

## **Wykład I**

### **Języki programowania**

### **Algorytmy – metody prezentacji i zapisu**

**[www.bartoszewski.uniwersytetradom.pl](http://www.bartoszewski.uniwersytetradom.pl)**



## Rodzaje języków programowania

---

Języki programowania możemy podzielić ze względu na:

- Poziom (języki niskiego i wysokiego poziomu),
- Paradygmat programowania (strukturalny, obiektowy, sterowany zdarzeniami itp.),
- Sposób kontroli typów zmiennych (ściśle i luźno typowane)
- Sposób wykonywania (kompilacja, interpretacja),
- Przeznaczenie,
- Platforma sprzętowa i programowa,



## Rodzaje języków programowania

---

**Poziom** (języki niskiego i wysokiego poziomu) - Języki programowania różnią się stopniem abstrakcji od sprzętu komputerowego:

### 1. Języki niskiego poziomu:

- Są bliższe językowi maszynowemu, czyli instrukcjom bezpośrednio wykonywanym przez procesor. Asembler, język maszynowy. Zapewniają dużą kontrolę nad sprzętem
- **Zalety:** Wysoka wydajność, precyzyjna kontrola nad zasobami.
- **Wady:** Kod jest trudny do zrozumienia i podatny na błędy.

### 2. Języki wysokiego poziomu:

- Bardziej abstrakcyjne, bliższe językowi naturalnemu, łatwiejsze w użyciu.
- **Zalety:** Łatwość pisania i utrzymania kodu, przenośność między platformami.
- **Wady:** Mniejsza kontrola nad sprzętem, czasem niższa wydajność.



## Paradygmat programowania:

1. **Strukturalny:** Kod jest zorganizowany w bloki (procedury, funkcje), z naciskiem na sekwencyjność i unik(PDO) modularność. (Pascal, c)
2. **Obiektowy:** Kod opiera się na obiektach, które łączą dane (pola) i zachowanie (metody). (c++, c#, Python, Java)
3. **Sterowany zdarzeniami:** Program reaguje na zdarzenia (np. kliknięcia, naciśnięcia klawiszy) (wszystkie systemy obsługujące GUI)
4. **Funkcyjny:** Programowanie opiera się na funkcjach matematycznych, unika stanów i efektów ubocznych (Haskell, Scala)
5. **Logiczny:** Opiera się na logice formalnej i regułach wnioskowania. (Prolog)
6. **Deklaratywny:** w którym programista określa, co ma być osiągnięte (wynik lub cel), zamiast szczegółowo opisywać, jak to zrobić (kroki proceduralne). (np. SQL),



## Sposób kontroli typów zmiennych

### 1. Ściśle typowane (statycznie typowane):

Typy zmiennych muszą być zdefiniowane w czasie kompilacji, a ich zmiana jest trudna lub niemożliwa. (C, C++, Java)

### 2. Luźno typowane (dynamicznie typowane):

Typy zmiennych są określane w czasie wykonywania, co pozwala na większą elastyczność. (Python, JavaScript, PHP).



## Rodzaje języków programowania

---

### Sposób wykonywania

1. **Kompilowane** – podczas kompilacji kod źródłowy jest tłumaczony na kod maszynowego, czyli kod binarny przeznaczony bezpośrednio do wykonania przez procesor. Powstają pliki wykonywalne (np. pliki .exe w systemie Windows)
2. **Interpretowane** – kod źródłowy jest tłumaczony na bieżąco i wykonywany przez dodatkowy program zwany interpreterem (środowiskiem uruchomieniowym). Przykładem języka interpretowanego jest JavaScript, dla którego środowiskiem uruchomieniowym jest przeglądarka internetowa.

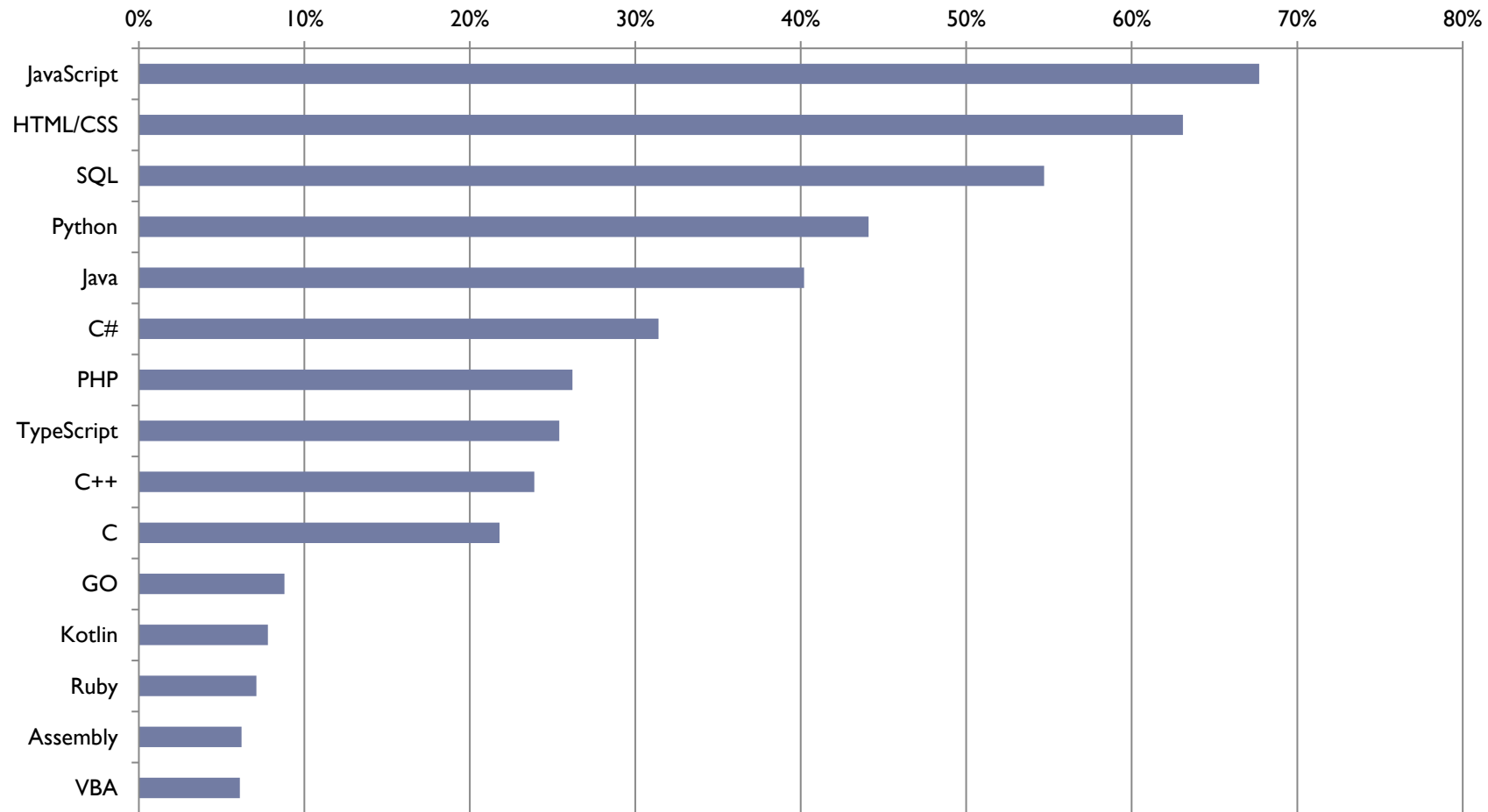
Kompilacja zapewnia najwyższą wydajność programom, lecz wygenerowany plik jest ściśle powiązany z platformą sprzętową.

Ogólnie - kompilowane języki są bardziej zbliżone do sposobu funkcjonowania sprzętu, przez co programowanie w nich jest trudniejsze. Języki interpretowane zapewniają większą przenośność programów, które często są niezależne od platformy i systemu operacyjnego.

Języki, w których nie da się realizować obliczeń - języki znaczników takie jak HTML czy XML), nie są zazwyczaj uznawane za języki programowania.

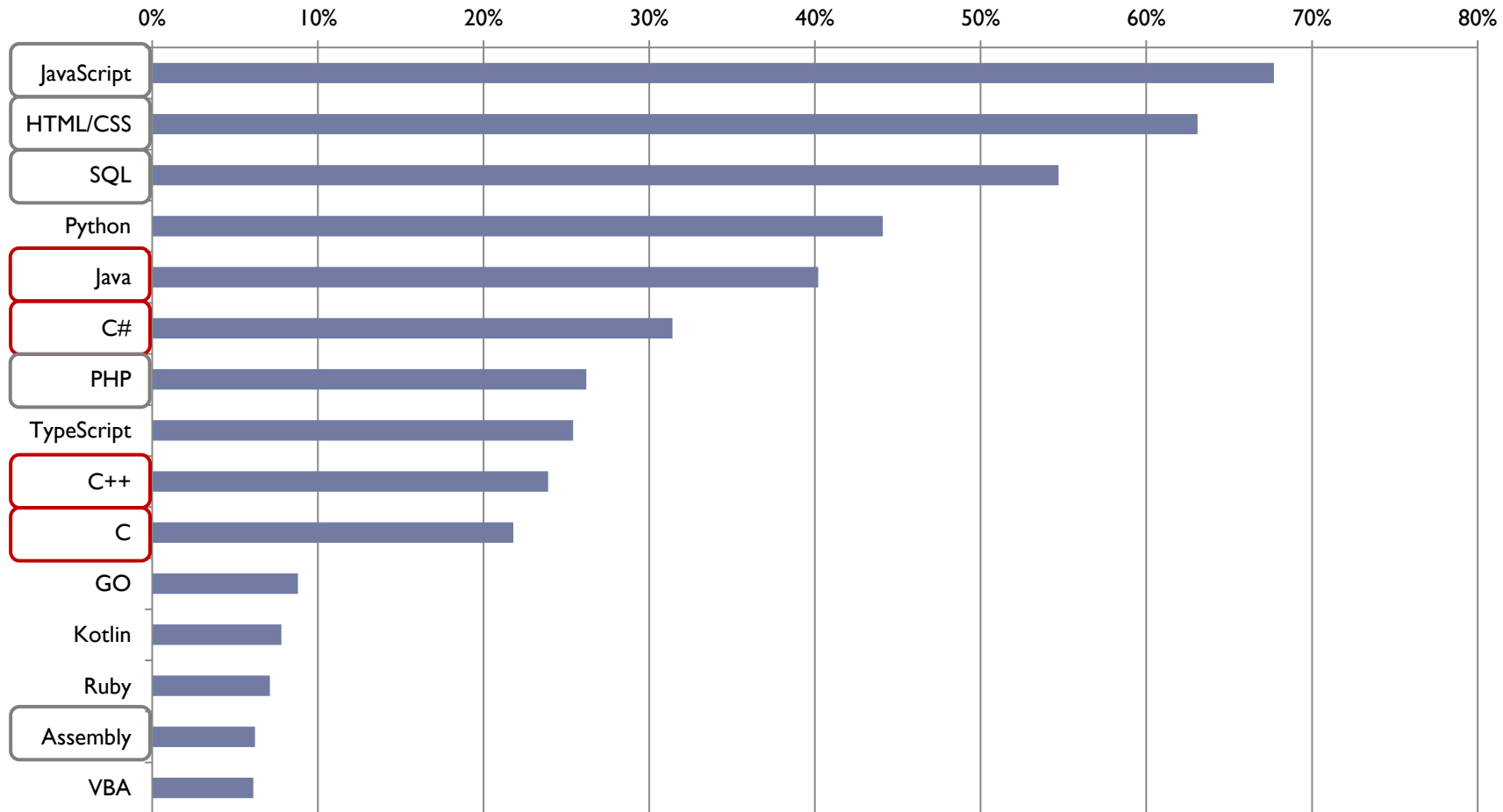


## Popularność języków programowania



<https://insights.stackoverflow.com/survey/2020>

## Popularność języków programowania



<https://insights.stackoverflow.com/survey/2020>





**Kod źródłowy** – program (plik testowy) napisany w języku programowania, czyli w języku algorytmicznym - czytelny dla programisty

**Kod wynikowy** - program zapisany jako ciąg rozkazów i danych w kodzie maszynowym procesora (w postaci czytelnej dla komputera), najczęściej w postaci binarnej.

**Proces tworzenia programu dla języków C / C++:**

- |               |   |
|---------------|---|
| 1. edytor     | - ( *.cpp ) kod źródłowy                          |
| 2. kompilator | - ( obj ) kod wynikowy                            |
| 3. Linker     | - ( *.exe ) kod wynikowy połączony z bibliotekami |

Opcjonalnie:

- debugger - śledzenie działania programu krok po kroku,



W skład środowiska programistycznego może wchodzić:

1. Edytor kodu,
2. Kompilator,
3. Narzędzia do testowania (debugowania),
4. System kontroli wersji,
5. Dla niektórych języków również środowisko uruchomieniowe niezbędne do uruchomienia programu (.NET, Wirtualna Maszyna Javy, itp.)



## Środowiska programistyczne dla języków C / C++

**Microsoft Visual C++** – zaawansowane IDE do zastosowań profesjonalnych (darmowe w wersji Community)



**Visual Studio Code** – nowoczesny, bardzo popularny i wszechstronny edytor programistyczny z ogromną bazą wtyczek rozszerzających jego możliwości. Wymaga zewnętrznego kompilatora C/C++ (np. MINGW) (darmowy)



**Code::Blocks** – proste, lecz już przestarzałe IDE dla C/C++, dobre na początek nauki (darmowe)



**Dev-C++** – mocno przestarzałe, lecz wciąż jeszcze spotykane w szkołach IDE dla C/C++ (darmowe)

## Kompilatory języka C / C++



**GCC** - GNU Compiler Collection – zestaw kompilatorów o otwartym kodzie źródłowym – najpopularniejszym portem GCC dla Windows jest **MINGW**



**Clang** – kompilator języków C, C++ oraz Objective-C,. Prace nad nim sponsorowane są przez Apple.



**MS Visual C++** – IDE posiadające własny kompilator języków C i C++

<https://isocpp.org/get-started>



# C++ Budowa programu

# Budowa programu

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      return 0;
8  }
```

Dołączanie plików nagłówkowych bibliotek

Przestrzeń nazw

return...; gdzie za kropki wstawiamy dowolną liczbę - kod wyjścia programu.

Funkcja main() - główna funkcja programu

## Pierwszy program (w języku C)

```
#include <conio.h>
```

```
#include <stdio.h>
```

```
using namespace std;
```

```
int main()
```

```
{
```

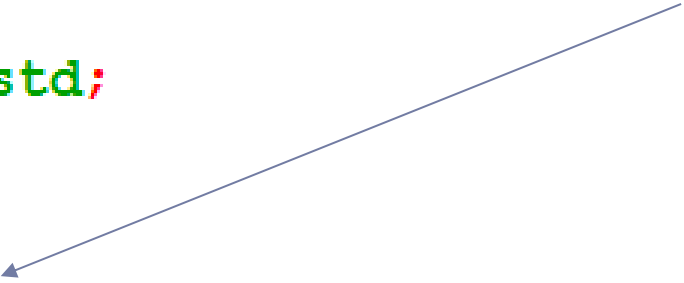
```
    printf("To nasz pierwszy program w C\n");
```

```
    getch();
```

```
    return 0;
```

```
}
```

Charakterystyczna dla  
języka C funkcja printf()



## Pierwszy program (w języku C++)

---

```
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    getch(); //zatrzymuje działanie programu
    return 0;
}
```





**int main( )**

- ✓ W języku C i C++ nie ma „programu głównego” jest za to funkcja o nazwie `main( )` która wykonywana jest zawsze jako pierwsza.
- ✓ Każdy program musi posiadać funkcję `main( )`



W językach C i C++ mamy do dyspozycji trzy rodzaje komentarzy:

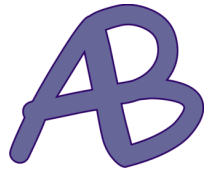
- komentarz jednowierszowy;

```
int main()  
{  
    // Komentarz jednowierszowy  
    return 0;  
}
```

- komentarz wielowierszowy;

```
int main()  
{  
    /* początek komentarza  
       komentujemy dalej....  
       i kończymy komentarz */  
    return 0;  
}
```

- komentarz wykonany za pomocą dyrektyw preprocesora.



### Zmienne (wstęp)

Zmienna to nazwa symboliczna reprezentująca fragment pamięci, który przechowuje dane określonego typu.

Języki c i c++ są ściśle typowane (statycznie typowane).

Typy zmiennych muszą być zdefiniowane w czasie kompilacji, a ich zmiana jest niemożliwa.



Nazwa typu	Zawartość	Przedział wartości	Zajęt. pamięć
<b>char</b>	znak	$-128 \div 127$	1 bajt
<b>int</b>	liczba całkowita	$-32768 \div 32767$	2 bajty
<b>long</b>	liczba całkowita	$-2147\text{mln} \div 2147\text{mln}$	4 bajty
<b>float</b>	liczba rzeczyw.	$10^{-38} \div 10^{38}$ (7cyfr)	4 bajty
<b>double</b>	liczba rzeczyw.	$10^{-308} \div 10^{308}$ (15 cyfr)	8 bajtów

**Modyfikatory typu:**

<b>signed</b>	→	ze znakiem ( $\pm$ ),	<b><u>int</u></b>	<b><u>char</u></b>	—
<b>unsigned</b>	→	bez znaku,	<b>int</b>	<b>char</b>	—
<b>short</b>	→	krótka (mniejsza),	<b>int</b>	—	—
<b>long</b>	→	długa (większa)	<b><u>int</u></b>	—	<b>double</b>

np. **unsigned long int** *dluga\_liczba\_bez\_znaku* ;

**Wartości domyślne:**

<b>long</b>	=	<b>long int</b>
<b>int</b>	=	<b>signed int</b>
<b>char</b>	=	<b>signed char</b>



***Deklaracja zmiennej*** - informuje kompilator, że dana nazwa jest znana. Jednak pamięć dla obiektu nie zostaje przydzielona. Do obiektu nie możemy się odwoływać, nie możemy mu przypisywać wartości – obiekt jeszcze nie istnieje.

**extern nazwaTypu nazwaZmiennej;**

Np.: extern int liczba;



*Definicja zmiennej* - rezerwuje miejsce w pamięci dla danej zmiennej. Po zdefiniowaniu ze zmiennej możemy korzystać.

*nazwaTypu* *nazwaZmiennej*;

Np.: int liczba;

Każda definicja jest jednocześnie deklaracją (ale nie odwrotnie).



*Inicjalizacja zmiennej* - polega na przypisaniu wartości do danej zmiennej w momencie jej deklaracji

*nazwaTypu nazwaZmiennej = wartość;*

Np.: `int liczba = 10;`





- Deklaracja** Informuje kompilator o istnieniu zmiennej: `extern int x;`
- Definicja** Rezerwuje pamięć: `int x;`
- Inicjalizacja** Nadaje wartość początkową: `int x = 5;`

## enum (typ wyliczeniowy)

**Typ wyliczeniowy enum** (ang. enumeration) to typ danych, który pozwala definiować zmienne mogące przyjmować jedną z ustalonych, nazwanych wartości całkowitych. Ułatwia to pisanie czytelnego i bezpiecznego kodu, szczególnie w sytuacjach, gdy zmienna może przyjmować ograniczony zestaw znanych stanów.

```
enum DzieńTygodnia
```

```
{  
    Poniedziałek,  
    Wtorek,  
    Sroda,  
    Czwartek,  
    Piątek,  
    Sobota,  
    Niedziela  
};
```

Domyślnie:

- Poniedziałek = 0,
- Wtorek = 1,...,
- itd.

Można jawnie  
przypisać wartości:

```
enum Status
```

```
{  
    OK = 1,  
    Error = -1,  
};
```

## enum (typ wyliczeniowy)



```
enum DzieńTygodnia
{
    Poniedziałek,
    Wtorek,
    Sroda,
    Czwartek,
    Piątek,
    Sobota,
    Niedziela
};
```

```
DzieńTygodnia dzisiaj;
```

```
dzisiaj = Sroda; // Przypisanie wartości odpowiadającej etykietcie Środa (2)
```

```
dzisiaj = 123; // To jest błąd, ponieważ 123 nie jest poprawną etykietą
```

```
dzisiaj = "Sroda"; // To jest błąd, ponieważ "Sroda" to łańcuch znaków a nie nazwa etykiety
```

```
cout << dzisiaj << endl;
```

```
    // Wypisze wartość enum odpowiadającą etykietcie Sroda, czyli 2
```

```
cout << DzieńTygodnia::Wtorek << endl;
```

```
    // Wypisze wartość enum odpowiadającą etykietcie Wtorek, czyli 1
```



**Stała** - jest typem zmiennej, dla której zabroniliśmy zmieniać wartości po jej zainicjalizowaniu. Zmienna tylko do odczytu.

```
const    nazwaTypu nazwaZmiennej = wartość;
```

Np.: `const int liczba = 10;`

Stałe muszą być za zainicjalizowane - inaczej nie ma to żadnego praktycznego sensu



## Operacje we/wy

## Klasy cout i cin (obiektowo w C++)

---

**Strumień** – to najprościej mówiąc jest to ciąg bajtów o nieokreślonej długości przesyłany pomiędzy nadajnikiem a odbiornikiem.

Wyróżniamy trzy rodzaje strumieni:

1. **Strumień konsoli – wczytanie z klawiatury i wypisanie na ekran**
2. Strumień plikowe
3. Strumień napisów

Do obsługi strumieni służą obiekty **cin** oraz **cout**

Domyślnym strumieniem jest strumień konsoli, którym będziemy posługiwać się w tym wykładzie.

## Klasy cout i cin (obiektowo w C++)

---

Wyprowadzenie wartości do strumienia wyjściowego (stdout)

```
cout << „tekst”;
```

```
cout << zmienna;
```

Wczytanie ze strumienia wejściowego (stdin)

```
cin >> zmienna;
```

Prototypy cin i cout znajdują się w bibliotece iostream.h

```
#include <iostream>
```



## Klasy cout i cin (obiektoowo w C++)

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hej tam.\n";
    cout << "To jest 5: " << 5 << "\n";
    cout << "Manipulator endl ";
    cout << "wypisuje nowa linie na ekranie.";
    cout << endl;
    cout << "To jest bardzo duza liczba:\t" << 70000;
    cout << endl;
    cout << "To jest suma 8 i 5:\t";
    cout << 8+5 << endl;
    cout << "To jest ulamek:\t\t";
    cout << (float) 5/8 << endl;
    cout << "I bardzo, bardzo duza liczba:\t";
    cout << (double) 7000 * 7000 << endl;
    return 0;
}
```

```
Hej tam.
To jest 5: 5
Manipulator endl wypisuje nowa linie na ekranie.
To jest bardzo duza liczba:    70000
To jest suma 8 i 5:            13
To jest ulamek:                0.625
I bardzo, bardzo duza liczba:    4.9e+007
```





# Instrukcje sterujące



W klasycznym języku C nie ma osobnych zmiennych przechowujących dane typu prawda-fałsz.

Tę rolę pełnić może każda zmienna, wyrażenie lub funkcja, która przyjmuje (lub zwraca) wartość zero lub różną od zera.

Wartość zero - > FAŁSZ

Wartość inna niż zero - > PRAWDA

```
int x = 5;
if (x)
{
    cout<<"x jest różne od zera\n";
}
```



W języku C++ dodana nowy wbudowany typ podstawowy - **bool**

```
bool a = true; // logiczna prawda
```

```
bool b = false; // logiczne fałsz
```

Można przypisywać inne typy, które zostaną przekonwertowane do bool

```
bool x = 5; // x == true
```

```
bool y = 0; // y == false
```

```
bool ok = true;  
if (ok)  
{  
    cout << "OK jest true";  
}
```

## Instrukcja warunkowa if

```
if (wyrażenie) instrukcja;
```

```
if (wyrażenie) instrukcja_1;  
else instrukcja_2;
```

```
if (wyrażenie)  
{  
    instrukcja_1;  
    instrukcja_2;  
}  
else instrukcja_3;
```



## Instrukcja warunkowa if

```
if (warunek)
{
    // blok kodu wykonywany, gdy warunek jest prawdziwy
}
```

```
if (warunek)
{
    // gdy warunek prawdziwy
}
else
{
    // gdy warunek fałszywy
}
```



## Wybór z 3 możliwości

```
if (warunek_1)
{
    // blok kodu wykonywany, gdy warunek_1 jest prawdziwy
}
else if (warunek_2)
{
    // blok kodu wykonywany, gdy warunek_1 jest fałszywy, a warunek_2 jest prawdziwy
}
else
{
    // blok kodu wykonywany, gdy oba warunki są fałszywe
}
```



## Instrukcja warunkowa if - przykład

### Równanie kwadratowe – wersja 1

```
int main()
{
    int a,b,c,delta;
    cout <<"a=";
    cin >> a;
    cout <<"b=";
    cin >> b;
    cout <<"c=";
    cin >> c;
    delta = b*b-4*a* c;
    if (delta >= 0)
    {
        double x1= (-b - sqrt(delta)) /(2*a);
        double x2= (-b + sqrt(delta)) /(2*a);
        cout << "x1=" << x1 << endl << "x2=" << x2;
    }
    else
    {
        cout <<"Brak zozwiazan";
    }
    return 0;
}
```



## Instrukcja warunkowa if - przykład

```
int main()
{
    int a,b,c,delta;
    cout <<"a="; cin >> a;
    cout <<"b="; cin >> b;
    cout <<"c="; cin >> c;
    delta = b*b-4*a* c;
    if (delta > 0)
    {
        double x1= (-b - sqrt(delta)) /(2*a);
        double x2= (-b + sqrt(delta)) /(2*a);
        cout << "x1=" << x1 << endl << "x2=" << x2;
    }
    else if (delta == 0)
    {
        double x0 = (double)(-b)/(2*a);
    }
    else
    {
        cout <<"Brak zozwiazan";
    }
    return 0;
}
```

Równanie kwadratowe –  
wersja 2





## Operator trójargumentowy ?:

---

warunek ? wyrażenie\_jeśli\_prawda : wyrażenie\_jeśli\_fałsz ;

Przykład:

```
cout << "Podaj liczbę punktów: ";  
cin >> punkty;  
punkty >= 10 ? cout << "Zaliczone" : cout << "Do poprawy";
```

**Porada:** Nie nadużywaj operatora ?: jeśli warunek i wyrażenia są złożone.



## Literatura:

---

W prezentacji wykorzystano przykłady i fragmenty:

- Grębosz J. : ***Symfonia C++, Programowanie w języku C++ orientowane obiektowo***, Wydawnictwo Edition 2000.
- Jakubczyk K.: *Turbo Pascal i Borland C++ Przykłady*, Helion.

Warto zajrzeć także do:

- Sokół R. : ***Microsoft Visual Studio 2012 Programowanie w Ci C++***, Helion.
- Kernighan B. W., Ritchie D. M.: ***język ANSI C***, Wydawnictwo Naukowo Techniczne.

Dla bardziej zaawansowanych:

- Grębosz J. : ***Pasja C++***, Wydawnictwo Edition 2000.
- Meyers S.: ***język C++ bardziej efektywnie***, Wydawnictwo Naukowo Techniczne