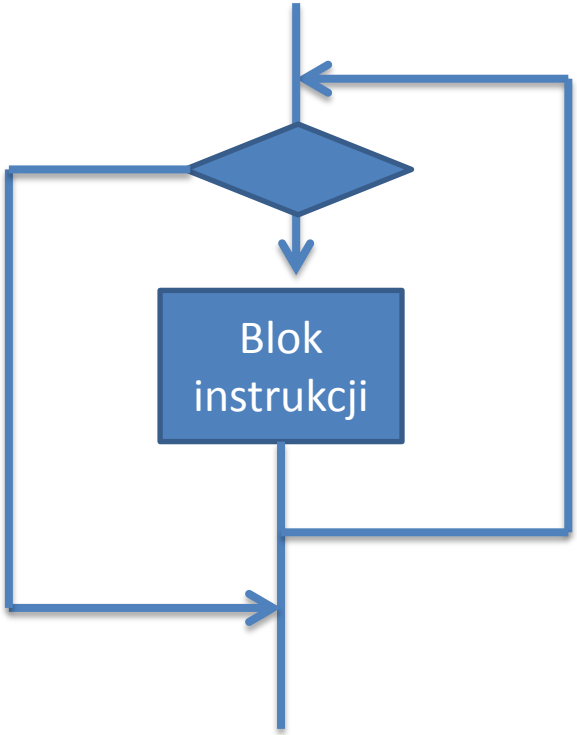
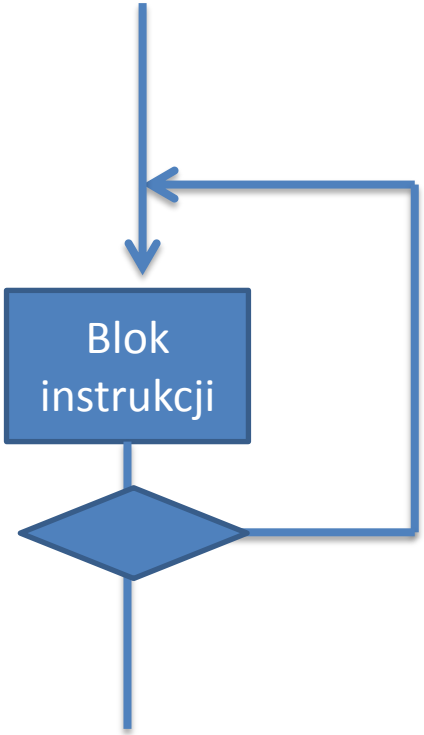
Programowanie C++

Wykład 4 (20.03.2017)

Pętle





- Warunek sprawdzany **przed** lub **po** iteracji

Pętle w różnych językach programowania (1)

Pascal

```
while number>0 do
begin
    sum := sum + number;
    number := number - 2;
end;
```

C++

```
while( a < 20 )
  {
      cout << “a: " << a << endl;
      a++;
   }
```

Python

```
while (count < 9):
    print 'The count is:', count
    count = count + 1
```

Pętle w różnych językach programowania (2)

Pascal

```
repeat
    sum := sum + number;
    number := number - 2;
until number<0;
```

C++

```
do {
    cout << "a: " << a << endl;
    a++;
  }
while ( a < 20 )
```

Python – brak odpowiednika

Pętle for

Pascal

```pascal
for i:=0 to 10 do
begin
  writeln('a: ', a);
end;
```

C++

```cpp
int sum = 0;

for (int i = 1; i < 6; i++)
{
    sum += i;
}
```

Pętle for

```
int sum = 0;

for (int i = 1; i < 6; i++)
{
   sum = sum + i;
}
```

```
int sum = 0;
int i = 1;

while (i < 6)
{
   sum = sum + i;
   i++;
}
```

```
int sum = 0;
int i = 1;

cout << "Podaj 2 liczby: ";
cin >> i;
cin >> j;
cout << i + j;

i = 0;
while (i < 6)
{
   sum = sum + i;
   i++;
}
```

Niejednoznaczność roli zmiennej *i*

Pętle *for* dla zakresu

```python
for i in range(1,10):
    print(i)
```

Python

```php
$tablica = array(1, 2, 3, 4);

foreach ($tablica as $i)
{
    print($i);
}
```

PHP

```cpp
for (int i : tablica )
{
    cout << i << endl;
}
```

C++

Dwie zmienne o tej samej nazwie

```
int i = 0;
double i = 0.5;

cout << i << endl;
```

```
int i = 0;

{
  double i = 0.5;

  cout << i << endl;
}
Cout << i << endl;
```

Zmienna lokalna dla zakresu

```
int i;

for(i = 0; i < 3; i++)
{
  for(i = 0; i < 5; i++)
  {
     cout << i << endl;
  }
}
```

```
for(int i = 0; i < 3; i++)
{
  for(int i = 0; i < 5; i++)
  {
     cout << i << endl;
  }
}
```

Zmienna lokalna dla zakresu

```
int i;

for(i = 0; i < 3; i++)
{
  for(i = 0; i < 5; i++)
  {
     cout << i << endl;
  }
}
```

```
for(int i = 0; i < 3; i++)
{
  for(int j = 0; j < 5; j++)
  {
     cout << j << endl;
  }
}
```

Funkcja

Argumenty → **Wynik**

zmienne lokalne

```
double srednia(double x, double y)
{
    return (x+y) / 2;
}
```

Definiowanie funkcji z argumentami

Argumenty funkcji

1. Definiuje się w nagłówku funkcji, w nawiasie okrągłym za nazwą.
2. Ich definicje rozdziela się przecinkami.
3. Są definiowane podobnie, jak zmienne wewnątrz funkcji.
4. Mogą być wykorzystywane jako zmienne wewnątrz funkcji.

Przykład – funkcja z jednym argumentem

```
double f(double x)
{
    return x*x + 1;
}
```

Zwraca wynik będący kwadratem jej argumentu zwiększonym o 1.

Struktura programu

Włączenie pliku nagłówkowego biblioteki

Używamy *przestrzeni nazw* std

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

int main() {

   int a = 10;
   a++;
   cout << "Hello world\n";
   cout << a;

   return 0;
}
```

Funkcja main()

Przekazywanie wartości argumentów do funkcji

Podczas wykonywania programu

1. Tworzone są zmienne, których nazwy i typy danych odpowiadają nazwom i typom danych argumentów funkcji.
2. Wartości znajdujące się na pozycjach odpowiadających argumentom w zapisie wywołania funkcji (np. `suma += sin(a);`) stają się początkowymi wartościami tych zmiennych.
3. Wykonywany jest kod reprezentowany przez instrukcje w treści funkcji, odwołujący się do tych zmiennych.
4. Wykonanie tego kodu może skończyć się wygenerowaniem wyniku, który jest wykorzystywany w sposób określony przez kod wywołujący funkcję.

Przekazywanie wartości argumentów do funkcji – przykład

Całka trapezowa dla funkcji `f()` z poprzedniego przykładu.

```
double trapez(double a, double b, int N)
{
    double delta, suma, x_j;
    int j;

    suma = (f(a) + f(b)) / 2;
    delta = (b - a) / N;
    for (j = 1, x_j = a; j < N; j++) {
        x_j += delta;
        suma += f(x_j);
    }
    suma *= delta;
    return suma;
}
```

```
double srednia(double x, double y)
{
    return (x+y) / 2;
}

double srednia(double x)
{
    return x;
}


double srednia(double x, double y,
                                double z)
{
    return (x+y+z) / 3;
}
```

```
// Funkcja zwraca srednia arytmetyczna lub
geometryczna
double srednia(double x, double y,
                        bool arytmetyczna = true)
{
  if(arytmetyczna)
        return (x+y)/2;
  else
  {
      if(x < 0)
            return 0;
      if(y < 0)
            return 0;
      return sqrt(x*y);
  }
}
```

Cout << srednia(4,5);

Funkcje i procedury

```
Procedure DrawLine;
Var Counter : Integer;
Begin
    textcolor(green);
    For Counter := 1 to 10 do
    Begin
        Write(chr(196));
    End;
End;
```

Pascal

```
Function PythagorasFunc(A: Real; B: Real) : Real;
Begin
    PythagorasFunc := SQRT(A*A + B*B);
End;
```

Procedura w C++

```
void wypisz(int x) {
   cout << x << endl;
   return;
}
```