

## Microsoft Visual C++ .NET

EDYCJA POLSKA: WYDAWNICTWO RM

### Str. 7

Jest: ...lub biblioteka Microsoft Foundation Class (MFC).  
Powinno być: ...lub biblioteka Microsoft Foundation Classes (MFC).  
Komentarz: MFC to *Microsoft Foundation Classes*.

### Str. 21

Jest:  

```
class nazwaklasy
{
Słowa kluczowe kontrolujące dostęp (Public: Private: lub Protected:)
Deklaracja zmiennych i metod klasy
}
```

### Powinno być:

```
{
Słowa kluczowe kontrolujące dostęp (public: private: lub protected:)
Deklaracja zmiennych i metod klasy
};
```

Komentarz: Słowa kluczowe - *public*, *private* i *protected* - muszą być pisane w całości małymi literami. Po nawiasie klamrowym, który kończy definicję klasy, powinien być średnik.

### Str. 89

Jest: 

```
CreditCardAccount * myAccount // Deklaracja wskaźnika
```

  
Powinno być: 

```
CreditCardAccount * myAccount; // Deklaracja wskaźnika
```

  
Komentarz: Brak średnika kończącego instrukcję.

### Str. 131

Jest:  

```
void Credit(double amount); // Tej funkcji nie można przeciążyć
void Debit(double amount); // ani tej
```

### Powinno być:

```
void Credit(double amount); // Tej funkcji nie można zastąpić
void Debit(double amount); // ani tej
```

Komentarz: Na stronach 131-133 w komentarzach występuje błąd polegający na występowaniu słowa *przeciążenie*. Nie chodzi tu o przeciążanie, ale o *zastępowanie*, czyli *nadpisywanie* (ang. *override*).

### Str. 131

Jest: 

```
virtual String * ToString(); // Tę funkcję można przeciążyć
```

  
Powinno być: 

```
virtual String * ToString(); // Tę funkcję można zastąpić
```

### Str. 131

Jest:

```
virtual bool CanDebit(double amount) = 0; // Tę funkcję
// trzeba przeciążyć
```

Powinno być:

```
virtual bool CanDebit(double amount) = 0; // Tę funkcję
// trzeba zastąpić
```

### Str. 132

Jest:

```
virtual String * ToString(); // Klasy potomne mogą przeciążyć
virtual bool CanDebit(double amount) = 0; // Klasy potomne muszą przeciążyć
```

Powinno być:

```
virtual String * ToString(); // Klasy potomne mogą zastąpić
virtual bool CanDebit(double amount) = 0; // Klasy potomne muszą zastąpić
```

### Str. 132

Jest:

```
virtual String * ToString(); // Można przeciążyć ToString
virtual bool CanDebit(double amount) = 0; // przeciążenie CanDebit jest
// konieczne
```

Powinno być:

```
virtual String * ToString(); // Można zastąpić ToString
virtual bool CanDebit(double amount) = 0; // zastąpienie CanDebit jest
// konieczne
```

### Str. 142 (w kolumnie *Odpowiednik typu w Managed C++*)

Jest: `wchar_+`

Powinno być: `wchar_t`

Komentarz: Zamiast znaku + powinna być litera t.

### Str. 151

Jest:

```
__value enum WeekDay
{
    Monday = 1, Tuesday, Wednesday, Thursday, Friday,
    Saturday, Sunday
}
```

Powinno być:

```
__value enum WeekDay
{
    Monday = 1, Tuesday, Wednesday, Thursday, Friday,
    Saturday, Sunday
}
```

Komentarz: Zbędny średnik przed słowem kluczowym `__value`.

### Str. 173

Jest: Wyjątki Managed C++ mogą ponadto wykorzystywać typy zarządzane (np. klasy `_gc` oraz typy `__value`).

Powinno być: Wyjątki Managed C++ mogą ponadto wykorzystywać typy zarządzane (np. klasy `__gc` oraz typy `__value`).

Komentarz: Brak jednego znaku podkreślenia przed słowem kluczowym `gc`.

### Str. 177

Jest:

```
try
{
    int n = 3;
    Console::WriteLines("Calling with n=3");
    func(n);
    Console::WriteLines("Calling with n=0");
    n = 0;
    func(n);
}
```

Powinno być:

```
try
{
    int n = 3;
    Console::WriteLine(S"Calling with n=3");
    func(n);
    Console::WriteLine(S"Calling with n=0");
    n = 0;
    func(n);
}
```

Komentarz: W nazwie metody wyświetlającej ciąg znaków nie powinna znajdować się litera *s*. Natomiast wielka litera *S* powinna znajdować się przed łańcuchem znaków przeznaczonym do wyświetlenia.

### Str. 196

Jest:

```
void func(int arr[], size_t size)
{
    for(size_t i=0; i<size; i++)
        console::WriteLine(arr[i]);
}
```

Powinno być:

```
void func(int arr[], size_t size)
{
    for(size_t i=0; i<size; i++)
        Console::WriteLine(arr[i]);
}
```

Komentarz: Przestrzeń nazw *Console* koniecznie musi zaczynać się wielką literą.

### Str. 198

Jest:

```
for(int j=0; j<3; j++)
    Console::Write("{0}", box(arr[i][j]));
```

Powinno być:

```
for(int j=0; j<3; j++)
    Console::Write("{0}", __box(arr[i][j]));
```

Komentarz: Brak jednego znaku podkreślenia przed słowem kluczowym *box*.

### Str. 204

Jest:

```
console.WriteLine("pn[{0},{1}] = {2}", __box(j),  
__box(k), __box(pn[j,k]));
```

Powinno być:

```
Console.WriteLine("pn[{0},{1}] = {2}", __box(j),  
__box(k), __box(pn[j,k]));
```

Komentarz: Przestrzeń nazw *Console* koniecznie musi zaczynać się wielką literą.

### Str. 210

Jest:

```
for (int i=0; i<pal->Count; i++)  
    Console.WriteLine("Item({0}) = {1}", __box(i), pal->get_Item(i));
```

Powinno być:

```
for (int i=0; i<pal->Count; i++)  
    Console.WriteLine("Item({0}) = {1}", __box(i), pal->get_Item(i));
```

Komentarz: Brak cudzysłowu zamykającego łańcuch znaków.

### Str. 214

Jest:

```
while (ie->MoveNext())  
    console.WriteLine(ie->Current);
```

Powinno być:

```
while (ie->MoveNext())  
    Console.WriteLine(ie->Current);
```

Komentarz: Przestrzeń nazw *Console* koniecznie musi zaczynać się wielką literą.

### Str. 217

Jest: ... (a nie jakieś inne, które rozpoczynają się od słów *\_\_get* oraz *\_\_set*).

Powinno być: ... (a nie jakieś inne, które rozpoczynają się od słów *get\_* oraz *set\_*).

Komentarz: Znak podkreślenia powinien być po słowach *set* i *get*.

### Str. 217

Jest: Console.WriteLine("Age of {0} is {1}", pp->Name, \_\_box(pp->Age));

Powinno być: Console.WriteLine("Age of {0} is {1}", pp->Name, \_\_box(pp->Age));

Komentarz: Zbędna spacja pomiędzy dwoma znakami podkreślenia a słowem *box*.

### Str. 232

Jest: console.WriteLine("Result of cube() is {0}", \_\_box(result2));

Powinno być: Console.WriteLine("Result of cube() is {0}", \_\_box(result2));

Komentarz: Przestrzeń nazw *Console* koniecznie musi zaczynać się wielką literą.

### Str. 250

Jest: using namespace system::Collections;

Powinno być: using namespace System::Collections;

Komentarz: Przestrzeń nazw *System* musi zaczynać się wielką literą.

**Str. 256**

Jest: `System::Drawing::Desing` rozszerza przestrzeń `System::Drawing` i pozwala...

Powinno być: `System::Drawing::Design` rozszerza przestrzeń `System::Drawing` i pozwala...

Komentarz: Przetawienie liter *n* i *g*.

**Str. 256**

Jest: `System::Drawing:Drawing2D` zawiera...

Powinno być: `System::Drawing::Drawing2D` zawiera...

Komentarz: Brak dwukropka w odwołaniu do przestrzeni nazw `System::Drawing::Drawing2D`.

**Str. 257**

Jest: Obsługę sieci zapewniają klasy `System::Net` oraz `System::Net:Sockets`.

Powinno być: Obsługę sieci zapewniają klasy `System::Net` oraz `System::Net::Sockets`.

Komentarz: Brak dwukropka w odwołaniu do przestrzeni nazw `System::Net::Sockets`.

**Str. 315**

Jest: `treeView1->Size = System:Drawing::Size(121, 273);`

Powinno być: `treeView1->Size = System::Drawing::Size(121, 273);`

Komentarz: Brak dwukropka w odwołaniu do przestrzeni nazw `System::Drawing`.

**Str. 318**

Jest:

```
catch(System::Exception* pe)
{
    MessageBox::Show(pe->Message, "Error");
    Return;
}
```

Powinno być:

```
catch(System::Exception* pe)
{
    MessageBox::Show(pe->Message, "Error");
    return;
}
```

Komentarz: Słowo kluczowe `return` musi być pisane z małej litery.

**Str. 321**

Jest: Kolekcja elementów formantu `ListView`

Powinno być: Kolekcja elementów formantu `ListView`

Komentarz: Zbędna spacja w nazwie formantu `List View`.

**Str. 341**

Jest: `System::Drawing::Drawing2` zapewnia...

Powinno być: `System::Drawing::Drawing2D` zapewnia...

Komentarz: Poprawna nazwa tej przestrzeni nazw to `System::Drawing::Drawing2D`.

**Str. 379**

Jest: Dokończenie tabeli na następnej stronie

Powinno być: Dokończenie tabeli na następnej stronie

Komentarz: Zbędna litera *n* w słowie `dokończenie`.

**Str. 499**

Jest: ...dodatkowe dane muszą być zapisywane w Rejestrze Windows.

Powinno być: ...dodatkowe dane muszą być zapisywane w rejestrze Windows.