Programowanie C++

Wykład 1 (02.03.2016)

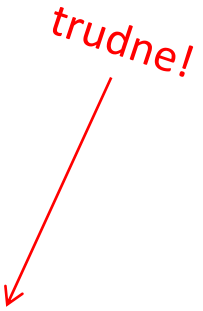
Sprawy organizacyjne

- Kontakt: Tomasz.Kazimierczuk@fuw.edu.pl
- 1h wykładu + 2h ćwiczeń / tydzień
- 3 kolokwia na ćwiczeniach + test ostatnim na wykładzie (4 x 25%)
- Uwaga: na kolokwium program nie kompilujący się dostaje 0 punktów
- Strona do wykładu (w tym zadania do sprawdzenia): http://www.fuw.edu.pl/~tkaz/teaching/programowanie2016/

Wikipedia

- **Programowanie komputerów** – proces projektowania, tworzenia, testowania i utrzymywania kodu źródłowego programów komputerowych (…)

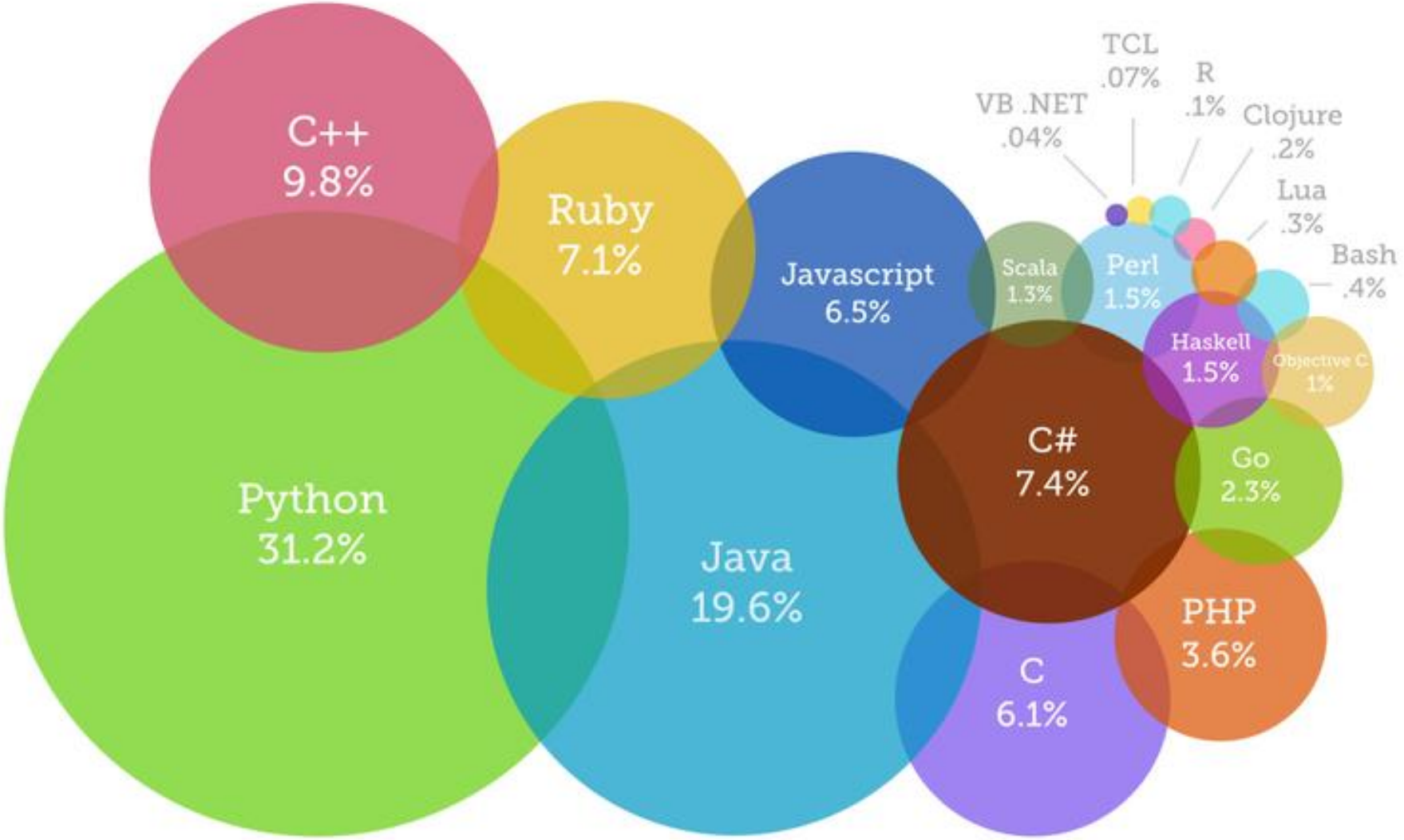
Wikipedia



- **Programowanie komputerów** – proces projektowania, tworzenia, testowania i utrzymywania kodu źródłowego programów komputerowych (…)

Te zajęcia służą nabraniu wprawy w posługiwaniu się językiem tak, aby nie przeszkadzał w programowaniu

Most Popular Coding Languages of 2015



C++ jest językiem kompilowanym

1. Tworzymy kod źródłowy w pliku tekstowym

```cpp
#include <iostream>
using namespace std;
int main() {

  cout << "Hello world";

  return 0;
}
```

program.cpp

2. Kompilujemy program

```
$ g++ program.cpp -o program
```

3. Uruchamiamy program

```
$ ./program
Hello world
```

Logowanie na Primusa

Dom —— ssh ——> primus.okwf.fuw.edu.pl

putty 

```
ssh login@tempac.fuw.edu.pl
ssh login@primus.okwf.fuw.edu.pl
```

Kompilator: g++

Edytor: nano, pico

Programowanie imperatywne

- Kolejne instrukcje zmieniają stan (np. wartości zmiennych)

- Podstawowe typy zmiennych:

  `int` – liczba całkowita, np. 2235

  `double` – liczba rzeczywista, np. 3.14

  `char` – pojedynczy znak ASCII (litera)

  `bool` – wartość logiczna (prawda lub fałsz)

Prześledźmy program:

Standardowy początek

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

int main() {
   int a = 0;
   int b = 0;

   a = 8;
   b = 9 + a;
   b = b + b;


   cout << a << " " << b << "\n";

   return 0;
}
```

Wypisywanie wartości zmiennych

Standardowy koniec

Wyrażenia w C++

Operator: + (np. a + b)

Wynik: suma wartości *a* i *b*

Efekt uboczny: brak

Operator: = (np. a = b)

Wynik: wartość *b*

Efekt uboczny: zmienna *a* przyjmuje wartość *b*

Operator: ++ (np. a++)

Wynik: wartość *a*

Efekt uboczny: zmienna *a* zwiększa się o 1

a = 3; b = 5;

b = (a = (a++))

Ile wynosi a?
Ile wynosi b?

b = a = b+a++

A teraz?

Precedence	Operator	Description	Associativity
1	::	Scope resolution	Left-to-right
2	a++ a--	Suffix/postfix increment and decrement	
	type() type{}	Functional cast	
	a()	Function call	
	a[]	Subscript	
	. ->	Member access	
3	++a --a	Prefix increment and decrement	Right-to-left
	+a -a	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(type)	C-style cast	
	*a	Indirection (dereference)	
	&a	Address-of	
	sizeof	Size-of[note 1]	
	new new[]	Dynamic memory allocation	
	delete delete[]	Dynamic memory deallocation	
4	.* ->*	Pointer-to-member	Left-to-right
5	a*b a/b a%b	Multiplication, division, and remainder	
6	a+b a-b	Addition and subtraction	
7	<< >>	Bitwise left shift and right shift	
8	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
9	== !=	For relational operators = and ≠ respectively	
10	a&b	Bitwise AND	
11	^	Bitwise XOR (exclusive or)	
12	|	Bitwise OR (inclusive or)	
13	&&	Logical AND	
14	||	Logical OR	
15	a?b:c	Ternary conditional[note 2]	Right-to-left
	throw	throw operator	
	=	Direct assignment (provided by default for C++ classes)	
	+= -=	Compound assignment by sum and difference	
	*= /= %=	Compound assignment by product, quotient, and remainder	
	<<= >>=	Compound assignment by bitwise left shift and right shift	
	&= ^= |=	Compound assignment by bitwise AND, XOR, and OR	
16	,	Comma	Left-to-right

Typ wyniku wyrażenia

- W wyrażeniu arytmetycznym mogą spotkać się zmienne różnych typów, np.

```
int a = 7;
double b = 5;

a + b    // jaki to ma typ?
```

- Wynik ma taki typ, jak ogólniejsza ze zmiennych (tu: double).
- Operator rzutowania zmienia wyrażenia

```
cout << 14 / 5 ;                // Wynik: 2
cout << ((double) 14) / 5;      // Wynik: 2.8
cout << (double) (14 / 5);      // Wynik: 2.0
```

Porównywanie

- ==, !=, >, <, >=, <=

- Wykorzystywane głównie w instrukcjach kontrolnych, np.

```
if (a == b)
    cout << "Zmienne a i b sa rowne";


while (a > b)
{
    b++;
    a--;
}
```

Algorytm Euklidesa

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

int main() {
  int a, b;
  cout << "Podaj dwie liczby naturalne";
  cin >> a >> b;

  int reszta;
  while(b != 0)
  {
      reszta = a % b;
      a = b;
      b = reszta;
  }

  cout << "Ich NWD wynosi " << a << "\n";
  return 0;
}
```