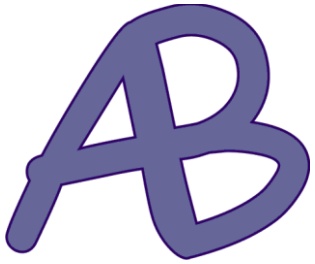


Wykład I

Języki programowania
Algorytmy – metody prezentacji i zapisu

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,
- Sposób wykonywania (kompilacja, interpretacja),
- Przeznaczenie,
- Platforma sprzętowa i programowa,

Rodzaje języków programowania

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)

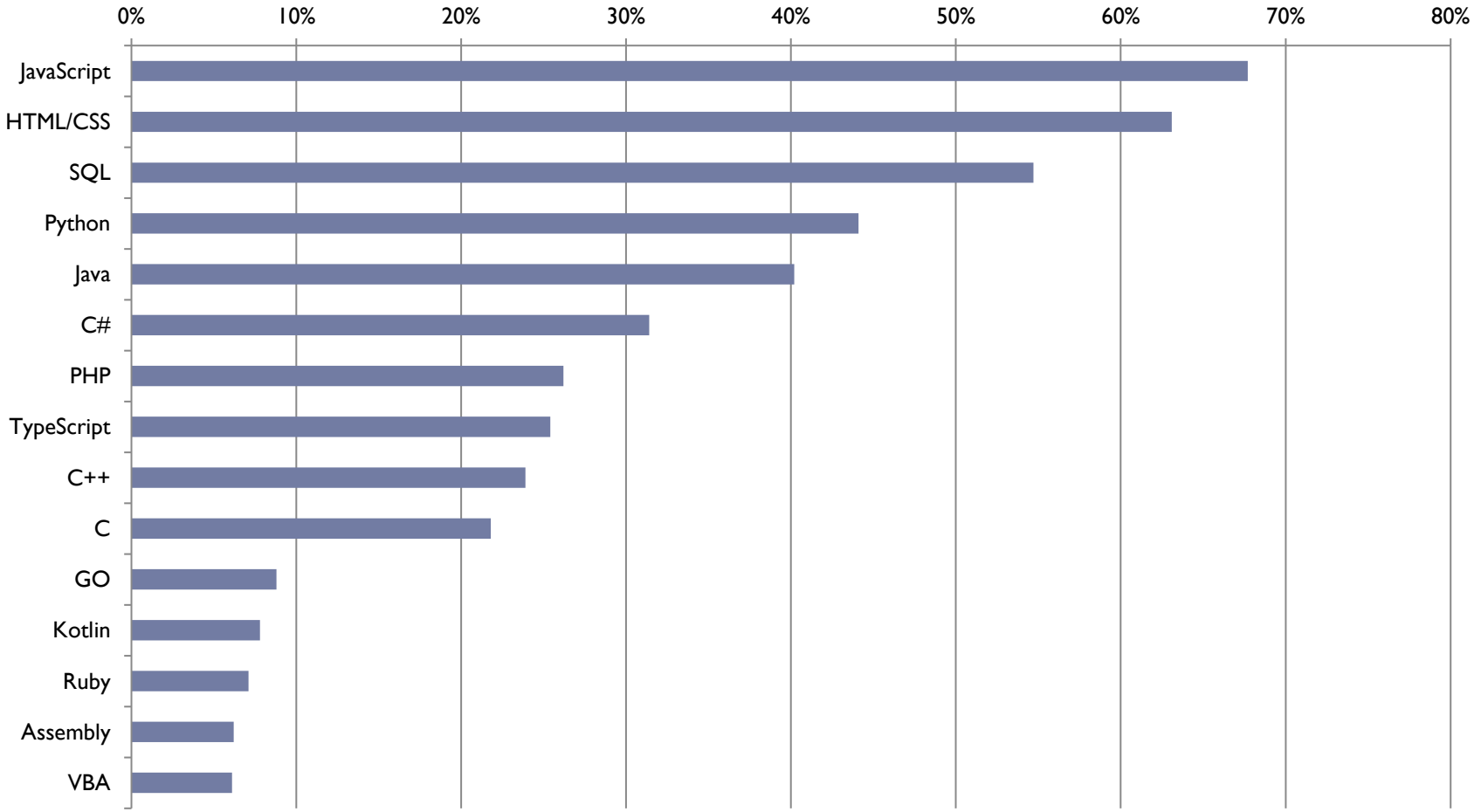
Interpretowane – kod źródłowy jest tłumaczony na bieżąco i wykonywany przez dodatkowy program zwany interpreterem (środowiskiem uruchomieniowym). Przykładem języka interpretowanego jest JavaScript, dla którego srodowiskiem uruchomieniowym jest przeglądarka internetowa.

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.

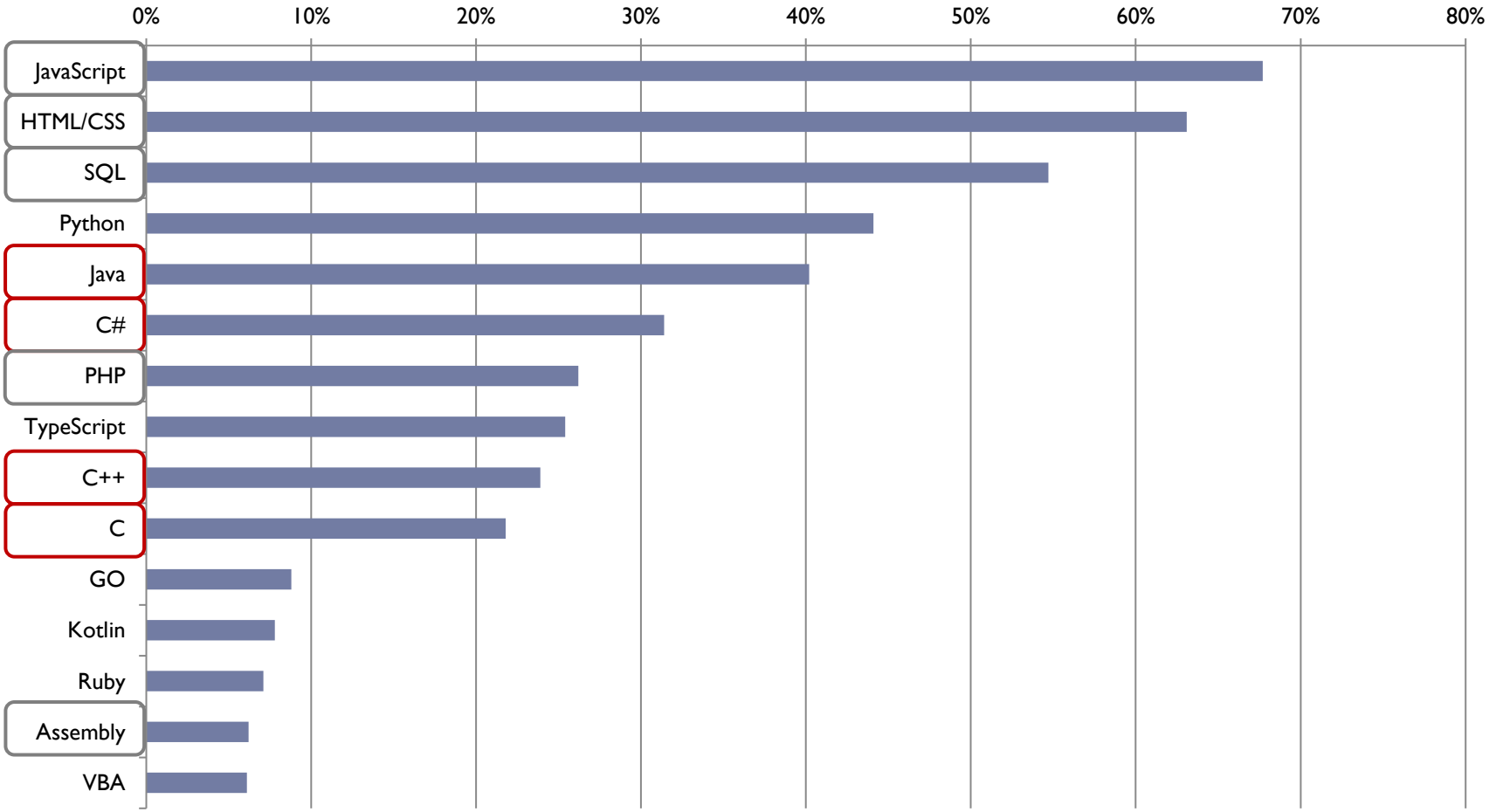
Języki, w których nie da się realizować obliczeń - języki znaczników takie jak HTML czy XML), nie są zazwyczaj uznawane za języki programowania.

Popularność języków programowania



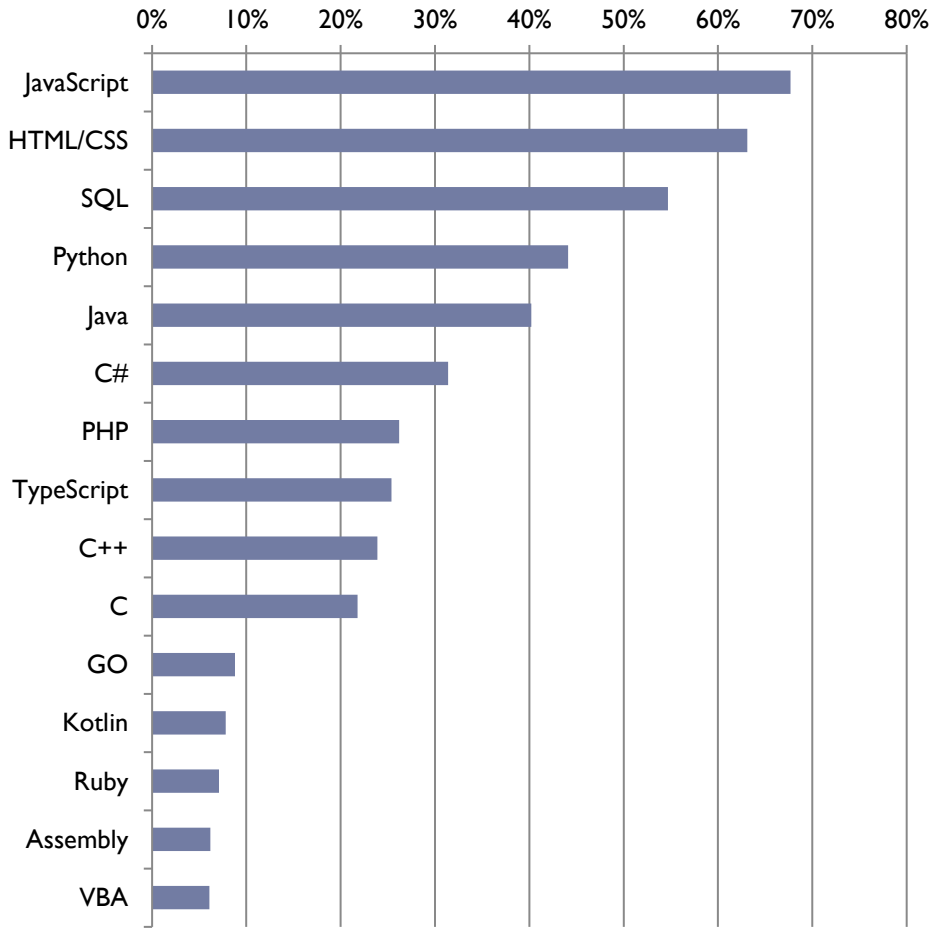
https://insights.stackoverflow.com/survey/2020

Popularność języków programowania



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

Popularność języków programowania



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

Języki programowania

C	Strukturalny język programowania wysokiego poziomu. Jego budowa i składnia jest podstawą dla większości współcześnie używanych języków programowania.
C++	Następca języka C rozszerzający go o obiektowe mechanizmy programowania.
C#	C# jest często określany jako następca (rozszerzenie) języka C++. Jest najczęściej stosowanym językiem programowania dla platformy .NET.
Java	Obiektowy, wieloplatformowy język programowania oparty w dużej mierze na języku C++. Java ma wiele zastosowań, między innymi jest natywnym językiem programowania dla systemu Android.
Python	Język programowania wysokiego poziomu o rozbudowanej bazie bibliotek standardowych. Python jest językiem interpretowanym, często stosowanym w postaci skryptów. Unikalną cechą Pythona jest używanie wcięć do definiowania bloków kodu zamiast nawiasów klamrowych stosowanych w większości innych języków.
GO	Kompilowany, wieloplatformowy język programowania składniowo podobny do języka C, lecz zmodyfikowany w celu łatwiejszego tworzenia aplikacji.

Języki programowania

PHP	skryptowy język programowania służący do dynamicznego generowania stron internetowych i budowania aplikacji webowych działający po stronie serwera (generujący i wysyłający do klienta gotowy dokument HTML).
JavaScript	Skryptowy język programowania stosowany głównie na stronach internetowych. Klasyczne skrypty JavaScript wykonywane są po stronie klienta (ich środowiskiem uruchomieniowym jest przeglądarka), jednak przy użyciu środowiska uruchomieniowego Node.js mogą być wykonywane po stronie serwera.
jQuery	Biblioteka programistyczna rozszerzająca język JavaScript, ułatwiająca między innymi manipulowanie
HTML	hipertekstowy język znaczników (ang. HyperText Markup Language) stosowany do tworzenia warstwy logicznej stron WWW
XML	uniwersalny język znaczników (ang. Extensible Markup Language) umożliwiający prezentowanie w strukturalizowany sposób różnego rodzaju danych. Przeznaczony jest między innymi do przenoszenia danych pomiędzy różnymi systemami.
SQL	(ang. Structured Query Language) – język zapytań służący do tworzenia, modyfikowania oraz komunikowania się z relacyjnymi bazami danych.

Środowisko programistyczne

Kod źródłowy - program napisany w języku takim jak Pascal lub C++,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, usuwanie błędów

Środowisko programistyczne

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.)

Środowisko programistyczne

Środowiska programistyczne dla języków C / C++

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

Visual Studio Code – nowy, 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

Pierwszy program (w języku C)

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

using namespace std;

int main()
{
    printf("To nasz pierwszy program w C\n");
    getch();
    return 0;
}
```

Pierwszy program (w języku C++)

```
1  #include <iostream>
2  #include <conio.h>
3
4  using namespace std;
5
6  int main()
7  {
8      cout << "Hello world!" << endl;
9      getch();  //zatrzymuje działanie programu
10     return 0;
11 }
```

Budowa programu

Budowa programu

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( )

Komentarze

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)

Zmienne

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

Zmienne

Modyfikatory typu:

signed	→	ze znakiem (±),	**int**	**char**	–
unsigned	→	bez znaku,	**int**	**char**	–
short	→	krótka (mniejsza),	**int**	–	–
long	→	długa (większa)	**int**	–	**double**

np. **unsigned long int** *dluga_liczba_bez_znaku* ;

Wartości domyślne:

long	=	**long int**
int	=	**signed int**
char	=	**signed char**

Zmienne

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;

Zmienne

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).

Zmienne

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

nazwaTypu nazwaZmiennej = wartość;

Np.: int liczba = 10;

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. **Strumienie konsoli – wczytanie z klawiatury i wypisanie na ekran**
2. Strumienie plikowe
3. Strumienie 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 (obiektowo 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

Prawda - Fałsz

W 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

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

Przykład:

```
cin >> i;
if (i!=0) cout << „i rozne od zera”;
else cout << „i rowne zero”;
```

Można i tak:

```
cin >> i;
if (i) cout << „i rozne od zera”;
else cout << „i rowne zero”;
```

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

Równanie kwadratowe – wersja 2

```
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;
}
```

Literatura:

W prezentacji wykorzystano przykłady i fragmenty:

- Gręboszj. : ***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.
- Kerninghan B. W., Ritchie D. M.: ***język ANSI C***, Wydawnictwo Naukowo Techniczne.

Dla bardziej zaawansowanych:

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