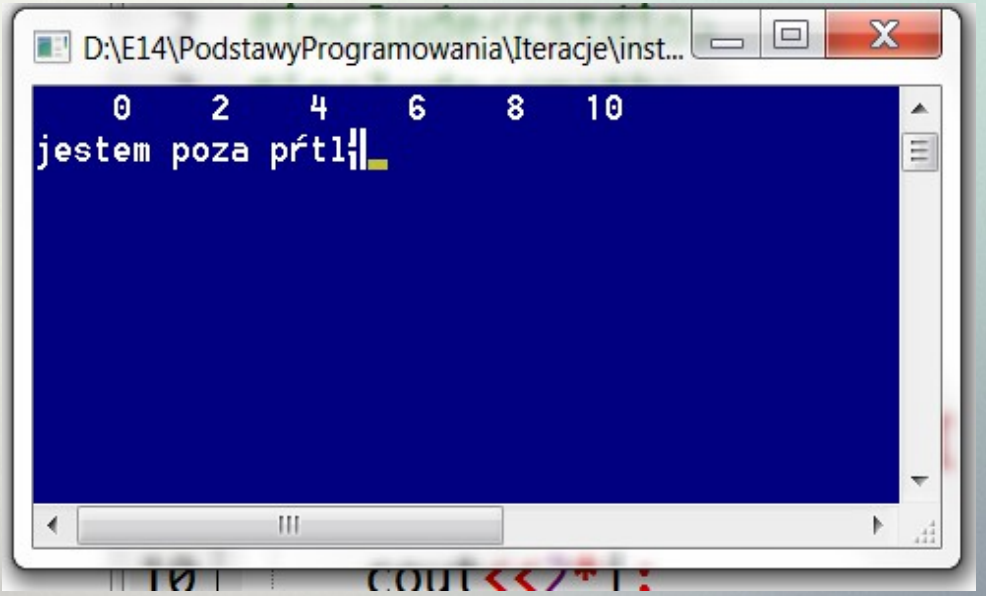


# Przykład

```
int main(){
unsigned int n,i;
cout<<"podaj liczbę naturalną"<<endl;
cin>>n;
if (n<=1){
        cout<<"liczba nie jest liczbą pierwszą i nie jest liczbą złożoną"<<endl;
        }
             else{
            i=2;
            bool p=true;
            while(i<n){

                        if(n%i==0){
                            p=false;
                            break;
                            }
                i++;
            }//koniec while
             if(p) cout<<"podana liczba jest liczbą pierwszą"<<endl;
                else cout<<"podana liczba jest liczbą złożoną"<<endl;
            }

cin.ignore();
getchar();
return 0;
}
```

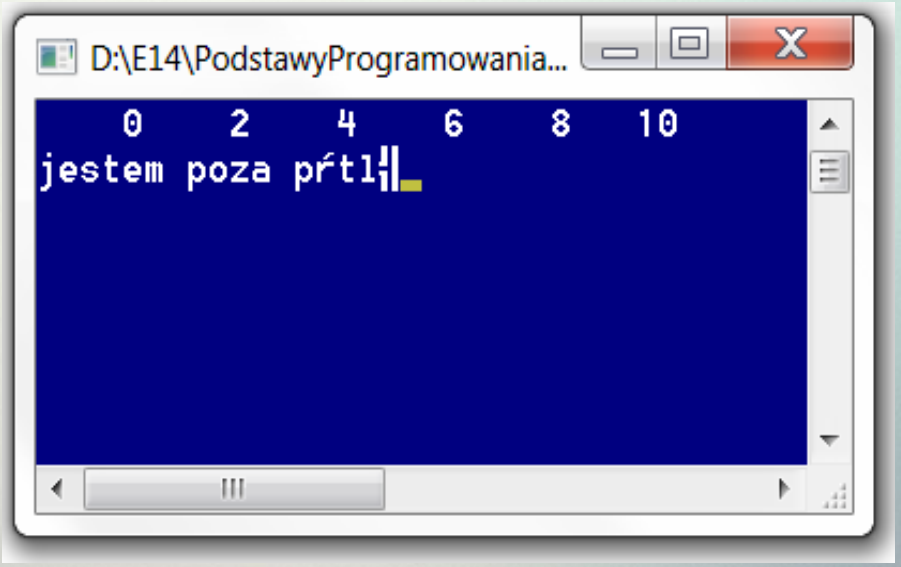
# Przykład

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
int main(){
   int suma = 0;

 for(int i=0;i<10;i++) {
    cout.width(5);
    cout<<2*i;
    suma=suma+2*i;
    if(suma>20 ) break;
 }
  cout<<endl;
  cout<<"jestem poza pętlą"  ;
cin.ignore();
getchar();
return 0;
}
```



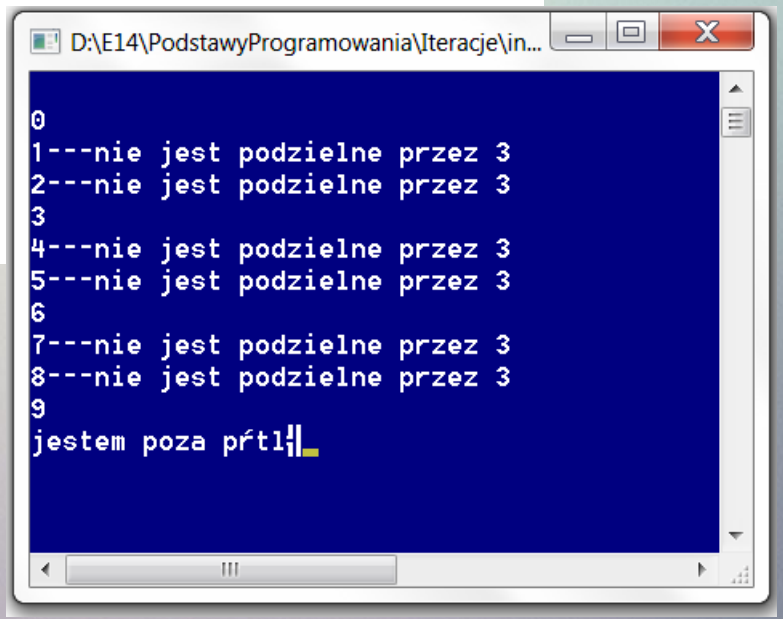
# Przykład

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
int main(){
   int suma = 0;

 for(int i=0;i<10;i++) {
    cout.width(5);
    cout<<2*i;
    suma=suma+2*i;
    if(suma>20 ) break;
 }
  cout<<endl;
  cout<<"jestem poza pętlą"  ;
cin.ignore();
getchar();
return 0;
}
```



# Przykład

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

using namespace std;
int main(){

 for(int i=0;i<10;i++) {
    cout<<endl<<i;
    if(i%3==0) continue;//przy spełnieniu warunku przechodzimy
                        //natychmiast do kolejnego przebiegu pętli
  cout<<"---nie jest podzielne przez 3";
 }
  cout<<endl;
  cout<<"jestem poza pętlą"  ;
cin.ignore();
getchar();
return 0;
}
```



# Generator liczb pseudolosowych w języku C++

**srand() - generator liczb pseudolosowych dostępny w bibliotece <cstdlib>**

**srand(time(NULL)) -za każdym razem inny punkt startowy**
**time-dostępny w bibliotece <ctime>**

**time(NULL)-oznacza, że wartością bazową w procesie generowania liczby przyjmowany jest odczytany z zegara czas, jaki upłynął w sekundach od 1 stycznia 1970 roku**

# Liczby losowe

Do generowania liczb losowych służy instrukcja rand(). Generuje ona liczbę całkowitą ze zbioru {0, 1, ….,RAND_MAX}

RAND_MAX- stała, zdefiniowana w bibliotece <cstdlib>

# Generator liczb pseudolosowych w języku C++

**rand() - zwraca kolejną liczbę pseudolosową w zakresie od 0 do RAND_MAX. Funkcja rand() i stała RAND_MAX zdefiniowane są w pliku nagłówkowym <cstdlib>.**
**Plik ten należy dołączyć do każdego programu wykorzystującego tę funkcję. Stała RAND_MAX ma wartość 32767 (lecz w innych implementacjach C++ może być większa - zawsze to sprawdzaj!).**

**srand(X0) - umożliwia zainicjowanie ziarna generatora pseudolosowego. Jeśli nie zainicjujemy X0, to funkcja rand() będzie generowała zawsze ten sam ciąg liczb pseudolosowych. Ziarno inicjujemy zwykle wartością zwracaną przez funkcję time(), ponieważ wartość ta przy każdym uruchomieniu programu będzie inna i w efekcie otrzymamy inny ciąg liczb pseudolosowych. Funkcja time() wymaga dołączenia pliku nagłówkowego <ctime>**

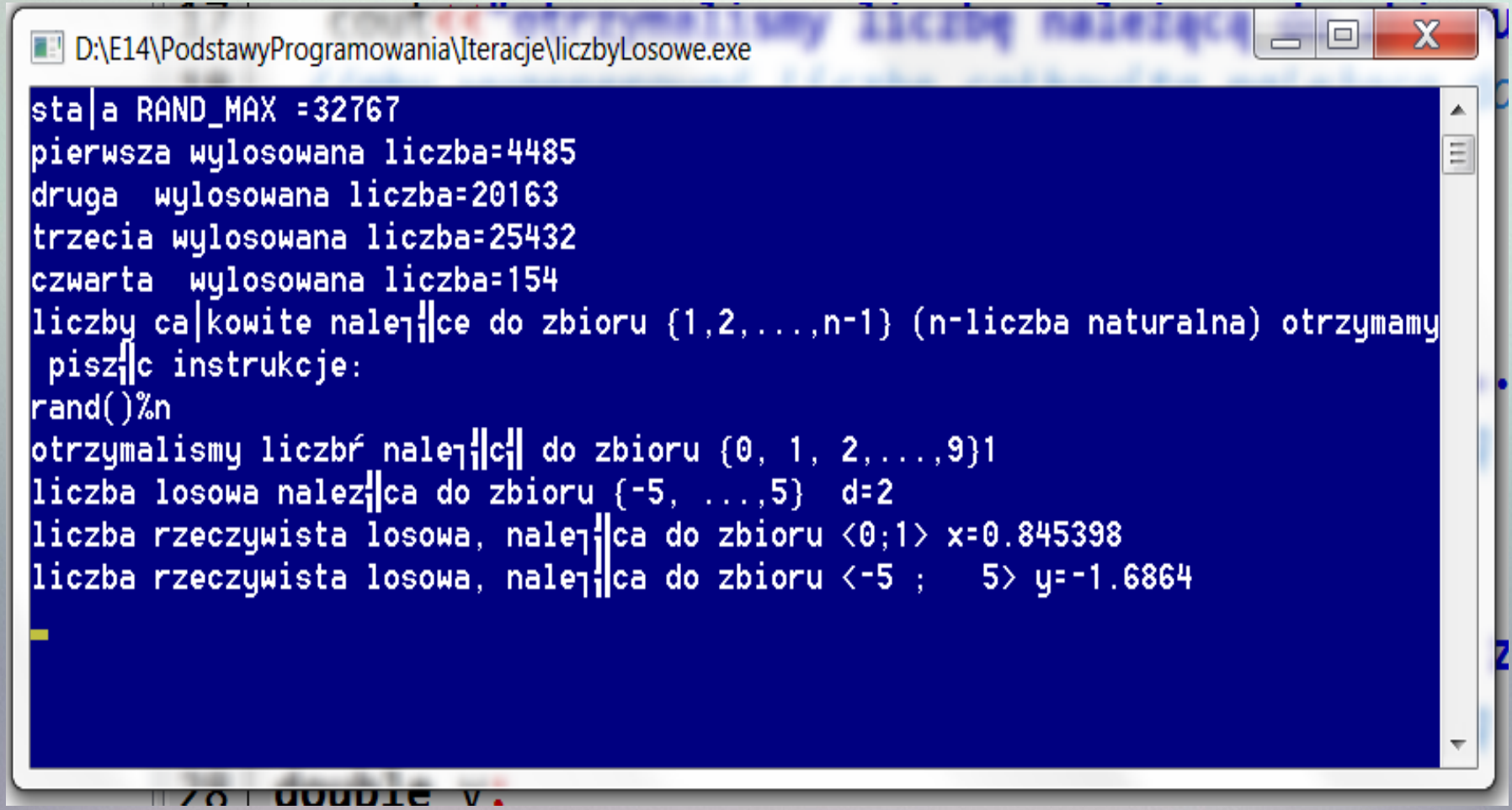
# Generator liczb pseudolosowych w języku C++

**srand(X0) – generator umieszcza się tylko raz, na początku programu. Jeżeli nie umieścimy generatora, zawsze otrzymamy taką samą sekwencję liczb losowych.**
**Spowodowane jest to tym, iż ziarno generatora zawsze startuje od takiej samej wartości. Aby to zmienić, musimy je na początku programu zainicjować wartością, która będzie przy każdym uruchomieniu inna. Tutaj wykorzystujemy funkcję time(), która zwraca wartość bieżącego czasu. Istotne dla nas jest jedynie to, iż wartość ta różni się przy każdym uruchomieniu programu, gdyż czas nieubłaganie płynie do przodu. Wynik funkcji time() jest ziarnem generatora i przekazujemy go jako parametr do funkcji srand(), która ustawi to ziarno.**

# Liczby losowe

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<ctime>
using namespace std;
int main(){
 srand(time(NULL)) ;
 cout<<"stała RAND_MAX ="<<RAND_MAX<<endl;
 cout<<"pierwsza wylosowana liczba="<<rand()<<endl;
 cout<<"druga  wylosowana liczba="<<rand()<<endl;
  cout<<"trzecia wylosowana liczba="<<rand()<<endl;
 cout<<"czwarta  wylosowana liczba="<<rand()<<endl;
 cout<<"liczby całkowite należące do zbioru {1,2,...,n-1} (n-liczba naturalna) otrzymamy pisząc instrukcje:"<<endl;
 cout<<"rand()%n"<<endl;
 int n=10;
  cout<<"otrzymalismy liczbę należącą do zbioru {0, 1, 2,...,9}"<<rand()%n<<endl;
 //aby wygenerować liczbę całkowitą należąca do zbioru {a, ....,b}
 //należy użyć instrukcji: a+rand()%(b+1-a)
 int a=-5, b=5, d;
d=a+rand()%(b+1-a);
cout<<"liczba losowa naleząca do zbioru {-5, ...,5}  d="<<d<<endl;
//jeżeli chcemy wygenerowac liczbę rzeczywistą ze zbioru<0,1> użyjemy instrukcji:   (double)rand()/(RAND_MAX+1)
double x;
x=(double)rand()/(RAND_MAX+1);
cout<<"liczba rzeczywista losowa, należąca do zbioru <0;1> x="<<x<<endl;
//jeżeli chcemy wygenerowac liczbę rzeczywistą ze zbioru<a ,b> użyjemy instrukcji:  a+ ((double)rand()/(RAND_MAX+1))*(b-a
double y;
y=a+ ((double)rand()/(RAND_MAX+1))*(b-a);
cout<<"liczba rzeczywista losowa, należąca do zbioru <-5 ;   5> y="<<y<<endl;
cin.ignore();
getchar();
return 0;
}
```

# Liczby losowe



## Zadanie domowe

Twoim zadaniem domowym jest napisanie prostej gry, która ma działać następująco:

1. Program losuje liczbę z przedziału od 1 do 1000.
2. Użytkownik zgaduje liczbę, która została wylosowana.
3. Jeżeli podana liczba jest za duża (za mała) gra wypisuje stosowny komunikat i powraca do kroku 2.
4. Jeżeli gracz trafi liczbę wylosowaną to progam kończy działanie, wypisując na ekran wylosowaną liczbę oraz liczbę 'strzałów', które oddał gracz.

Gra ma być zabezpieczona przed możliwością wprowadzenia błędnych wartości liczbowych.

**Zad 2**

**Napisz program, który wygeneruje 10 liczb całkowitych należących do zbioru {-10, -9, …, 9, 10}, wypisze je na ekranie i wyznaczy sumę liczb: a) dodatnich,  b) ujemnych**