44 Stosowanie instrukcji, funkcji, procedur, obiektów i metod wybranych języków programowania

EFEKTY KSZTAŁCENIA Z PODSTAWY PROGRAMOWEJ:

- PKZ(E.b)(3) dobiera oprogramowanie użytkowe do realizacji określonych zadań;
- E.14.3(4) stosuje instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania.

W TYM ROZDZIALE:

- dowiesz się, jakie języki programowania mogą być wykorzystane do tworzenia aplikacji;
- poznasz programowanie strukturalne i obiektowe;
- dowiesz się, na czym polega programowanie sterowane zdarzeniami;
- nauczysz się stosować instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania.

Wprowadzenie

Języki programowania opisują, jak komputer ma rozumieć instrukcje i wyrażenia, za pomocą których został zapisany algorytm. Algorytm zapisuje się w postaci **kodu źródłowego**, który następnie jest przetwarzany na instrukcje wykonywane przez procesor.

Sposoby przetwarzania kodu źródłowego

- **Kompilacja kodu źródłowego** – kod źródłowy jest tłumaczony do postaci **języka maszynowego**, czyli elementarnych operacji gotowych do przetworzenia przez procesor komputera. Języki tego typu są określane jako kompilowane języki programowania. Kompilacja jest wykonywana jeden raz, w jej wyniku otrzymuje się plik wykonywalny (np. z rozszerzeniem .exe), który może być wykonany wielokrotnie. Języki kompilowane zapewniają najwyższą wydajność programom, lecz kod maszynowy jest ściśle powiązany z platformą sprzętową. Przykładami popularnych języków kompilowanych są C++, C# oraz często używany w dydaktyce język Pascal.
- **Interpretacja kodu źródłowego** – kod źródłowy jest na bieżąco tłumaczony i wykonywany przez dodatkowy program, zwany interpreterem. Języki tego typu są określane jako interpretowane języki programowania. Interpretacja jest wykonywana podczas każdego uruchomienia programu. Przykładami popularnych języków interpretowanych, stosowanych do tworzenia aplikacji internetowych, są JavaScript oraz PHP.
- **Kompilacja do kodu bajtowego** – kod źródłowy jest kompilowany do **kodu pośredniego** (kodu bajtowego), który jest następnie wykonywany przez maszynę wirtualną. Kod pośredni nie zależy od systemu operacyjnego i procesora, co umożliwia uruchomienie programu na różnych platformach sprzętowych i programowych. Przykładem takiego języka jest Java.

Podstawowe sposoby programowania, nazywane paradygmatami

- **Programowanie strukturalne** – zaleca podział kodu źródłowego programu na procedury i hierarchicznie ułożone bloki z wykorzystaniem struktur algorytmicznych: sekwencji, selekcji, iteracji (pętli) i rekursji. W programowaniu strukturalnym unika się stosowania instrukcji skoku. Komunikacja między procedurami jest realizowana przez dobrze określone interfejsy;
- **Programowanie obiektowe** – programy definiuje się za pomocą obiektów – elementów łączących **stan** (czyli dane, nazywane polami) i **zachowanie** (czyli procedury, nazywane metodami). Obiektowy program komputerowy jest zbiorem takich obiektów komunikujących się pomiędzy sobą w celu wykonywania określonych zadań.

Programy mogą pracować w środowisku tekstowym (programy konsolowe) lub środowisku graficznym wykorzystującym **graficzny interfejs użytkownika** (GUI). **Programowanie sterowane zdarzeniami** jest powiązane z graficznymi środowiskami systemów operacyjnych oraz z programowaniem obiektowym. Określa sposób pisania programów z punktu widzenia procesu przekazywania sterowania między poszczególnymi modułami tej samej aplikacji. Według tego paradygmatu przepływ sterowania w programie jest niemożliwy do przewidzenia i przez cały czas działania programu generowane są **zdarzenia** (*events*), na które musi on odpowiadać. Przykładami zdarzeń są: naciśnięcie myszy, klawiszy, takt zegara systemowego, a także zdarzenia sieciowe i inne. Po zajściu zdarzenia uruchamia się odpowiednia **procedura obsługi** tego zdarzenia.

Jednym z popularnych kompilowanych języków programowania, umożliwiających programowanie strukturalne i obiektowe, jest C++. Istnieje wiele programów typu **zintegrowane środowisko programistyczne** (IDE),

zawierających aplikacje (środowisko) służące do tworzenia, modyfikowania, testowania i konserwacji oprogramowania w tym języku, np. DevC++, Visual Studio. Kod źródłowy programu może być tworzony za pomocą edytora tekstu wbudowanego w środowisko, a następnie skompilowany za pomocą wbudowanego kompilatora. Przykładowy kod źródłowy programu w C++ jest pokazany na listingu 3.44.1.

Listing 3.44.1

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
 int skladnik1, skladnik2, suma;
 skladnik1=1;
 skladnik2=2;
 suma = skladnik1 + skladnik2;
 cout << "Suma : „ << suma << endl;
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

Program wykonuje dodawanie dwóch liczb i wyświetla wynik. W celu wielokrotnego wykorzystania kodu można wydzielić procedury wykonujące określone zadania, np. w listingu 3.44.2 pokazano procedurę int dodaj (int sk1, int sk2) otrzymującą dwa argumenty, wykonującą dodawanie liczb i zwracającą wynik tego dodawania. Procedura taka może być wywołana w programie wielokrotnie. W programie można zdefiniować wiele procedur (funkcji) o takiej samej nazwie, ale tylko pod warunkiem, że mają one różne zestawy argumentów, np. różną liczbę lub różne typy argumentów. Zjawisko to nosi nazwę **przeciążania funkcji**.

Listing 3.44.2

```
#include <cstdlib>
#include <iostream>
using namespace std;
int dodaj(int sk1, int sk2)
{
 int wynik;
 wynik = sk1 + sk2;
 return wynik;
}
int main(int argc, char *argv[])
{
 int skladnik1, skladnik2, suma;
 skladnik1=1;
 skladnik2=2;
 cout << "Suma : " << dodaj(11, 22) << endl;
 cout << "Suma : " << dodaj(skladnik1, skladnik2) << endl;
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

Programista C++ (i innych języków obiektowych) może samodzielnie tworzyć skomplikowane typy danych, używane w programach. Definicja takiego typu danych jest nazywana **klasą**. W klasie oprócz **pól** zawierających dane można umieszczać **metody** (procedury). **Obiekt** jest egzemplarzem danej klasy. Przykład tworzenia prostej klasy, obiektu i ich wykorzystania został pokazany na listingu 3.44.3.

Listing 3.44.3

```
#include <cstdlib>
#include <iostream>
using namespace std;
class TPunkt{
```

```
public:
 int x;
 int y;
 void wypisz();
};
void TPunkt::wypisz()
{
cout << "x = " << x;
cout << "y = " << y;
cout << endl;
}
int main(int argc, char *argv[])
{
 TPunkt poczatek, koniec;
 poczatek.x=1;
 poczatek.y=1;
 poczatek.wypisz();
 system(„PAUSE");
 return EXIT_SUCCESS;
}
```

Konstruktor to metoda automatycznie wywoływana podczas tworzenia obiektu. Można za jego pomocą nadać wartości początkowe składowym obiektów, zaalokować pamięć itp. Konstruktor definiuje się jak inne metody, ale nie ma on typu zwracanego, a nazwa konstruktora musi być identyczna z nazwą klasy. **Destruktor** jest to metoda wywoływana automatycznie podczas niszczenia obiektów. Nie przyjmuje żadnych argumentów, a w klasie może być tylko jeden. Przykład użycia konstruktora pokazano na listingu 3.44.4.

Listing 3.44.4

```
#include <cstdlib>
#include <iostream>
#include <math.h>
using namespace std;
class TOdcinek{
public:
 double dlugosc_w_osi_x;
 double dlugosc_w_osi_y;
 double dlugosc();
 TOdcinek();
};
double TOdcinek::dlugosc()
{
double wynik;
wynik=sqrt( dlugosc_w_osi_x *dlugosc_w_osi_x + dlugosc_w_osi_y *dlugosc_w_osi_y);
return wynik;
}
TOdcinek::TOdcinek()
{
 dlugosc_w_osi_x=1;
 dlugosc_w_osi_y=1;
}
int main(int argc, char *argv[])
{
 TOdcinek odc1;
 cout << odc1.dlugosc() << endl;
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

Jeżeli dysponujemy pewną klasą i chcielibyśmy rozszerzyć jej możliwości, zachowując jednocześnie dotychczasowe funkcje, możemy stworzyć nową klasę, która dziedziczy po już istniejącej. Klasa, z której są dziedziczone pola lub metody, jest nazywana **klasą bazową** (rodzic), a klasa dziedzicząca – **klasą pochodną** (dziecko). Składowe klasy bazowej mogą być dziedziczone w trybie:

- **prywatnym** (private) – wszystkie składowe z klasy bazowej stają się prywatnymi w klasie pochodnej; ten tryb jest domyślny;
- **chronionym** (protected) – składowe publiczne z klasy bazowej stają się chronionymi w klasie pochodnej;
- **publicznym** (public) – nie zmienia w klasie pochodnej trybu dostępu do składowych klasy bazowej.

W listingu 3.44.5 została utworzona klasa TPunkt2D reprezentująca punkt na płaszczyźnie. Klasa TPunkt3D dziedziczy składowe x i y z klasy TPunkt2D i dodatkowo ma rozszerzenie w postaci zmiennej z reprezentującej oś z. Dla każdej z klas został utworzony obiekt, któremu przypisano wartości współrzędnych we wszystkich dostępnych osiach.

Listing 3.44.5

```
#include <cstdlib>
#include <iostream>
using namespace std;
class TPunkt2D{
public:
 int x;
 int y;
};
class TPunkt3D : public TPunkt2D{
public:
 int z;
};
int main(int argc, char *argv[])
{
 TPunkt2D punkt2;
 TPunkt3D punkt3;
 punkt2.x=1;
 punkt2.y=1;
 cout << punkt2.x << endl;
 cout << punkt2.y << endl;
 punkt3.x=11;
 punkt3.y=11;
 punkt3.z=11;
 cout << punkt3.x << endl;
 cout << punkt3.y << endl;
 cout << punkt3.z << endl;
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

LITERATURA

- J. Grębosz, *Symfonia C++ Standard. Programowanie w języku C++ orientowane obiektowo. Tom I i II*, Edition 2000, Kraków 2008.

NOTATKI

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

ZADANIE 1.

Wyszukaj w internecie informacje dotyczące wybranych pojęć stosowanych w programowaniu obiektowym. Wyjaśnij, co te pojęcia oznaczają. W edytorze tekstu wpisz odpowiednie informacje zgodnie z poniższą formatką. Zapisz dokument.

Pojęcie	Znaczenie
Dziedziczenie	
Hermetyzacja	
Przeciążenie funkcji	

ZADANIE 2.

Skorzystaj z wybranego języka programowania, napisz aplikację realizującą funkcje prostego kalkulatora wykonującego podstawowe operacje matematyczne (dodawanie, odejmowanie, dzielenie i mnożenie). Dla wykonania każdej operacji utwórz odpowiednią procedurę, która otrzyma dwa argumenty i zwróci wynik. Zademonstruj działania kalkulatora. W edytorze tekstu wpisz odpowiednie informacje zgodnie z poniższą formatką. Zapisz dokument.

Wybrane środowisko programistyczne	
Wybrany język programowania	
Kod źródłowy aplikacji	

ZADANIE 3.

Skorzystaj z wybranego obiektowego języka programowania, napisz aplikację służącą do przechowywania danych teleadresowych (np. nazwisko, imię, adres, telefon, e-mail). Dane poszczególnych osób muszą być przechowywane jako obiekty. Aplikacja powinna umożliwiać zapisywanie danych w pliku tekstowym oraz ich wyszukiwanie na podstawie nazwiska lub numeru telefonu. Zademonstruj działanie aplikacji. W edytorze tekstu wpisz odpowiednie informacje zgodnie z poniższą formatką. Zapisz dokument.

Wybrane środowisko programistyczne	
Wybrany język programowania	
Kod źródłowy aplikacji	

ZADANIE 4.

Wyszukaj w internecie informacje na temat współczynnika masy ciała (BMI) i sposobu jego obliczania. Skorzystaj z wybranego obiektowego języka programowania i napisz aplikację służącą do obliczania współczynnika masy ciała (BMI). Dane poszczególnych osób powinny być przechowywane jako obiekty klasy **TOsoba**. Klasa **TOsoba** powinna zawierać pola: **nazwisko**, **imie**, **wzrost**, **waga** i metodę **oblicz_BMI()**. Ponadto w klasie należy zdefiniować konstruktor, otrzymujący cztery argumenty, oraz destruktor. Zadaniem konstruktora jest zamiana wzrostu podawanego w centymetrach na metry. Zadaniem destruktora jest wyświetlenie komunikatu informującego o usunięciu obiektu. Zademonstruj działanie aplikacji. W edytorze tekstu wpisz odpowiednie informacje zgodnie z poniższą formatką. Zapisz dokument.

SPRAWDŹ SWOJE UMIEJĘTNOŚCI

Wybrane środowisko programistyczne	
Wybrany język programowania	
Kod źródłowy aplikacji	

ZADANIE 5.

Skorzystaj z wybranego obiektowego języka programowania i napisz aplikację służącą do obliczania obwodu oraz pola powierzchni figur geometrycznych. Klasa bazowa **TFigura** powinna zawierać składową **nazwa**. Klasy **TProstokat** i **TKolo** dziedziczą z klasy **TFigura**. Ponadto każda z nich zawiera metodę **pole()** i **obwód()** obliczające odpowiednio pole i obwód figury. Zademonstruj działanie aplikacji. W edytorze tekstu wpisz odpowiednie informacje zgodnie z poniższą formatką. Zapisz dokument.

Wybrane środowisko programistyczne	
Wybrany język programowania	
Kod źródłowy aplikacji	

Rozwiązania zadań zapisz w pliku pod nazwą **AI_44_nazwisko.doc**. Przedstaw do oceny nauczycielowi.

PODSUMOWANIE

TEST 44. Część pisemna egzaminu zawodowego

Zadanie 1.
Kod źródłowy w języku C++ jest przetwarzany podczas kompilacji do

A. kodu maszynowego. B. kodu binarnego. C. kodu bajtowego. D. kodu interpretowanego.

Zadanie 2.
Które zdanie jest prawdziwe?

A. W programie nie mogą istnieć dwie procedury o identycznej nazwie.
B. W programie mogą istnieć procedury o identycznej nazwie.
C. W programie mogą istnieć dwie procedury o identycznej nazwie, pod warunkiem że mają różne zestawy argumentów.
D. Procedura w programie może być wywołana jednokrotnie.

Zadanie 3.
Składowe klasy bazowej są typu publicznego. Klasa potomna dziedziczy składowe w trybie chronionym. Odziedziczone składowe w klasie potomnej będą typu

A. prywatnego.
B. chronionego.
C. publicznego.
D. nie można dziedziczyć składowych w trybie chronionym.

Zadanie 4.
Metoda automatycznie wywoływana podczas tworzenia obiektu to

A. składowa. B. destruktor. C. konstruktor. D. funkcja wirtualna.

Zadanie 5.
Po wystąpieniu zdarzenia

A. następuje przerwanie wykonywania programu.
B. następuje przerwanie wykonywania aktualnej procedury.
C. jest uruchamiane przerwanie systemowe.
D. jest wykonywana procedura obsługi zdarzenia.