Metodyka programowania
Przykładowe pytania

Platforma .NET

1. Co to jest .NET?
2. Z jakich głównych elementów (dwóch!) składa się platforma .NET?
3. Wymień najważniejsze biblioteki .NET i ich przeznaczenie.
4. Co to jest IL? Jaka jest relacja między IL a językami takimi jak C# J# oraz językiem maszynowym?
5. Do jakiego stopnia aplikacje napisane w C# są przenośne (międzyplatformowe)? Od czego zależy możliwość ich uruchomienia np. na platformie Linux, Android?

Podstawy C#

6. Wymień typy proste języka C#. Podaj jakiego rodzaju wartości można w nich zapisywać.
   Boolean – wartości logiczne true i false; Int32, Int16, Int64 – liczby całkowite ze znakiem (różny zakres wartości, np. Int16: -32000 … +32000); UInt32, UInt16, UInt64 – liczby całkowite bez znaku; Double, Single – liczby rzeczywiste w notacji zmiennoprzecinkowej, String – łańcuchy znaków.
7. Do czego służy metoda Parse typów prostych? Podaj przykład użycia.
   Metoda Parse konwertuje łańcuch znaków na wartość innego typu, np. łańcuch "123" na wartość całkowitą 123. Dzięki metodzie Parse można przekształcić dane wprowadzane przez użytkownika (z klawiatury) na liczby.
   ```
   String s;
   Int32 n;
   s = Console.ReadLine();
   n = Int32.Parse(s);
   ```
8. Dlaczego przedstawiony poniżej fragment kodu wywołuje błąd podczas kompilacji programu (komunikat błędu: Cannot implicitly convert type 'int' to 'short')?
   ```
   Int32 x = 7;
   Int16 y;
   y = x;
   ```
   Konwersja automatyczna jest uznawana za błędną, kiedy zachodzi ryzyko utraty wartości lub precyzji zapisu liczby. Zmienna typu Int32 może zawierać wartość zbyt dużą, aby dało się ją zapisać w zmiennej typu Int16.
9. Wiedząc, że Form2 jest nazwą klasy (a zatem typu referencyjnego), wyjaśnij na czym polega błąd we fragmencie kodu:
   ```
   Form2 f2;
   f2.ShowModal();
   ```
   Deklaracja zmiennej typu referencyjnego nie powoduje utworzenia

obiektu. Po zadeklarowaniu referencji trzeba utworzyć obiekt operatorem new i dopiero wtedy można go użyć. Zatem powinno to wyglądać np. tak:

```
Form2 f2 = new Form2();
f2.ShowModal();
```

10. Wiedząc, że Form2 jest nazwą klasy (a zatem typu referencyjnego), wyjaśnij ile obiektów typu Form2 zostanie utworzonych przez program:

```
Form2 f = new Form2();
Form2 g = f;
Form2 h = g;
```

Zostanie utworzony tylko jeden obiekt. Za tworzenie obiektu jest odpowiedzialny operator new, natomiast operator przypisania kopiuje tylko referencję obiektu, a nie sam obiekt. W kodzie powyżej zmienne f, g oraz h będą referencjami do tego samego obiektu.

11. Wymień operatory arytmetyczne dwuargumentowe (podaj ich symbole i znaczenie).

* mnożenie
/ dzielenie
% reszta z dzielenia
+ dodawanie
- odejmowanie

12. Wymień operatory relacji i równości (podaj ich symbole i znaczenie). Podaj przykład użycia operatora równości w instrukcji warunkowej.

> większe
>= większe lub równe
< mniejsze
<= mniejsze lub równe
== równe
!= nierówne
Przykład:

```
if (x == 0)
  Console.WriteLine("x jest równe zero");
```

13. Wymień rozszerzone operatory przypisania (podaj ich symbole). Podaj przykład użycia i opisz działanie jednego z nich.

*=
/=
+=
-=
Przykład:

```
Int32 x = 7;
x *= 2;
Console.WriteLine("x = " + x.ToString()); // wydrukuje x = 14
```

Wyrażenie x *= 2 jest wykonywane tak jak wyrażenie x = x *2, tj. wartość zmiennej jest wymnażana przez wartość podaną po prawej stronie operatora, a wynik jest z powrotem zapisywany w zmiennej.

14. Podaj nazwę operatora != (wykrzyknik równe). Opisz typ i sposób wyznaczania wyniku. Podaj przykład użycia.

Jest to jeden z operatorów równości, oznacza „nie równe”. Zwraca wartość typu Boolean, tj. true (kiedy wartości po lewej i prawej stronie operatora są różne) albo false (w przeciwnym przypadku).
Przykład:

```
Int32 a=7, b=33;
if (a != b)
  Console.WriteLine("a i b są rózne”);
else
  Console.WriteLine("a i b są równe”);
```

(w podobnym kontekście mogą się pojawić pytania o inne operatory, np. *=, %, >, <=, ==, &&, ||, !, itd.)

15. Jaki jest wynik dzielenia (odpowiedź proszę uzasadnić):

```
Double x;
x = 3 / 4;
```

16. Co to jest hierarchia operatorów ?

Określona dla danego języka programowania kolejność operatorów, uwzględniana przy obliczaniu wyrażeń. Np. w wyrażeniu

```
y = a + b * c;
```

najpierw wykonywane jest mnożenie, później dodawanie i dopiero na końcu przypisanie (ponieważ operator mnożenia stoi w hierarchii wyżej, niż operator dodawania, a ten wyżej, niż operator przypisania). Aby zmienić kolejność wykonywania operatorów, należy stosować nawiasy, np.:

```
y = (a + b) * c
```

17. Uszereguj operatory arytmetyczne + - * / % zgodnie z ich hierarchią

Operator dodawania (+) ma taką samą hierarchię, jak operator odejmowania (-). Podobnie operator mnożenia (*) ma taką samą hierarchię, jak operator dzielenia (/) i operator reszty z dzielenia (%). Operatory + i - stoją w hierarchii o jeden szczebel niżej, niż operatory * i /.

18. Co to jest wiązanie operatorów ?

Określona dla danego języka programowania kolejność wykonywania operatorów o tej samej hierarchii. Większość operatorów ma wiązanie lewe (tzn. wykonywane są od lewej do prawej). Nieliczne, w tym operatory przypisania, maja wiązanie prawe. Np. w wyrażeniu

```
y = a / b * c;
```

najpierw jest wykonywane dzielenie, a dopiero potem mnożenie – czyli y jest obliczane jako (a/b)*c;
Aby zmienić kolejność wykonywania operatorów, należy stosować nawiasy, np.:

```
y = a / (b * c);
```

19. Opisz sposób obliczania wyrażenia i podaj wartości zmiennych po jego zakończeniu:
```
Int32 y = 4, a = 2, b = 3;
y *= a = b += 2;
```
20. Zamień wyrażenie w języku C++ na wzór matematyczny:
```
y = a+b / c-d; albo
y = a/b/c - a/b*c; itp.
```
(należy uwzględnić hierarchię i wiązanie operatorów)
21. Zamień wzór matematyczny na wyrażenie w języku C++

$$y = \frac{3a}{\sqrt{a} + b}$$

(należy uwzględnić hierarchię i wiązanie operatorów)
22. Przedstaw składnię instrukcji pustej. W jakich sytuacjach instrukcja pusta może się przydać?

; (średnik)

Stosuje się tam, gdzie składnia języka wymaga podania instrukcji, a nic konkretnego nie ma być zrobione. Instrukcja pusta przydaje się bardzo rzadko, a jej użycie w instrukcjach warunkowych i iteracyjnych najczęściej jest błędem, np.:
```
if (delta<0);
  Console.WriteLine("Brak pierwiastków");
```
Średnik po warunku tworzy instrukcję pustą (która jest lub nie jest wykonywana, zależnie od wyniku sprawdzenia warunku), natomiast wydruk jest wykonywany zawsze, ponieważ w tej sytuacji nie jest częścią instrukcji warunkowej.
23. Przedstaw składnię instrukcji grupującej (bloku instrukcji). Kiedy się ją stosuje? Podaj przykład.
```
{
 //dowolna ilość instrukcji
}
```
Stosuje się wtedy, gdy składnia pozwala na użycie tylko jednej instrukcji, a trzeba użyć większej ich liczby. Przykład:
```
for (i=1; i<=10; i++)
{
  k = i*i;
  Console.WriteLine(String.Format("{0} ^2 = {1}", i, k));
}
```
Intencja programisty jest, aby każda iteracja składała się z dwóch instrukcji (obliczenia i wydruk). Składnia for pozwala na użycie tylko jednej instrukcji, zatem konieczne jest użycie instrukcji grupującej. Gdyby blok został pominięty:
```
for (i=1; i<=10; i++)
  k = i*i;
Console.WriteLine(String.Format("{0} ^2 = {1}", i, k));
```
to pętla wykonała by wielokrotnie tylko obliczenia, zaś wydruk, w tej sytuacji nie będący częścią instrukcji for, został by wykonany jeden raz, po zakończeniu pętli.

24. Przedstaw składnię instrukcji warunkowej if.
Opisz kolejność wykonywania operacji. Podaj przykład użycia.

```
if (warunek)
  instrukcja
```

Program oblicza wartość wyrażenia podanego jako "warunek". Jeżeli warunek jest spełniony (wartość logiczna true), program wykonuje instrukcję, w przeciwnym wypadku instrukcja jest pomijana

```
if (a != b)
  Console.WriteLine("a i b są rózne”);
```

25. Przedstaw składnię instrukcji warunkowej if-else.
Opisz kolejność wykonywania operacji. Podaj przykład użycia.

```
if (warunek)
  instrukcja_1
else
  instrukcja_2
```

Program oblicza wartość wyrażenia podanego jako "warunek". Jeżeli warunek jest spełniony (wartość logiczna true), wykonywana jest tylko instrukcja_1, w przeciwnym wypadku wykonywana jest tylko instrukcja_2

```
if (x != 0)
  y = sin(x)/x;
else
  y = 1;
```

26. Przedstaw składnię instrukcji iteracyjnej for.
Opisz kolejność wykonywania operacji.

```
for (inicjacja; warunek; inkrementacja)
  instrukcja
```

Program wykonuje wyrażenie "inicjacja", następnie oblicza wartość wyrażenia podanego jako "warunek". Jeżeli warunek jest spełniony (wartość logiczna true), to wykonywana jest instrukcja i wyrażenie "inkrementacja", a następnie opisane czynności są powtarzane, począwszy od sprawdzenia warunku. Jeżeli warunek nie jest spełniony, to następuje wyjście z pętli.

27. Przedstaw składnię instrukcji iteracyjnej while.
Opisz kolejność wykonywania operacji.

```
while (warunek)
  instrukcja
```

Program oblicza wartość wyrażenia podanego jako "warunek". Jeżeli warunek jest spełniony (wartość logiczna true), wykonywana jest instrukcja, a następnie opisane czynności są powtarzane. Jeżeli warunek nie jest spełniony, to instrukcja nie jest wykonywana i następuje wyjście z pętli

28. Przedstaw składnię instrukcji iteracyjnej for. Podaj dowolny przykład użycia.

Składnia:

```
for (inicjacja; warunek; inkrementacja)
  instrukcja
```

Przykłady:
Wydrukowanie liczb od 1 do 10

```
int i;
for (i=1; i<=10; i++)
  Console.WriteLine(i.ToString());
```

Obliczenie sumy liczb od 1 do 100

```
int i, s=0;
for (i=0; i<100; i++)
  s += i;
Console.WriteLine(s.ToString());
```

29. W jakich sytuacjach stosuje się raczej pętlę for, niż while.
Podaj przykłady takich sytuacji.
Użycie pętli for, czyli tzw. pętli z licznikiem, jest wygodniejsze, kiedy liczba iteracji jest z góry znana. Przykłady: obliczanie silni, drukowanie zawartości tablicy.
30. W jakich sytuacjach stosuje się raczej pętlę while, niż for.
Podaj przykłady takich sytuacji.
Użycie pętli while, czyli tzw. pętli z warunkiem, jest wygodniejsze, kiedy liczba iteracji nie jest z góry dokładnie znana. Przykłady: wyznaczanie NWP metodą Newtona, czytanie pliku tekstowego (nieznana ilość linii).
31. Podaj składnię instrukcji try-catch i opisz jej działanie.

```
try
{
  // instrukcje potencjalnie prowadzące do błędu
}
catch
{
  // obsługa błędu
}
```

Jeżeli podczas wykonywania instrukcji zawartych w bloku try wystąpi błąd krytyczny (tzw. wyjątek programowy), to wykonanie tych instrukcji zostanie przerwane, po czym wykonane zostaną instrukcje z bloku catch. Jeżeli błąd nie wystąpi, instrukcje bloku catch są pomijane.
Błąd krytyczny powstały poza blokiem try powoduje natychmiastowe zamknięcie aplikacji.
32. Podaj składnię i przykład użycia instrukcji try-catch.

```
try
{
  // instrukcje potencjalnie prowadzące do błędu
}
catch
{
  // obsługa błędu
}
```

Przykład:

```
Int32 n, k;
String s;
try
{
  s = Console.ReadLine();
  n = Int32.Parse(s);
  k = n * n;
  Console.WriteLine(k.ToString());
}
catch
{
  Console.WriteLine("Błędne dane!");
}
```

Jeżeli podczas konwersji łańcucha s na wartość całkowitą wystąpi błąd, to instrukcja n = Int32.Parse(s) zostanie przerwana (n nie zmieni wartości), a następujące po niej dwie instrukcje w ogóle nie zostaną wykonane – zamiast tego program wykona instrukcję z bloku catch.

33. Jakie elementy udostępnia klasa Math? Czy trzeba tworzyć obiekt klasy Math, aby z tych elementów skorzystać? Podaj przykład użycia.
    Udostępnia funkcje matematyczne (m.in. pierwiastek, sinus, cosinus, logarytm, potęga, exponenta itd.) oraz stałe. Nie trzeba (a ściślej nie można) tworzyć obiektu klasy Math, ponieważ wszystkie elementy składowe są statyczne. Przykład:

```
y = Math.Sin(fi * Math.PI / 180.0);
```

34. Jakie elementy udostępnia klasa Console? Czy trzeba tworzyć obiekt klasy Console, aby z tych elementów skorzystać? Podaj przykład.
    Klasa Console reprezentuje standardowe wejście i wyjście aplikacji konsolowych (tj. klawiaturę i ekran). Udostępnia m.in. funkcje do wczytywania i wyświetlania danych oraz właściwości okna konsoli (kolor tekstu i tła, tytuł i położenie okna konsoli na ekranie itp.).
    Przykład:

```
String s;
Int32 n;
Console.WriteLine("Podaj liczbę");
s = Console.ReadLine();
n = Int32.Parse(s);
```

Zdarzenia

35. Co to znaczy, że aplikacja jest „sterowana zdarzeniami” (ang. Event Driven)?
36. Wymień i krótko scharakteryzuj zdarzenia (3-4) bezpośrednio związane z myszką.
37. Wymień i krótko scharakteryzuj zdarzenia (3) bezpośrednio związane z klawiaturą.

38. Wymień i krótko scharakteryzuj zdarzenia (3-4) związane z cyklem życia formularzy i aplikacji.
39. Dlaczego, w większości przypadków, program obsługuje zdarzenie OnClick, a nie OnMouseUp?

Komponenty

40. Wymień i krótko scharakteryzuj (tj.: wygląd, typowe zastosowania, charakterystyczne cechy itp.) komponent klasy przycisk (Button).

    Inne komponenty w takim kontekście:
    pole wyboru (CheckBox), przycisk radiowy (RadioButton),
    etykieta (Label), pole tekstowe (TextBox), czasomierz (Timer)

41. Scharakteryzuj właściwości: Name, Text, TextAlign,
    (jak właściwość wpływa na wygląd lub działanie komponentu)

    Inne właściwości w takim kontekście:
    Visible, Enabled, ReadOnly, Font, ForeColor, BackColor, AutoSize, Anchor

42. W jaki sposób użytkownik może „kliknąć” przycisk?
    (są 4 sposoby)
43. Jaki jest skutek umieszczenia we właściwości Text, np. komponentu RadioButton, znaku „&” (ampersand)?
44. Etykieta ma właściwość Text „Wsp. &a”. Jaki skutek da wciśnięcie kombinacji klawiszy <Alt>+A
45. Scharakteryzuj właściwości komponentu Label: AutoSize, Text, TextAlign

    Inne komponenty i ich właściwości w takim kontekście:
    TextBox => ReadOnly, Multiline, WordWrap, ScrollBars, PasswordChar
    Button => Text, Enabled,
    CheckBox => Checked, Text, CheckAlign
    RadioButton => Checked, Text, CheckAlign
    Timer => Interval, Enabled

46. Aby uniemożliwić edycję tekstu w komponencie TextBox, można użyć właściwości Enabled albo ReadOnly. Jaka jest różnica między nimi?
47. Komponenty CheckBox i RadioButton mogą posłużyć do wybierania opcji przez użytkownika. Jaka jest różnica między nimi?

Formularze

48. Scharakteryzuj właściwości formularza (Form): FormBorderStyle, StartPosition.

    Inne właściwości w takim kontekście:
    AcceptButton, CancelButton, DialogResult

49. W jakim stopniu i w jaki sposób programista może zmieniać wygląd ramki i paska tytułu okna?
    (co można i jak się to robi, a czego nie można zrobić)
50. Drugie okno programu można wyświetlić używając metody Show() albo ShowDialog(). Jaka jest różnica między nimi?
51. Co to jest „okno modalne” systemu Windows?
    (przykład, wygląd, zachowanie, relacje z innymi oknami programu)
52. Do czego służy właściwość DialogResult formularzy?
53. Do czego służy właściwość DialogResult komponentów typu przycisk (Button)?
54. Komponent typu przycisk (Button) ma właściwość Text „&Anuluj” i jest ustawiony jako CancelButton formularza, który został wyświetlony metodą ShowDialog().
    W jaki sposób użytkownik może zamknąć ten formularz, aby było to równoważne kliknięciu przycisku „Anuluj”?
    (są 4 sposoby)