Wstęp do informatyki- wykład 8

Operatory – operator przecinkowy
Pętla `while` i `do..while`

Treści prezentowane w wykładzie zostały oparte o:

- S. Prata, Język C++. Szkoła programowania. Wydanie VI, Helion, 2012
- *www.cplusplus.com*
- Jerzy Grębosz, Opus magnum C++11, Helion, 2017
- B. Stroustrup, Język C++. Kompendium wiedzy. Wydanie IV, Helion, 2014
- S. B. Lippman, J. Lajoie, Podstawy języka C++, WNT, Warszawa 2003.

Instrukcje iteracyjne - pętla `while`

```
while(wyr) instrukcja1;
```

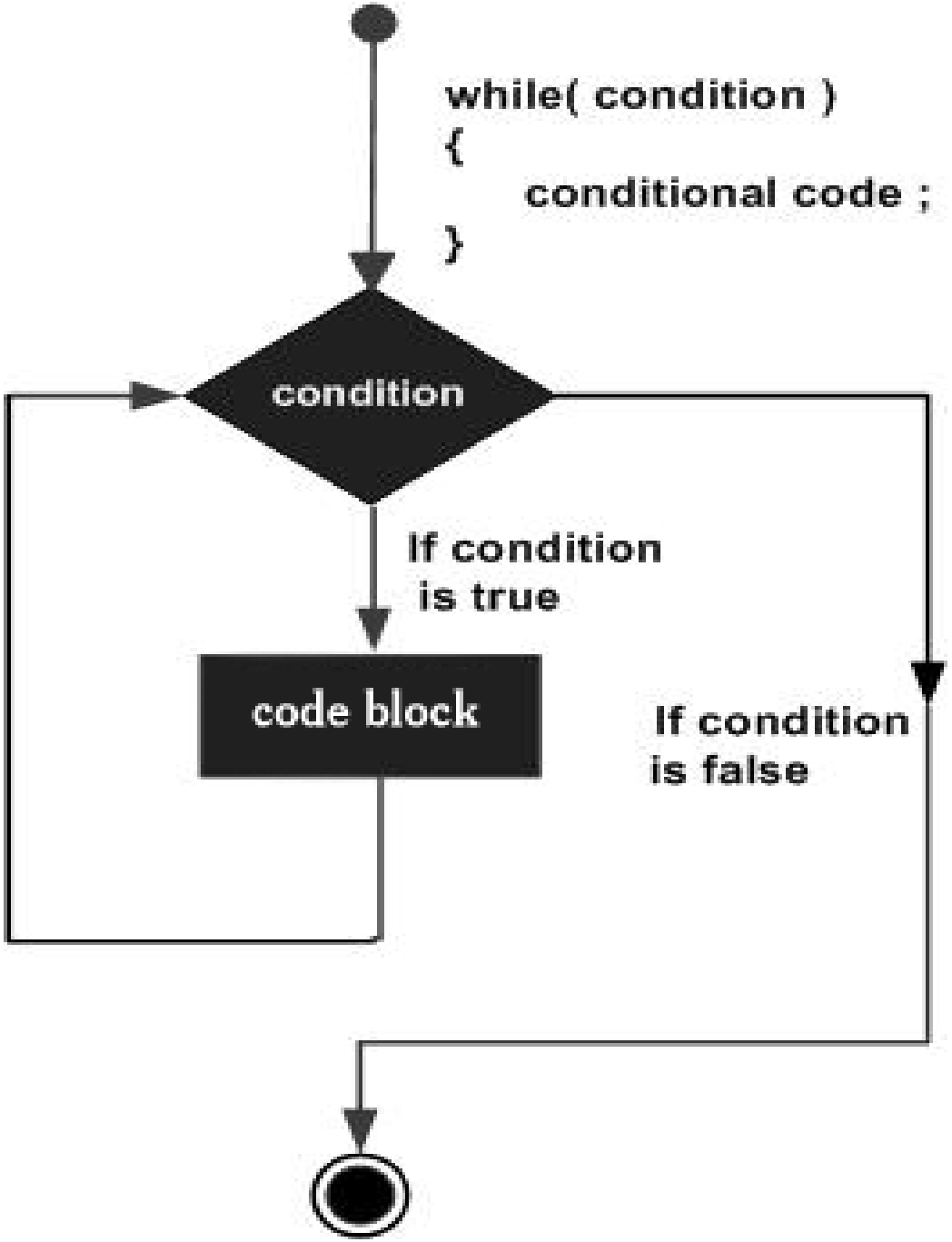
dopóki prawdziwe jest wyr wykonuj instrukcja1

instrukcja jest powtarzana, dopóki **`wyr`** jest prawdziwe (`true`). Obliczenie wartości wyrażenia następuje przed każdym kolejnym wykonaniem instrukcji.

Zwróćmy uwagę, że pierwsze obliczenie wartości **`wyr`** odbywa *się* przed wykonaniem instrukcji **`instrukcja1`**, czyli możliwa jest sytuacja, gdy **`instrukcja1`** nie zostanie wykonana ani razu.

Ponadto przed pętlą **`while`** należy dokonać przygotowania: inicjalizacji zmiennych występujących w wyrażeniu **`wyr`**.

Instrukcje iteracyjne - pętla `while`



Pętla while - przykład

Napisz program, który wyznaczy pierwszą liczbę naturalną postaci 2^n, która jest większa od losowo wybranej liczby lim:

```
int main()
{
  int lim = rand(); //liczba losowa z [0, RAND_MAX]
  cout<<"wylosowana granica to: "<<lim<<endl;

  int liczba = 1;
  while ( liczba <= lim )//dopóki nie przekroczymy lim
      liczba *= 2;//liczymy kolejną potegę liczby 2
  cout << "Szukana liczba to:" << liczba << endl;
  return 0;
}
```

Pętla do...while...

Pętla **do...while...** czyli: **rób... dopóki...**

Instrukcja ta pozwala na realizację innego rodzaju pętli programowej:

```
do
   instrukcja
while( wyr );
```

czyli :

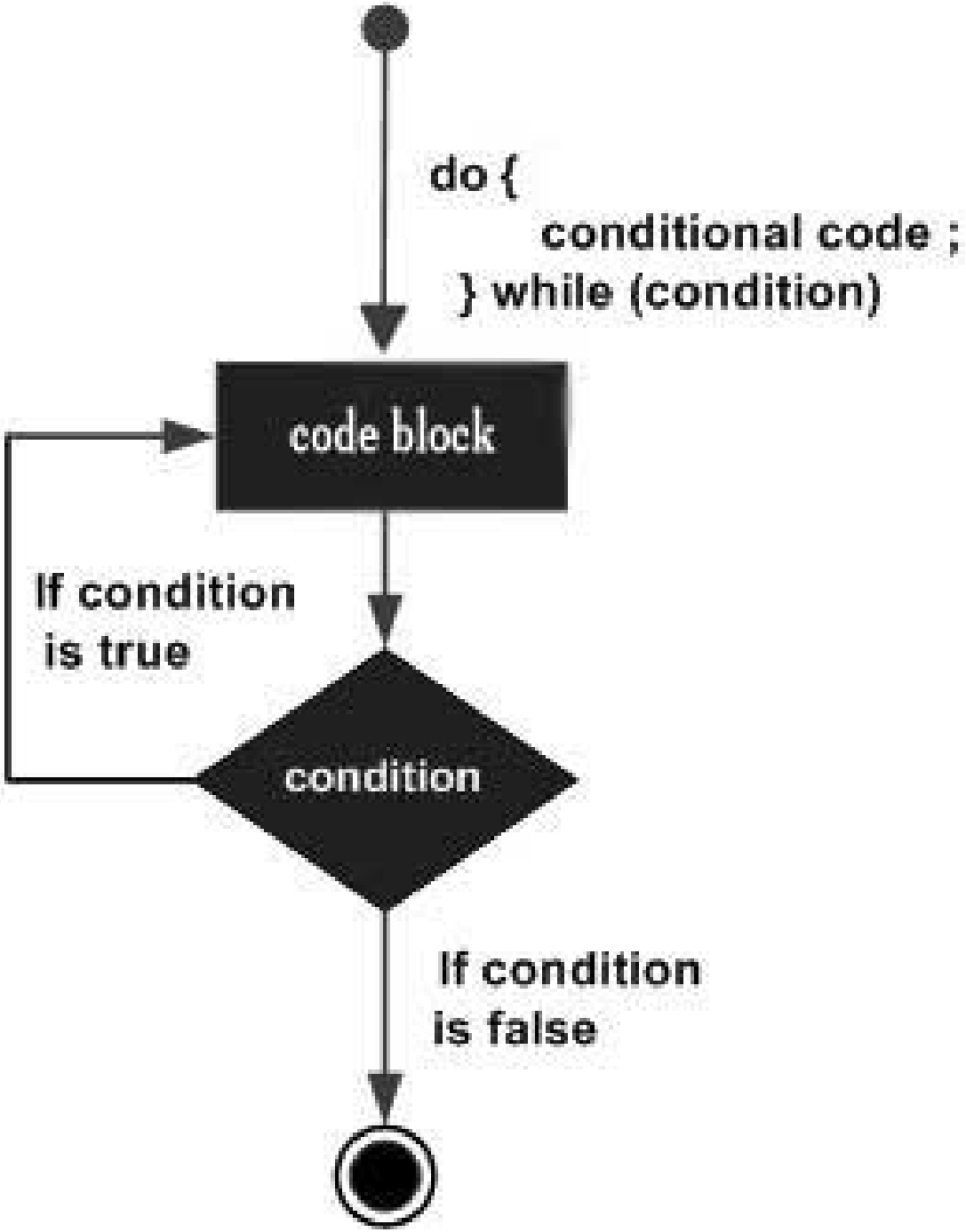
```
rób instrukcja1 dopóki ( wyr );
```

Działanie jej jest takie:

Najpierw wykonywana jest **instrukcja1**.

Następnie sprawdzana jest wartość **wyr**. Jeśli **wyr** ma wartość **true** to wykonanie **instrukcja1** zostanie powtórzone, po czym znowu sprawdzany jest warunek **wyr**... i tak w kółko, dopóki warunek będzie spełniony(prawdziwy).

Pętla do...while...



Pętla do...while -przykład – petla zaporowa

Napisz program, który będzie wczytywał dane od użytkownika aż do chwili, gdy poda on liczbę z przedziału [5, 100]

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

int main()
{
  int n;
  do
  {
    cout << "Podaj liczbe z przedziału [5, 100]";
    cin >> n;
  }while (n  < 5 || n > 100);

  cout << "Liczba z przedziału wynosi:" << n << endl;
  return 0;
}
```

Pętle – przykład – suma cyfr liczby

Przykład. Napisz program, który dla wczytanej z klawiatury liczby całkowitej oblicza sumę cyfr tej liczby

Przykład rozkład liczby na kolejne cyfry l.naturalnej dla `n=125`

```
125 % 10 = 5 -ostatnia cyfra
125 / 10 = 12-pozbywamy się ostatniej cyfry
12 % 10 = 2  -kolejna „ostatnia” cyfra
12 / 10 = 1  -pozbywamy się ostatniej cyfry
1 % 10 = 1   -kolejna „ostatnia” cyfra
1 / 10 = 0   -koniec
```

Algorytm obliczania sumy cyfr liczby n>=0:

Dopóki n>0 (są jeszcze cyfry)

- obliczamy ostatnia cyfra (%10) i dodajemy ją do sumy,
- pozbywamy się ostatniej cyfry (/10)

Pętle – przykład – suma cyfr liczby

Zauważmy, że dla liczb ujemnych operator % zwraca wartość ujemną *np. dla* `n=-125` mamy `-125 % 10 = -5`

Cyfry liczby powinny być nieujemne, zatem dla liczb ujemnych będziemy wykonywać obliczenia na ich wartości bezwzględnej:

```
int n, rob;
cout << "Podaj liczbe calkowita ";
cin  >> n;
//funkcja-wartosc bezwzględna abs():
rob = abs(n);  // należy dodać: #include <cmath>
//lub
rob = (n<0)? -n : n; //operator warunkwy
//lub instrukcja selekcji
if(n<0) rob = -n;
else rob = n;
```

Pętle – przykład – suma cyfr liczby

```
int main()
{
  int n;
  cout << "Podaj liczbe calkowita ";
  cin  >> n;
  //jeśli n<0 to n%10<0, a cyfry liczby>0, to rob = abs(n)
  int  rob = (n<0)? -n : n; //w rob  wartość bezwzględna z n
  int s = 0;   //w zmiennej s będziemy przechowywać sumę liczb
  while(rob > 0)  //dopóki są jeszcze cyfry
  {
    s += rob % 10;//do sumy dodajemy ostatnią cyfrę liczby rob
    rob /= 10;  //liczbę rob dzielimy przez 10,
                //skracamy w ten sposób ostatnia cyfrę
  }
  cout << "Suma cyfr liczby " << n << " wynosi " << s << endl;
  return 0;
}
```

Pętle – przykład – suma cyfr liczby

Output dla `n=-125` :

```
Podaj liczbe calkowita -125

Suma cyfr liczby -125 wynosi 8
```

Zauważmy, że dla `n=0`, pętla nie wykona się ani razu. Ale suma cyfr liczby `0` jest równa `0`, a tyle wynosi wartość zmiennej `suma` przed pętlą.

```
Podaj liczbe calkowita 0

Suma cyfr liczby 0 wynosi 0
```

Pętle – przykład – suma cyfr liczby

Przykład. Napisz program, który dla wczytanej z klawiatury liczby całkowitej oblicza ile cyfr ma wczytana liczba.

Zauważmy, że `n=0`, ma jedną cyfrę, zatem jeśli chcielibyśmy skorzystać z pętli z powyższego przykładu, należy najpierw sprawdzić czy wczytana liczba nie jest zerem (`if` przed pętlą `while`), lub użyć pętli `do-while`, która wykona się przynajmniej raz.

Pętle – przykład – liczba cyfr liczby- pętla while

```
int main()
{
  int n;
  cout << "Podaj liczbe calkowita n=";
  cin  >> n;
  int rob = abs(n);//można rob=n; bo tylko liczymy cyfry
  int ilcyfr = 0;//licznik cyfr liczby
  if ( rob == 0 )  //liczba n=0
    ilcyfr = 1;  //zero ma 1 cyfra
  else  //liczba !=0
    while (rob != 0)  //dopoki są cyfry
    {
        ilcyfr++;//rob!=0 ma cyfrę(y) -> zwiększamy licznik
        rob /= 10; //pozbywamy się już policzonej cyfry
    }
  cout << "Liczba " << n << " ma " << ilcyfr << " cyfr(y).";
  return 0;
}
```

Pętle – przykład – liczba cyfr liczby - pętla do-while

```
#include <iostream>
#include <cmath>  //bo używamy funkcji abs
using namespace std;
int main()
{
  int n;
  cout << "Podaj liczbe calkowita ";
  cin  >> n;
  int rob = abs(n);//można rob=n; bo tylko liczymy cyfry
  int il = 0; //zmienna - licznik ilości cyfr
  do
  {
     il++; //il = il+1;  il += 1; zwiększamy licznik
     //bo mamy pierwszą lub kolejna cyfrę liczby
     rob /= 10; //n = n/10;

  } while (rob != 0);
  cout << "Liczba cyfr liczby " << n << " to " << il ;
  return 0;
}
```

Pętle – przykład – pętla z licznikiem

Napisz program, który oblicza sumę 1/1+1/2 +1/3+...+1/n , gdzie n jest liczbą >0 podaną przez użytkownika (pętla zaporowa).

```
#include <iostream>
using namespace std;
int main()
{
  int n; //n ma być liczbą >0
  do
  {
     cout << "Podaj liczbe calkowita";
     cin  >> n;
  } while (n<=0);//((!(n>0))
```

Pętle – przykład – pętla z licznikiem

```
// 1/1 + 1/2 +1/3 + … + 1/n
double s = 0.0; //suma odwrotności l.rzeczywista
int k = 1; //dodatkowy licznik, który będzie
 //przechodził po kolejnych mianownikach od 1 do n
while(k <= n)  //dopóki są jeszcze odwrotnosci
{
   s += 1.0/k;//do sumy dodajemy kolejne składniki
              //(1.0 aby dzielenie było rzeczywiste,
              // a nie całkowite)
   k++;  //zwiększamy licznik – przechodzimy do
         // kolejnej odwrotnosci
}
cout << "Suma odwrotnosci wynosi" << s << endl;
return 0;
}
```

Operator przecinkowy

Operator przecinkowy jest dwuargumentowy: po dwóch stronach przecinka dwa wyrażenia

```
wyr1 , wyr2
```

Działanie jego polega na:

- obliczeniu wyrażenia ***wyr1*** i zignorowaniu rezultatu;
- obliczeniu wyrażenia ***wyr2***; jego wartość staje się wartością całego wyrażenia.

Często operator przecinkowy stosuje się w części inicjalizacyjnej lub inkrementującej nagłówka pętli **for**