ISNN 1899-8534

Kwartalnik Wojewódzkiego Ośrodka Doskonalenia Informatycznego i Politechnicznego

eWODIiP

Edukacja w drodze do społeczeństwa informacyjnego

Nr 3 (grudzień)/2008

SPIS TREŚCI

http://kwartalnik.wodip.opole.pl

kwartalnik@wodip.opole.pl

Komputer dla ucznia

mgr Lesław Tomczak

Miesiąc październik przyniósł dobrą nowinę dla polskiej edukacji, ruszył program Prezesa Rady Ministrów „**Komputer dla ucznia**”. Program zakłada przekazanie komputerów typu „laptop” każdemu uczniowi szkoły gimnazjalnej, bez znaczenia jest fakt czy uczeń uczęszcza do szkoły publicznej czy też nie publicznej.

Kancelaria Prezesa Rady Ministrów odpowiedzialna za realizację programu współpracuje przy jego realizacji z poszczególnymi resortami, w tym Ministerstwem Finansów, Ministerstwem Infrastruktury, Ministerstwem Edukacji Narodowej, Ministerstwem Rozwoju Regionalnego. Ministerstwo Edukacji Narodowej odpowiedzialne za przygotowanie kadry nauczycielskiej do wykorzystania w toku edukacyjnym przekazanych uczniom laptopów bardzo szybko podjęło decyzję o przeprowadzeniu szkoleń dla nauczycieli.

W roku 2008 przeznaczono środki na szkolenia 2 tysięcy nauczycieli w każdym województwie. W przypadku naszego regionu szkolenia prowadziła Ogólnopolska Fundacja Edukacji Komputerowej przy wsparciu naszego Ośrodka. Ku naszemu pozytywnemu zaskoczeniu zainteresowanie udziałem w programie a poprzez to w szkoleniach było olbrzymie. Świadczy to o bardzo poważnym podejściu do programu szkół gimnazjalnych w naszym regionie.

Tak jak wspomniałem powyżej KPRM współpracuje przy realizacji programu z wieloma podmiotami, również i my mamy swój wkład w tą współpracę. WODIiP wraz z LO Otmuchów prezentował na posiedzeniu Zespołu Programowego KPRM projekt „e-Szkoła, szkołą z przyszłością”. Nasza prezentacja spotkała się z olbrzymim zainteresowaniem.

Na zlecenie KPRM WODIiP przygotował koncepcję projektu e-Szkoły i jego miejsca w programie „Komputer dla ucznia”. Sam program niesie wiele pytań i wyzwań, które nie są proste do rozwiązania. Jednym z nich jest wybór rodzaju komputera dla ucznia, czy ma

być to notebook, czy też zwykły laptop. Postawienie tego pytania producentom sprzętu, w perspektywie olbrzymich zakupów spowodowała prześciganie się producentów w pomysłach oraz nowościach technicznych. Rodzą się pytania, czy ograniczać możliwości techniczne na korzyść dłuższego działania baterii, czy też lepiej wybrać sprzęt dobrze wyposażony i rozwiązać problem ładownia baterii.

Kolejny problem to problem związany z zarządzeniem dostępem do sieci lokalnej a tym samym do internetu. Nie wiele osób wie, że pojawienie się w szkole w perspektywie 3-letniej ilości komputerów przenośnych równej liczbie uczniów to olbrzymie wyzwanie administracyjne. I to zarówno w zakresie dosłownego zrozumienia słowa administracja jak i kontekstu administrowania od strony informatycznej. Nie bez znaczenia jest również problem serwisowania sprzętu, wsparcia technicznego i jego ubezpieczenia, a więc problem własności.

W opracowaniu przekazanym do KPRM WODIiP zwrócił uwagę na fakt podstawowy, to najpierw nauczyciel musi mieć stworzone warunki wykorzystania „laptopa" w potem należy przekazywać sprzęt uczniom. Muszę przyznać, że nasze sugestie zostały przyjęte z należytą powagą i tu trzeba wspomnieć również o zasobach edukacyjnych, zarówno tych przygotowywanych przez nauczycieli jak i tych przygotowanych przez wydawnictwa. Zasoby edukacyjne muszą być udostępnione szkole albo na serwerze regionalnym albo na platformie wydawnictwa, jedno jest pewne nikt nie zrobi tego za darmo. Na organizację tego procesu również potrzebne są środki finansowe, które powinny być uruchomione przed dostarczeniem sprzętu.

WODIiP planuje złożenie w 2009r. do Regionalnego Programu Operacyjnego Priorytetu Społeczeństwo informacyjne projektu typu e-Szkoła, aby wesprzeć w naszym regionie działania KPRM. O zakresie naszego projektu i kolejności działań w następnym numerze.

Dzisiaj życzę wszystkim naszym czytelnikom Do siego Roku 2009.

Społeczeństwo informacyjne w edukacji[1]

dr Beata Bułka

Witam państwa serdecznie i jednocześnie dziękuję panu Lesławowi Tomczakowi Dyrektorowi WODIiP w Opolu za zaproszenie do wzięcia udziału w tak koniecznym w dzisiejszych czasach Forum na temat *Społeczeństwa informacyjnego*.

W swoim dzisiejszym wystąpieniu pragnę państwu przedstawić w okrojony, ale mam nadzieję trafnie, temat jakim jest: „Społeczeństwo informacyjne w edukacji".

Kilka słów wprowadzenia

Zbliżamy się do okrągłej rocznicy od ogłoszenia przez Ministerstwo Edukacji Narodowej reformy systemu edukacji w Polsce. Najwięcej uwagi poświęcono konstytucji programowej ale też strukturze systemu edukacji. Wyróżniono w niej etapy trójczłonowej drogi edukacyjnej i dla każdej z nich przypisano motto: aktywność (szkoła podstawowa), samodzielność (gimnazjum), dojrzałość (szkoła ponadgimnazjalna).

Reformę oparto na zasadach funkcjonowania współczesnych systemów szkolnych takich jak: demokratyczność (drożność, szeroki profil, ustawiczność, elastyczność i uspołecznienie kształcenia), prymat wartości i umiejętności nad wiadomościami, uczenia się alternatywnego i innowacyjnego. Odnoszą się one do aksjologicznych, teleologicznych, treściowych, infrastrukturalnych i ewaluacyjnych aspektów edukacji (Koszczyc, 2007).

Można również poddać weryfikacji idee i cele zawarte w podstawach programowych: dostosowanie poziomów kształcenia do okresów rozwoju dziecka, dbałość o ich rozwój fizyczny i zdrowotny, zapewnienie im lepszego rozumienia siebie, innych i świata; wyrównywanie szans edukacyjnych; poprawa jakości edukacji; podniesienie się poziomu społe-

[1] Główne tezy i zagadnienia powyższego artykułu zostały przedstawione na Ogólnopolskim Forum Społeczeństwa Informacyjnego, Opole 21-22.10.2008 r.

czeństwa przez upowszechnienie wykształcenia średniego i wyższego. Oparcie reformy systemu edukacji na tych ideach, celach i zadaniach jest trafniejsze, niż bazowanie na postulacie *wszechstronnego rozwoju uczniów,* wynikającego z Deklaracji Praw Człowieka ONZ. Nie można się rozwijać w każdym kierunku równocześnie i równomiernie, tak jak nie można równocześnie wchodzić wieloma szlakami turystycznymi na szczyt górski (Denek, 2007). Zatem celem edukacji nie jest wszechstronny, lecz wielostronny rozwój osobowości uczniów.

W Narodowym Planie Rozwoju 2007-2013 zaliczono edukację do ważnych inwestycji prorozwojowych. Zakłada się również podwyższenie kompetencji kluczowych i promowanie kształcenia z zakresu nauk ścisłych, ponieważ nauki te kształtują sposób rozumowania, kojarzenia, porządkowania i selekcji informacji, co jest niezbędne w społeczeństwie wiedzy.

Wróćmy na chwilę do kompetencji kluczowych ponieważ nawiązują one bezpośrednio do głównego zagadnienia mojego wystąpienia.

W czerwcu 2005 roku w Brukseli (grupa robocza powołana przez Komisję Europejską) zaproponowano następujący zestaw kompetencji, wymaganych od nauczyciela:

- pracy wielokulturowej i w zorganizowanej społecznie klasie
- tworzenie dogodnych warunków do uczenia się (ma być organizatorem procesu uczenia się i uczynić ze swych uczniów badaczy; tworzy programy nauczania, ciągle się szkoli i doskonali; stale usprawnia swoją pracę, działa we wszelkiego rodzaju stowarzyszeniach i organizacjach; jest animatorem życia społeczno-kulturalnego w regionie)
- włączenie technologii informacyjno-komunikacyjnej do codziennego funkcjonowania uczniów

- pracy w zespole (współpracy przy tworzeniu programów nauczania, organizacji procesu kształcenia i oceniania)
- współpracy ze środowiskiem lokalnym i rodzicami
- dostrzegania i rozwiązywania problemów
- ciągłego rozszerzania i pogłębiania swojej wiedzy i doskonalenia umiejętności
- wykształcenia w uczniach postawy obywatelskiej i społecznej
- programowania takiego rozwoju kompetencji uczniowskich, które pozwolą im z sukcesem funkcjonować w społeczeństwie wiedzy (motywacji do nauki, nauczania uczenia się, krytycznego przetwarzania informacji, posługiwania się komputerem i Internetem, twórczości i innowacyjności, rozwiązywania problemów, współpracy z innymi, łatwości komunikacji z innymi, poruszania się w kulturze wizualnej oraz przedsiębiorczości)
- łączenia kształtowania wymaganych kompetencji z nauczaniem, uczenia się danego przedmiotu.

Wspomniane gremium zaproponowało także *„europejskie”* zasady odnoszące się do zawodu nauczycielskiego. Jest to zawód: wymagający wyższego wykształcenia; osadzony w kontekście uczenia się przez całe życie (zasada 3*L -long, life, learning); mobilny; oparty na partnerstwie (Sielatycki, 2005).

Tak przygotowany do pracy nauczyciel będzie w stanie kształtować podobne kompetencje wśród uczniów. Na kompetencje kluczowe ucznia składają się: wiedza, umiejętności i postawy, w tym m.in. …”*Umiejętności posługiwania się nowoczesnymi technologiami informacji i komunikacji (ICT) z komputerem i Internetem na czele (w zakresie wyszukiwania, selekcjonowania i wykorzystywania danych” (Jurkiewicz, 2006).*

Żyjąc we współczesnym świecie dostrzegamy, że gdziekolwiek byśmy nie spojrzeli, wszędzie dotykamy zagadnień związanych ze społeczeństwem informacyjnym, również a może przede wszystkim w edukacji.

Co to takiego jest to społeczeństwo informacyjne?

Przedstawię państwu dwie z wielu (z ponad 100 jak podaje prof. Szczepański) definicji Społeczeństwa informacyjnego. Otóż:

Społeczeństwo informacyjne – to społeczeństwo, w którym towarem staje się informacja traktowana jako szczególne dobro niematerialne, równoważne lub cenniejsze nawet od dóbr materialnych. Przewiduje się rozwój usług związanych z 3P (przesyłanie, przetwarzanie, przechowywanie informacji).

Termin został wprowadzony w 1963 roku przez Japończyka T. Umesamo (wersja oryginalna "jōhōka shakai") w artykule o teorii ewolucji społeczeństwa opartego na technologiach informatycznych, a spopularyzowany przez K. Koyama w 1968 roku w rozprawie pt. "Wprowadzenie do Teorii Informacji" (Introduction to Information Theory). W Japonii powstał również "Plan utworzenia społeczeństwa informacyjnego, jako cel narodowy na rok 2000". Była to realna strategia zakładająca informatyzację kraju, prowadzącą do rozwoju intelektualnego kraju oraz tworzenia wiedzy, a nie dalsze uprzemysławianie kraju i wzrost dóbr materialnych.

Teorie rozwoju społecznego tłumaczą społeczeństwo informacyjne jako kolejny etap rozwoju społecznego, po społeczeństwie przemysłowym. Nazywane jest również mianem społeczeństwa post nowoczesnego, ponowoczesnego lub poprzemysłowego. Z punktu widzenia społecznego podziału pracy, społeczeństwem informacyjnym będzie nazywana zbiorowość w której 50% plus jedna osoba lub więcej, spośród zawodowoczynnych, zatrudnionych jest przy przetwarzaniu informacji. Daniel Bell określał pracę człowieka przednowoczesnego jako grę człowieka z przyrodą, człowieka nowoczesnego jako grę

człowieka z naturą nieożywioną a pracę człowieka ponowoczesnego jako grę miedzy ludźmi. Cechy charakterystyczne społeczeństwa informacyjnego to m.in.:

- wysoko rozwinięty sektor usług, przede wszystkim sektor usług nowoczesnych (bankowość, finanse, telekomunikacja, informatyka, badania i rozwój oraz zarządzanie), w niektórych krajach w tym sektorze pracuje przeszło 80% zawodowo czynnej ludności, przy czym sektor usług tradycyjnych przekracza nieznacznie 10%
- gospodarka oparta na wiedzy
- wysoki poziom skolaryzacji społeczeństwa
- wysoki poziom alfabetyzmu funkcjonalnego w społeczeństwie
- postępujący proces decentralizacji społeczeństwa
- renesans społeczności lokalnej
- urozmaicanie życia społecznego.

Formułowane są liczne teorie, m.in. Daniela Bella, Manuela Castellsa, które pozwalają rozważać różne warianty rozwoju społecznego - od pesymistycznych (system Georga Orwella, system rozproszony), poprzez "pośrednie" (m.in. społeczeństwa informacyjne charakterystyczne dla demokracji zachodnich), do optymistycznych (system wolności wyboru, system homeostatyczny) (Goban-Klass, Sienkiewicz, 1999).

Definicje społeczeństwa informacyjnego są wieloaspektowe. Rozpatrują to społeczeństwo np. w aspekcie technologicznym, jako kreowane przez Internet i jego możliwości (m.in. David Nicholas). Aspekt ekonomiczny społeczeństwa informacyjnego traktuje przetwarzanie informacji jako podstawę tworzenia dochodu narodowego i źródło utrzymania dla większości społeczeństwa (np. Tomasz Goban-Klas, Piotr Sienkiewicz). Aspekt demokratyczny tego społeczeństwa każe spojrzeć na nie, jak na społeczeństwo poinformowa-

ne, gdzie każdy ma prawo do informowania i bycia informowanym (A. Lepa). Peter Drucker naczelne znaczenie przypisuje w nim nie tylko informacji, lecz także wiedzy, określając je jako społeczeństwo wiedzy. Zaś Daniel Bell równie mocno podkreśla znaczenie wiedzy i edukacji, jako drogi dostępu do umiejętności i władzy (Kurek-Kokocińska, 2001).

Edukacja w społeczeństwie informacyjnym jest częstym przedmiotem dyskusji i rozważań. Wspomniane wcześniej liczne definicje społeczeństwa informacyjnego, opisując jego działanie, podkreślają wiodącą rolę informacji, a zwłaszcza kwestię dostępu do niej. Dostęp do informacji nie jest jedynym gwarantem dobrego funkcjonowania w społeczeństwie informacyjnym. Nie mniej ważne jest sprawne poruszanie się wśród informacji, polegające przede wszystkim na ocenie wartości informacji i ich źródeł. W tym miejscu należy wspomnieć o zjawisku – jak mówi prof. Teresa Hejnicka-Bezwińska- tzw. zaczadzenia smogiem informacyjnym, kojarzonym przede wszystkim z Internetem. To sytuacja, w której kompetencje użytkownika są niewystarczające do zakwalifikowania informacji, z którymi ma styczność, jako prawdziwych i istotnych lub nie. Ocena ta ma kluczowe znaczenie w podjęciu decyzji o ich wykorzystaniu bądź rezygnacji z użycia informacji. Wiąże się z umiejętnością filtracji informacji, zarówno na poziomie jednostki, grupy, jak i społeczeństwa. Receptą na wspomniane wyżej i inne problemy funkcjonowania w społeczeństwie informacyjnym ma być odpowiedni model edukacji. W ogólnym zarysie polegać ma ona na nadaniu uczestnikom społeczeństwa informacyjnego takiego poziomu kompetencji, który pozwoli im skutecznie i krytycznie przekształcić informację w wiedzę i wykorzystać ją. Edukacja nie może się, więc ograniczać do "kolekcjonowania" informacji (Hejnicka-Bezwińska, 2000)

W rozważaniach na temat edukacji w kontekście kształtowania polskiego społeczeństwa informacyjnego oraz integracji z Unią Europejską, jednym z haseł jest metanauczanie. Zadaniem placówek oświatowych wszystkich szczebli będzie *[...] przygotowanie absolwentów do samokształcenia i uczestnictwa w procesie edukacji obejmującej całe*

życie [...] (Banach, Rajkiewicz, 2002). Powraca wspomniana wyżej umiejętność przekształcania informacji w wiedzę. Ważne staje się również wpojenie umiejętności korzystania z szeroko pojętych mediów jako źródeł informacji i narzędzi służących kształceniu i rozwojowi, co wchodzi w zakres zadań edukacji medialnej. Postulowany model edukacji powinien łączyć w sobie, m.in. wychowanie humanistyczne i techniczne, zapewniać wszechstronny rozwój i nie dopuścić *[...] do przekształcenia homo sapiens w homo videns - człowieka postrzegającego świat prawie wyłącznie za pomocą obrazów [...]* (Banach, Rajkiewicz, 2002). Podkreśla się również rolę tzw. edukacji europejskiej, związanej z integracją Polski z Unią Europejską. Edukacja, co bardzo ważne, nie może być oderwana od realiów rynku pracy i kształcić armii przyszłych bezrobotnych. Ma polegać m. in. na odejściu od wąskiej specjalizacji w kształceniu zawodowym i wyższym oraz rozwoju kształcenia ustawicznego w stałej łączności z podmiotami gospodarczymi.

Szkolnictwo wyższe w Polsce ma również do spełnienia swoją misję edukacyjną. Jednym z ważniejszych elementów jest jego udział w tworzeniu społeczeństwa światłego, którego członkowie są przygotowani do ciągłego kształcenia się, związanego także z gotowością do ewentualnej zmiany zawodu. Tym samym mają intelektualne kompetencje do korzystania z edukacji ustawicznej i interaktywnej. To jedna z podstaw kształtowania społeczeństwa informacyjnego (Pelczar, 2001).

Nierozerwalnie z pojęciem społeczeństwo informacyjne pojawia się pojęcie technologia informacyjna.

Technologia informacyjna, **IT** (akronim od ang. Information Technology) – jest to dziedzina wiedzy obejmująca informatykę (włącznie ze sprzętem komputerowym oraz oprogramowaniem używanym do tworzenia, przesyłania, prezentowania i zabezpieczania informacji), telekomunikację oraz narzędzia i inne technologie związane z informacją. Dostarcza ona użytkownikowi narzędzi, za pomocą których może on pozyskiwać informacje, selekcjonować je, analizować, przetwarzać, zarządzać i przekazywać innym ludziom.

Tak rozumiana technologia informacyjna jest ważnym elementem społeczeństwa informacyjnego, nie tylko jako narzędzie pracy ale również jako element edukacji. Obecnie w Polsce na każdym etapie kształcenia a nawet w wielu przedszkolach prowadzi się zajęcia z technologii informacyjnej/informatyki po to aby przygotować młodego człowieka do życia w społeczeństwie opartym na wiedzy. Budzić może niepokój fakt, że jeszcze nie wszystkie szkoły mają pracownię komputerową na miarę XXI wieku. Mówię tu np. o szkole podstawowej z Opola (przez grzeczność nie wymienię której) miasta wojewódzkiego, która nie spełnia podstawowych kryteriów - siedem komputerów sprawnych w pracowni! (część uczniów z grupy siedzi na środku sali i czeka aż pozostali skończą swoją pracę przy komputerach, w międzyczasie żeby była jasność nie wykonywali żadnych zadań związanych z tematem lekcji!). Inna szkoła podstawowa w małej miejscowości województwa mazowieckiego nawet nie ma pracowni komputerowej. W jednej z klas stoi dwa a może już nawet trzy komputery a uczniów jest/było dwunastu! Zajęcia z informatyki odbywają się najczęściej jako zajęcia teoretycznie!

Nie dogonimy Europy takim tempem jakie mamy w chwili obecnej.

Wspomnę, w celach porównawczych, o programie amerykańskim zakładającym powszechne dostarczenie do klas szkolnych technologii informacyjnej, który opiera się na czterech filarach (Siemieniecki, 1998):

- nowoczesne multimedialne komputery i urządzenia wspomagające naukę będą dostępne dla każdego ucznia
- klasy będą połączone ze sobą oraz ze światem zewnętrznym (pełny dostęp do Internetu)
- oprogramowanie edukacyjne będzie integralną częścią programów, które powinny się odznaczać wysoką jakością oraz motywować do działania
- nauczyciele będą przygotowani do wykorzystania nowoczesnej technologii.

Wymienione filary są wspierane różnymi projektami m.in. projekt CSEP (Computer Science Education Projekt) służący badaniom nad opracowywaniem materiałów edukacyjnych oraz program „Zapytaj ERIC'a (Ask ERIC), którego celem jest dotarcie z wiadomościami edukacyjnymi do nauczycieli. Więcej informacji na temat tego programu znajdziecie państwo na stronie http://ericir.syr.edu

Innym przykładem dobrych rozwiązań w edukacji jeśli chodzi o tworzenie społeczeństwa informacyjnego jest Anglia czy bliższe nam Niemcy (niektóre Landy realizują TI jako integralne części poszczególnych przedmiotów wykorzystując do tego jedną z metod aktywizujących nauczanie mianowicie projekt) (Tomczak, 2008). W Anglii natomiast wprowadzono system dualny, gdzie wprowadzono Information Technology (technologię informacyjną) jako odrębny przedmiot lub jako integralną część przedmiotów nauczania, dając szkołom całkowitą swobodę wyboru (Siemińska-Łosko, 2006).

Mówiąc o społeczeństwie informacyjnym mamy na myśli również edukację innowacyjną składająca się z kolejnych trio, a są nimi: innowacyjny nauczyciel, innowacyjny uczeń i innowacyjna szkoła. Co rozumiemy pod pojęciem innowacyjność?

Innowacja to nic innego jak wprowadzenie czegoś nowego, charakteryzuje się nowatorstwem, reformą, ulepszeniem. Może ona dotyczyć wszelkich dziedzin i sfer oddziaływań w różnych kierunkach. Innowacyjne są więc wszystkie ulepszenia maszyn i urządzeń, reformy systemów, jak i tworzenie zupełnie nowych rzeczy, zjawisk lub wartości. Innowacje mogą dotyczyć zarówno najwyższych technologii, jak i elementów życia codziennego.

Jak podają źródła tylko ok. 10 % osób na świecie można zaliczyć do społeczeństwa kreatywnego. Musimy pamiętać, że innowacyjny nauczyciel to nie tylko nowatorski ale również otwarty, śledzący na bieżąco co się dzieje w koło i to nie tylko w sferze pracy (obowiązków) zawodowej ale i w sferze życia codziennego; nauczyciel krytyczny ale i komunikatywny. Jest jeszcze jedna ważna rzecz, mianowicie: działalność innowacyjna

zarówno nauczyciela, ucznia czy też szkoły nie może być jednorazowa. Innowacyjność to proces wieloetapowy, a może nawet wieloletni.

Pamiętać również należy, że innowacyjny uczeń to uczeń wymagający, zadający wiele pytań, dociekliwy, czasami denerwujący ale za to błyskotliwy, kreatywny, chcący zmieniać siebie i innych, chcący zmieniać świat. Jak określa Jan Polak to uczeń „kłopotliwy". Uczeń potrzebujący dobrego ukierunkowania czyli dobrego innowacyjnego nauczyciela, a co za tym idzie innowacyjnej szkoły, która daje możliwości do rozwoju, która stwarza szansę. I nie należy się martwić, że budując społeczeństwo informacyjne w oparciu o edukację czy za pomocą edukacji poprzez wprowadzanie technologii informacyjnej coś bezpowrotnie tracimy. Czasami można usłyszeć głosy w dyskusjach na forum, że wprowadzając technologię informacyjną do szkół zabieramy dzieciom możliwość samodzielnego wykonania czegoś (lub uczestniczenia w doświadczeniach) w ramach np. zajęć z techniki, fizyki czy chemii. Nic podobnego! Innowacyjny nauczyciel wykorzysta technologię informacyjną do pokazania rzeczy, których sam nie może lub nie jest w stanie zademonstrować ale na pewno nie zastąpić. Wykorzystanie technologii informacyjnej w edukacji to jedno a wiedza jak z niej umiejętnie korzystać to drugie. Wiele informacji uczniowie mogą zdobywać sami (poszukiwanie informacji), mogą uczyć się np. w pracy grupowej selekcjonowania informacji, a uczestnicząc w realizacji np. projektu doskonale uczą się wykorzystywania zdobytych informacji. I to cała filozofia wykorzystywania technologii informacyjnej na zajęciach lekcyjnych, pozalekcyjnych czy też pozaszkolnych.

Społeczeństwo informacyjne to także rodzice uczniów (byłych, obecnych i przyszłych). Dla nich też jest rzeczą bardzo istotną czy mogą na bieżąco śledzić pracę (działalność) szkoły jako instytucji ale też szkoły jako środowiska wychowawczego; śledzić pracę nauczycieli czy też oczywiście postępy swojego dziecka w nauce. Zadawać pytania na forum, czyli kontaktować się z innymi rodzicami („swojej" klasy lub całej społeczności szkolnej). Rodzice powinni mieć możliwość nie tylko osobistego kontaktu z wychowawcą, dyrekto-

rem, czy innymi nauczycielami/opiekunami swojego dziecka. Przecież bywają przypadki, że rodzice nie mogą z różnych przyczyn losowych uczestniczyć w zebraniach. Wtedy wystarczyłby e-mail, lub wgląd w dziennik lekcyjny czy też na odpowiednią podstronę www i wszystko byłoby oczywiste. Zaangażowanie rodziców w życie szkoły, o którym ostatnio mówi się najczęściej w sposób negatywny mogłoby ulec radykalnej zmianie. Rodzic sam mógłby decydować o tym kiedy i w jaki sposób, i z kim chce się spotkać, porozmawiać, jakie problemy rozwiązać, co zrobić dla szkoły (szczerość szkoły jeśli chodzi o jej potrzeby), itd. wymieniać można by wiele.

Podsumowując spróbujmy sobie odpowiedzieć na pytanie czy możemy powiedzieć, że społeczeństwo informacyjne jest już w edukacji, czy edukacja współtworzy społeczeństwo informacyjne?

Oczywiście, że tak, chociażby dlatego, że większość, czyli na pewno 50%+1 osoba (a tak zakłada definicja) społeczności szkolnej „pracuje" z informacją i wykorzystuje do tego celu technologię informacyjną. Miejmy nadzieję, że w najbliższym dziesięcioleciu będziemy mogli powiedzieć, że cała społeczność szkolna, czyli i nauczycieli, i uczniowie, i ich rodzice oraz administracja będzie społeczeństwem informacyjnym. Ale żeby to osiągnąć potrzebne są nie tylko pieniądze (pewnie wielu z państwa tu obecnych tak pomyślało) ale przede wszystkim chęci. W myśl powiedzenia „Chcieć to móc". Nauczyciele powinni starać się być kreatywnymi w ten sposób pokażą uczniom, że nauka, że szkoła nie musi być dla nich złem koniecznym. Ostatnie badania PISA z roku 2003 nie są optymistyczne, jeśli chodzi o postrzeganie szkoły przez uczniów. Z danych raportu polskiego wynika, że 59,8% uczniów negatywnie ocenia szkołę, dla 10% jest ona stratą czasu, aż 31,7% uczniów ocenia negatywnie to jak szkoła przygotowuje ich do dorosłego życia (w aglomeracjach wskaźnik ten jest jeszcze większy- ok. 50%) (Romaniuk, Sztabiński, 2003) 68,8% uczniów nie pozostaje w dobrych stosunkach z nauczycielami, 50% uczniów uważa, że nauczyciele nie interesują się ich sytuacją, ich życiem rodzinnym, warunkami

domowymi. 58,9% ujawnia słabe lub bardzo słabe poczucie przynależności do szkoły a 22% nie identyfikuje się z nią (Koszczyc, 2007)

W tym miejscu należy zadać pytanie dlaczego tak jest? Kazimierz Denek trafnie podaje jedną z nich. „Wielu nauczycielom z oporem przychodzi zrozumienie, że nauczanie i uczenie się to nie monolog pedagogiczny, na który składają się nakazy i zakazy, lecz rozumny dialog między nauczycielem a uczniem. Jak przekonać, że konieczna jest zmiana stosunku do uczniów, którzy czują, przeżywają, mają swoje problemy, które niejednokrotnie przerastają możliwości ich rozwiązania przez uczniów. Jak przekonać nauczycieli, że uczniowie myślą, spostrzegają, wyciągają wnioski, oceniają pracę nauczycieli i ich postawy" (Denek, 2005)

Moim skromnym zdaniem pomóc w tym może właśnie umiejętność komunikowania się (i z uczniem i jego rodzicem) oraz wprowadzenia do warsztatu pracy nauczyciela technologii informacyjnej, co pozwoli wspólnie tworzyć społeczność informacyjną.

Tylko dzięki wspólnej pracy nauczyciela z uczniem i ich rodzicami będziemy mogli przygotować uczniów charakteryzujących się: *przedsiębiorczością, zdecydowaniem i samodzielnością w podejmowaniu decyzji, inicjatywą, umiejętnością adaptacji do nowych warunków, operatywnością, łatwością uczenia się* (Denek, 2002).

I tak przygotowani, wszyscy razem, nauczyciele, uczniowie i ich rodzice, będziemy mogli sprostać wymaganiom współczesnego świata, poradzić sobie z zalewem informacji, by wreszcie zrealizować ideał: wiedzieć, umieć, wspólnie żyć i być[2].

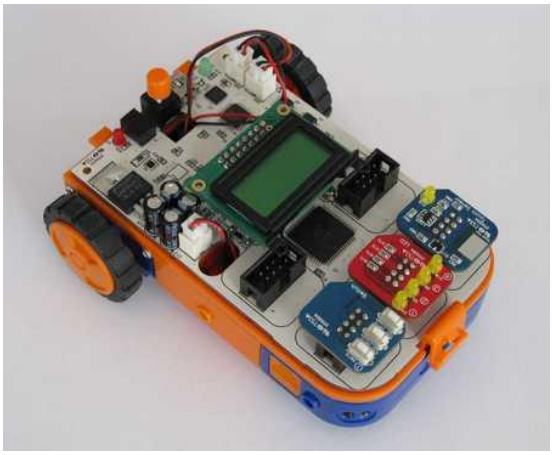
[2] Edukacja. Jest w niej ukryty skarb (Raport UNESCO Międzynarodowej Komisji do spraw Edukacji dla XXI wieku, pod przewodnictwem J. Delorsa), Warszawa 1999

LITERATURA:

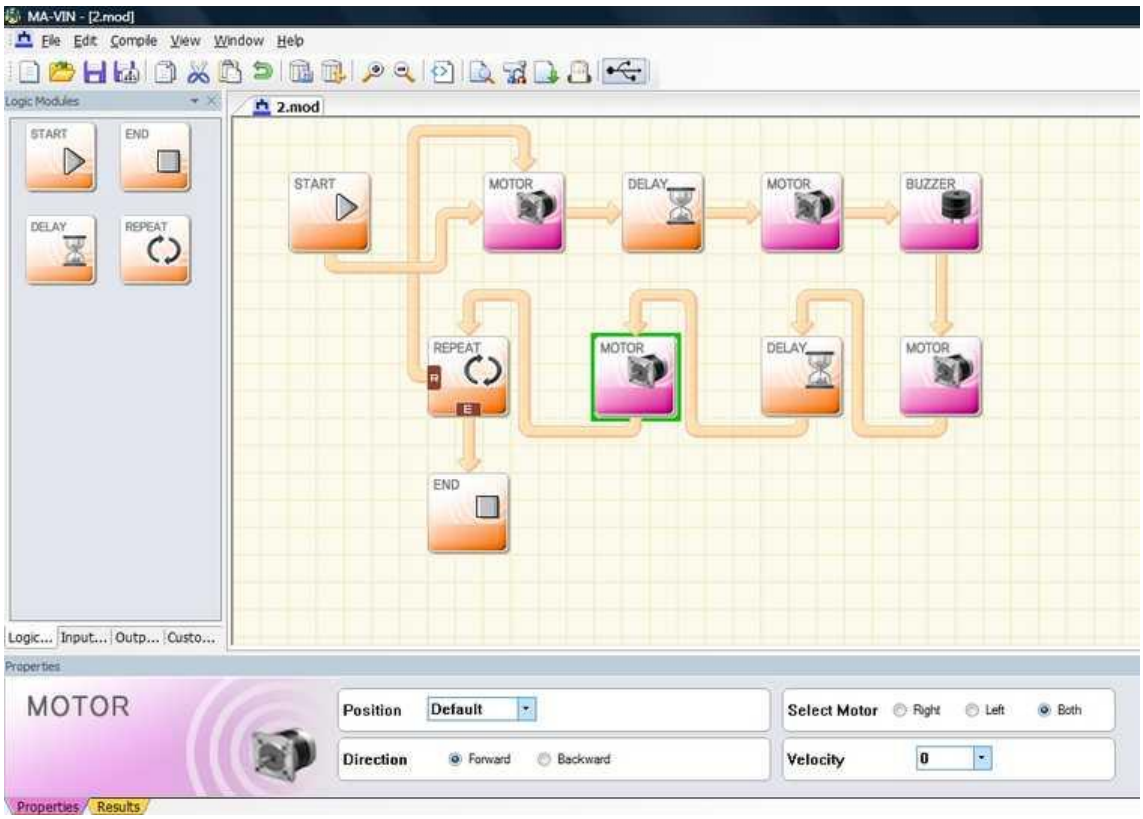
1. BANACH, Czesław, RAJKIEWICZ, Antoni. Najpilniejsze problemy do rozwiązania w systemie edukacji w latach 2004-2015. [w:] Strategia dla Polski po wejściu do Unii Europejskiej na lata 2004-2015: Polska w Unii Europejskiej: konferencja u Prezydenta RP Aleksandra Kwaśniewskiego w dniach 25-26 czerwca 2002. Warszawa: "Elipsa", 2002
2. DENEK, Kazimierz. *Ku dobrej edukacji*. Toruń-Leszno 2005
3. DENEK, Kazimierz. *Poza ławką szkolną*. Poznań 2002
4. GOBAN-KLAS, Tomasz, SIENKIEWICZ, Piotr. *Społeczeństwo informacyjne: szanse, zagrożenia, wyzwania*. Kraków: Wyd. Fundacji Postępu Telekomunikacji, 1999
5. HEJNICKA-BEZWIŃSKA, Teresa. *Imperatyw wykształcenia w społeczeństwie informatycznym*. [w:] Etos edukacji w XXI wieku: zbiór studiów. Pod red. Ireny Wojnar. Warszawa: "Elipsa", 2000
6. KOSZCZYC, Tadeusz, JONKISZ, Julian, TOCZEK-WERNER, Sylwia (red.). *Edukacja jutra*. Wyd. Wrocławskie Towarzystwo Naukowe. Wrocław 2007. ISBN 978-83734-048-8
7. KUREK-KOKOCIŃSKA, Stanisława. *Społeczeństwo biblioteczne jako społeczeństwo informacyjne*. Zagadnienia Informacji Naukowej. 2001, nr 2
8. KUŹNICKI, Leszek. *Nauka i edukacja w strategii rozwoju Polski do r. 2020*. Nauka. 2001, nr 4
9. PELCZAR, A. *Perspektywy szkolnictwa wyższego w Polsce: wyzwania i dylematy*. Nauka. 2001, nr 4
10. ROMANIUK, A., SZTABIŃSKI, P. *Uczniowie i ich szkoły (część raportu z III edycji badań jakości kształcenia pod egidą OECD/PISA),* Instytut Filozofii i Socjologii PAN, 2003
11. SIEMIENIECKI, Bolesław. *Technologia informacyjna w edukacji*, [w:] Strykowski, W. (red.), II Międzynarodowa Konferencja „Media a Edukacja", wyd. eMPi^2, Poznań 1998
12. SIEMIŃSKA-ŁOSKO, A. *Internet w przygotowaniu nauczycieli do stosowania technologii informacyjnej,* wyd. Adam Marszałek, Toruń 2006. ISBN 83-7441-454-5
13. *Strategia rozwoju Polski do roku 2020. T. 2, Studia eksperckie na temat 20-lecia 2001-2020*. Warszawa: "Elipsa", 2000. ISBN 83-7151-408-5 (a zwłaszcza KUPISIEWICZ, Czesław, BANACH, Czesław. Strategia rozwoju edukacji do r. 2020); STEFAŃSKA-MATUSZYN, Maria (oprac. red.). Strategia rozwoju Polski do roku 2020: synteza. Warszawa: "Elipsa", 2001. ISBN 83-7151-389-5
14. TOMCZAK, Lesław. *ICT w niemieckiej szkole*. J. Juszkiewicz, Z. Rudnicki, L. Tomczak,www.partnerstwodlaprzyszlosci.edu.pl/pdp/Shared%20Documents/audycje/audycje2008.htm, dostęp <18 październik 2008>

MA-VIN – robot, który nauczy algorytmiki i logicznego myślenia

mgr Janusz Podolak



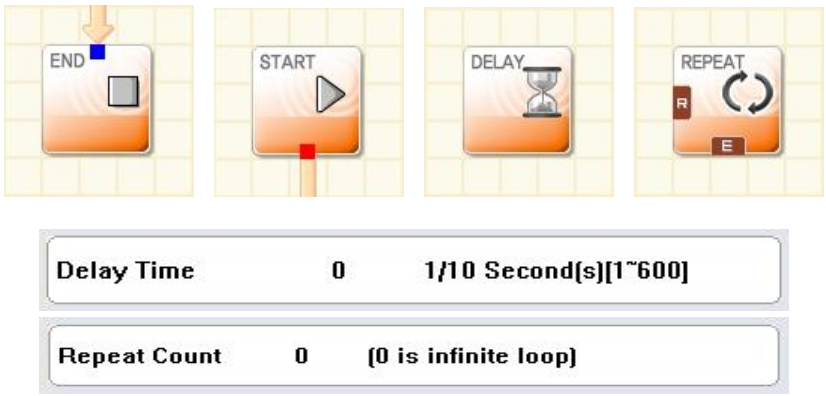
MA-VIN jest robotem, który można zaprogramować. W tym celu należy skorzystać z odpowiedniego kompilatora. Opisany tutaj pochodzi ze strony producenta Hitec Robotics. Jest to program Mavin Software Beta 2.0 (po zainstalowaniu program należy uruchomić jako administrator).



Programowanie odbywa się za pomocą modułów reprezentowanych przez ikony. Moduły podzielone są na logiczne, wejściowe i wyjściowe. Część z nich posiada dodatkowe parametry, które ustal użytkownik.

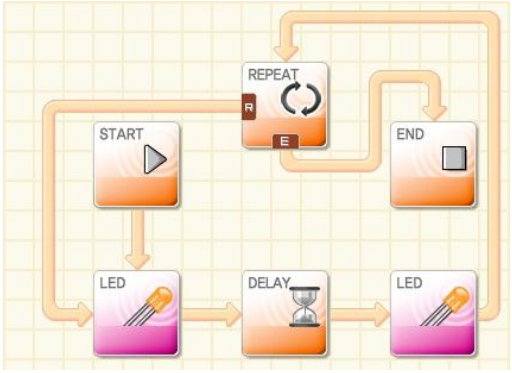
W zestawie odnajdziemy elementy, które można umieścić w specjalnych złączach na płycie głównej robota. Wszystkie te elementy mają swoje odpowiedniki w modułach programu.

Moduły logiczne



W każdym programie występuje moduł początkowy i końcowy (START i END), bez dodatkowych parametrów. Moduły te przeciwieństwie do pozostałych, w których znajdują się punkty wyjścia i wejścia, zawierają po jednym punkcie odpowiednio wyjścia\wejścia. Dla odróżnienia punkty wyjścia są czerwone a wejścia niebieskie.

Pozostałe moduły logiczne to DELAY i REPEAT. Wymagają one podania dodatkowych parametrów. W przypadku modułu DELAY ustalamy po jakim czasie ma być wykonana następna instrukcja. W module REPEAT podajemy liczbę powtórzeń programu, od wskazanego miejsca. Wartość „0” oznacz, że pętla będzie wykonywana w nieskończoność. Może ją przerwać ewentualnie inna instrukcja lub wyłączenie robota.

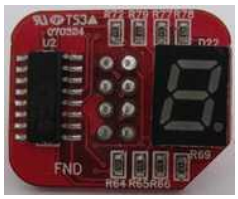


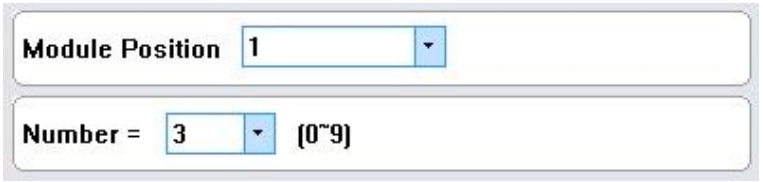
Rysunek obok przedstawia zastosowanie opisanych modułów. Występuje tu moduł wyjściowy LED, który zapala a następnie, po określonym w module DELAY czasie, wyłącza diody. Liczba powtórzeń zależy od parametru określonego w module REPEAT.

Moduły wyjścia

Moduły wyjścia wywołują określone reakcje. Tworząc program wydajemy, za ich pomocą polecenia robotowi. Na przykład: rusz do przodu, zaświeć diodę czy wyświetl komunikat.

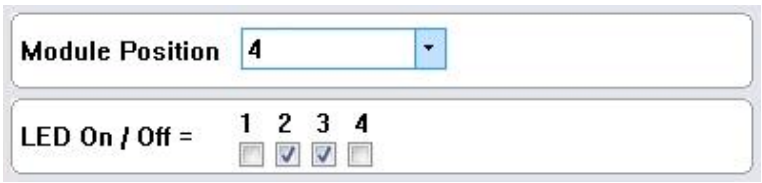
BUZZER – wydaje dźwięk. W ustawieniach parametrów ustalamy, w którym miejscu jest umieszczony na płycie (jedno z pięciu) oraz ile razy ma być odtworzony dźwięk.





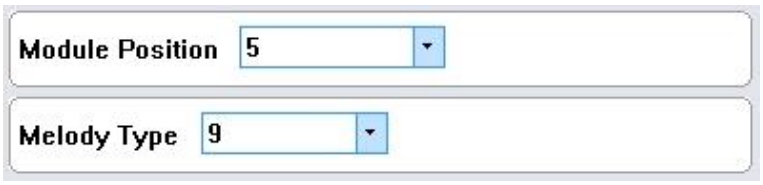
FND – wyświetla liczbę w zakresie od 0 do 9.

LCD – wyświetla wprowadzony komunikat w dwóch wiersz. Maksymalnie w jednym wierszu można umieścić osiem znaków.



LED – zapala diody. Do dyspozycji mamy cztery diody, które mogą być zapalone w dowolnej konfiguracji.


MOTOR – uruchamia silniki robota. Silniki są niezależne, dlatego można uruchomić obydwa albo tylko jeden z nich. Do wyboru mamy jeszcze kierunek obrotu oraz prędkość od 0 do 17.

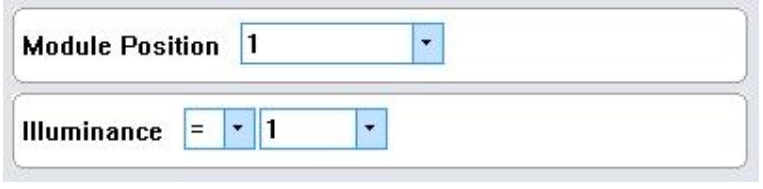



MELODY – odtwarza jedną z dziesięciu zaprogramowanych melodii.

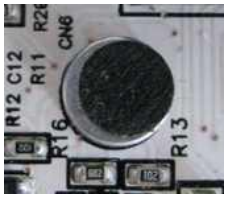

MUSIC – pozwala na zaprogramowanie własnej melodii. Jeden moduł odpowiada jednej nucie (taki sam element jak w module MELODY).

Moduły wejścia

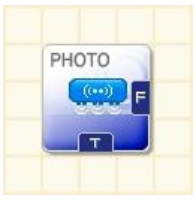
We wszystkich modułach wejścia następuje reakcja na czynnik zewnętrzny. Jeżeli taki czynnik występuje otrzymujemy wartość prawda (T) w przeciwnym wypadku fałsz (F).



CDS – czujnik oświetlenia. Reaguje na zmiany w oświetleniu. Można ustawić poziom zmian oświetlenia w skali od 1 do 5.



MIC – czujnik dźwięku. Reaguje na dźwięki. Czujnik jest wbudowany w płytę główną robota.



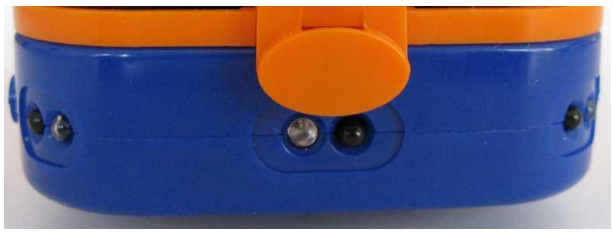
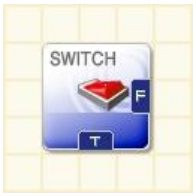
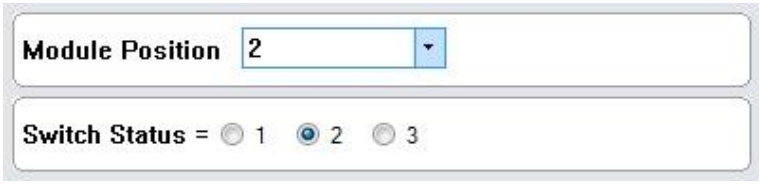
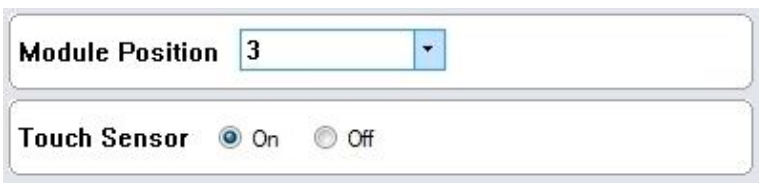


PHOTO – czujniki podczerwieni. Potrafią wykryć przeszkodę lub np. linię na drodze. Czujniki są na stałe wbudowane w robocie, po trzy z przodu i z dołu. Każdy z nich: prawy, środkowy i lewy można ustawić jako aktywny lub nieaktywny.





SWITCH – przyciski. Można ustawić, który z trzech przycisków ma spowodować dalszą realizację programu.



TOUCH – czujnik dotyku. Przy ustawieniu „On” dotknięcie spowoduje otrzymanie wartości prawda (T), natomiast przy ustawieniu ”Off” wartości fałsz (F).

Opis programu

Program jest prosty w obsłudze. Najistotniejsza jest inwencja twórcza autora programu. Moduły wybieramy i przemieszczamy „łapiąc" je lewym przyciskiem myszy. Połączenia miedzy modułami tworzymy korzystając z prawego przycisku myszy. W każdym module wyjście jest oznaczone kolorem czerwonym, a wejście kolorem niebieskim. Większość z modułów ma po dwa wejścia i wyjścia. Usunąć moduł można klawiszem DEL po uprzednim zaznaczeniu. Zaznaczony moduł ma zieloną obwódkę. Połączenie usuniemy tworząc nowe połączenie z tego samego punktu wyjścia. Moduły wybieramy w lewym oknie programu. Są one podzielone w grupy zgodnie z wcześniejszym opisem.

Pasek narzędzi


Nowy program. W głównym oknie programu pojawia się nowa plansza z modułem START.


Otwórz. Pozwala otworzyć gotowy, wcześniej zapisany program.


Zapisz. Zapisuje program w określonej przez użytkownika lokalizacji.


Zapisz moduł. Zapisuje program jako nowy moduł, który można później wykorzystać jako podprogram. Tak utworzone moduł odnajdziemy w lewym oknie, zakładka *Custom Modules*.


Kopiuj.


Wytnij.


Wklej.


Cofnij.


Usuń moduł.


Usuń połączenia. Dotyczy wszystkich połączeń wybranego modułu.


Powiększ ekran główny.


Pomniejsz ekran główny.


Kod źródłowy w języku C. Otwiera się w notatniku i można go edytować.


Logiczna poprawność. Sprawdza poprawność programu. Wskazuje np. błędne połączenia.


Kompilacja. Kompiluje gotowy program. Operacja niezbędna przed wczytaniem programu do pamięci robota.


Pobieranie. Pobieranie programu przez robota poprzez kabel USB.


Symulacja. Wywołuje pojawienie się okienka z prawej strony, w którym zilustrowane jest działanie programu.


Status USB. Jeżeli ikona jest zielona, to znaczy, że połączenie komputera z robotem jest aktywne. Tylko w takiej sytuacji można wgrać program do pamięci robota.

Pierwsza pomoc

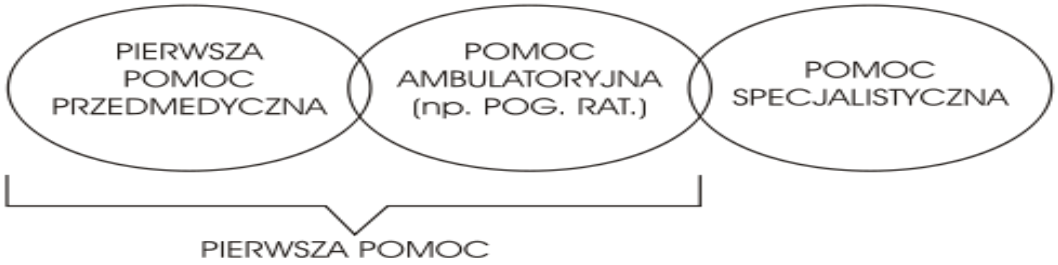
mgr Wiesław Bartoszek

Pierwsza pomoc — zespół czynności wykonywanych w razie urazu lub nagłego ataku choroby w celu zminimalizowania niekorzystnych następstw.

Pierwsza pomoc - rodzaje

- **pierwszą pomoc medyczna** (**przedlekarską**) - zespół czynności ratunkowych wykonywanych przez osoby bez wykształcenia medycznego
- **pierwszą pomoc lekarską** prowadzoną przez lekarza lub ratownika medycznego najczęściej w wyniku interwencji pogotowia ratunkowego.

Łańcuch ratunkowy



Łańcuch ratunkowy jest zespołem czynności ratowniczych zgrupowanych w trzy ogniwa:

- **pomoc przedlekarską** polegającą na udzielenie podstawowej pomocy na miejscu wypadku,
- **pierwszą pomoc lekarską** polegającą na dodatkowej, zaawansowanej pomocy medycznej oraz transporcie do placówki pomocy specjalistycznej, w ramach tej formy pomocy wymienia się także pomoc udzielaną na oddziałach ratunkowych
- **pomoc specjalistyczną** prowadzoną zwykle na wyspecjalizowanych oddziałach szpitalnych.

Zakres pierwszej pomocy

W zakres pierwszej pomocy przedlekarskiej wchodzą takie czynności jak:

1. zabezpieczenie miejsca wypadku
2. sprawdzenie stanu poszkodowanego
3. zapewnienie sobie pomocy, wezwanie pogotowia ratunkowego prowadzenie resuscytacji krążeniowo – oddechowej.
4. wykonanie innych czynności ratunkowych zależnych od stanu pacjenta

Zabezpieczenie miejsca wypadku

Zabezpieczenie miejsca wypadku ma na celu ochronę zarówno poszkodowanego, ratownika, jak i osób trzecich (gapiów, innych uczestników ruchu drogowego, itp.). Standardowo w wypadkach komunikacyjnych zatrzymuje się ruch na danym odcinku drogi. W tym celu na drodze, w odpowiednio oddalonym miejscu ustawia się trójkąt ostrzegawczy. Trójkąt ostrzegawczy w razie konieczności może być zastąpiony np. samochodem. W przypadku drgawek (np. epilepsji) konieczne jest usunięcie twardych przedmiotów.

Sprawdzenie stanu poszkodowanego. Funkcje życiowe

U pacjenta nieprzytomnego należy określić, czy oddycha, przykładając policzek nad jego usta, obserwując zarazem czy unosi się klatka piersiowa.). Przy określaniu innych nieprawidłowości kluczowe znaczenia ma obserwacja nieprzytomnego chorego. O ile pacjent jest przytomny, ratownik może spróbować zebrać wywiad - jest to istotne szczególnie przy chorobach przewlekłych (takich jak cukrzyca), jeśli to one spowodowały, że chory potrzebuje pomocy (np. podania glukozy).

Wzywanie pomocy

Wezwania pomocy należy dokonać po ustaleniu stanu poszkodowanego, ale przed rozpoczęciem udzielania pomocy. Zwykle zawiadamia się albo pogotowie ratunkowe, albo straż pożarną. Drugą ze służb wzywa się do wypadku, gdy potrzebne może być użycie specjalistycznego wyposażenia do bezpiecznego wyciągnięcia poszkodowanego, ugaszenia pożaru, neutralizacji wycieku z cysterny, baku samochodu, itd. Po zawiadomieniu straży pożarnej na miejsce wypadku przybędą zarazem strażacy, karetka pogotowia, jak i inne potrzebne służby (policja, pogotowie gazowe, itd).

Przy zgłaszaniu wezwania należy podać (istotna kolejność):

1. miejsce zdarzenia
2. rodzaj zdarzenia (wypadek drogowy, wypadek w pracy, etc.)
3. liczbę poszkodowanych
4. stan poszkodowanych
5. imię i nazwisko wzywającego pomocy
6. numer telefonu, z którego dzwonimy .

Bezpieczeństwo ratownika

Zarówno bezpieczeństwo ratownika, jak i osób postronnych jest priorytetem. Nie należy podejmować akcji ratunkowej, jeśli istnieje realne zagrożenie dla świadków zdarzenia (np. wybuchem). Należy pamiętać, że trudniej jest ratować dwie osoby niż jedną - z jednej ofiary i ratownika mogą nagle zrobić się dwie ofiary.

Ratownik powinien unikać kontaktu z krwią, ponieważ grozi to zakażeniem (głównie HBV, HCV i HIV) oraz używać specjalnych maseczek w czasie prowadzenia sztucznego oddychania.

Regulacje prawne

Obowiązek udzielania pomocy reguluje prawo. W Polsce, konsekwencje prawne za zaniechanie takiej pomocy przewiduje art. 162 KK:

> *§ 1. Kto człowiekowi znajdującemu się w położeniu grożącym bezpośrednim niebezpieczeństwem utraty życia albo ciężkiego uszczerbku na zdrowiu nie udziela pomocy, mogąc jej udzielić bez narażenia siebie lub innej osoby na niebezpieczeństwo utraty życia albo ciężkiego uszczerbku na zdrowiu, podlega karze pozbawienia wolności do lat 3.*

§ 2. Nie popełnia przestępstwa, kto nie udziela pomocy, do której jest konieczne poddanie się zabiegowi lekarskiemu albo w warunkach, w których możliwa jest niezwłoczna pomoc ze strony instytucji lub osoby do tego powołanej.

Wprowadzenie do programowania obiektowego na przykładzie języka Turbo Pascal

mgr Witold Rudolf

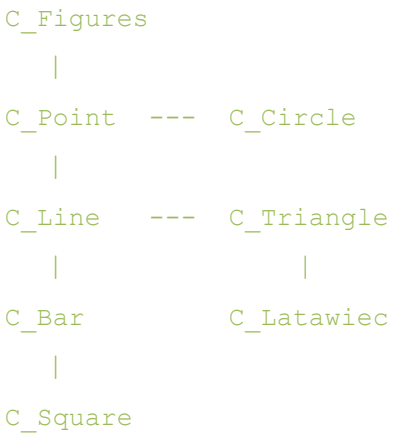
UWAGA: Jest to kontynuacja artykułu „Wprowadzenie do programowania obiektowego na przykładzie języka Turbo Pascal" zamieszczonego w poprzednim numerze Kwartalnika

Polimorfizm

Polimorfizm - zjawisko występowania różnych postaci wśród zwierząt lub roślin należących do tego samego gatunku, **wielopostaciowość**. (Słownik Wyrazów Obcych, PWN, Warszawa 1980).

Wielopostaciowość odnosi się do wielu obiektów z naszego świata, zarówno żywych, jak i nieożywionych. Wśród zwierząt np. mrówki występują w postaci robotnic, żołnierzy i królowych. A wśród tworów sztucznych? Ileż może być postaci omawianego na początku krzesła? Ile jest postaci trójkąta, kwadratu czy okręgu?

Dla naszego przykładu figur geometrycznych przyjmijmy, że klasy potomne są szczególnymi postaciami klasy `C_Figures`. Polimorfizm pozwala nam zatem traktować klasę `C_Triangle` lub `C_Square` jako polimorficzne z klasą `C_Figures`, gdyż, jak pamiętacie, przyjęliśmy następującą hierarchię:



A zatem C_Triangle i C_Square są potomkami klasy C_Figures. Każda klasa potomna do danej jest z nią polimorficzna. Ale uwaga: C_Line ani C_Square nie są polimorficzne z C_Circle, ponieważ nie są jej potomkami!

Co nam daje zastosowanie polimorfizmu? Pozwala nam na oszczędniejsze gospodarowanie pamięcią komputera i elastyczniejsze budowanie programu. Zastanówmy się. Gdybyśmy chcieli zmodyfikować program FIG-3 tak, aby można było wyświetlać dowolną figurę bez ingerencji w kod programu, konieczne byłoby zdefiniowanie osobnej zmiennej dla każdej klasy. Musielibyśmy zatem używać 7 zmiennych. Polimorfizm umożliwia nam dynamiczne korzystanie z jednej zmiennej. Wymaga to jednak wprowadzenia pojęcia **typu wskaźnikowego**.

Temat *wskaźniki* jest ogromny - w zasadzie zmienne wskaźnikowe pozwalają na zrobienie dowolnych rzeczy z pamięcią komputera. Nie będziemy rozwodzić się nad wszystkimi aspektami - skupimy się na **obiektach dynamicznych** i **obiektach polimorficznych**. Zacznijmy od tych pierwszych.

Obiekt dynamiczny pojawia się w pamięci komputera w wyniku wywołania procedury New i po wykorzystaniu obiektu może zostać usunięty z pamięci za pomocą procedury Done. Aby móc korzystać z obiektów dynamicznych, musimy zdefiniować **typ wskaźnikowy** dla danej klasy obiektów. Np. dla klasy C_Figures deklaracja typu wskaźnikowego wyglądałaby tak:

```
type
  P_Figures = ^C_Figures;     { wskaźnik do klasy obiektów }
  C_Figures=object            { jest to klasa obiektów }
              p0:PointType;
```

i tak dalej. Dla klas potomnych typ wskaźnikowy deklarujemy podobnie - P_Point wskazuje na C_Point, P_Circle wskazuje na C_Circle itd. Jeżeli teraz w programie głównym naszą

zmienną `F` zadeklarujemy nie jako obiekt konkretnej klasy, ale jako wskaźnik do klasy `C_Figures`, czyli tak:

```
var
  F : P_Figures;
```

to uzyskamy w ten sposób dostęp do **dowolnego** obiektu klasy potomnej do `C_Figures`. Jak? Dzięki polimorfizmowi. Wystarczy przy inicjowaniu zmiennej `F` podawać jej konstruktor z odpowiedniej klasy, np tak:

RYSOWANIE OKRĘGU

...

```
InitGraph(dr,gr,'');  { zainicjowanie trybu graficznego }
F := New(P_Circle, Init(100,100,50);
```

... (ciąg dalszy programu)

RYSOWANIE PROSTOKĄTA

...

```
InitGraph(dr,gr,'');  { zainicjowanie trybu graficznego }
F := New(P_Bar, Init(100, 100, 90, 50));
```

... (ciąg dalszy programu)

RYSOWANIE LATAWCA

...

```
InitGraph(dr,gr,'');  { zainicjowanie trybu graficznego }
F := New(P_latawiec, Init(100, 100, 30, 50));
```

... (ciąg dalszy programu)

Zapis

```
F := New(P_Circle, Init(100,100,50);
```

jest interpretowany jako polecenie utworzenia w pamięci komputera za pomocą konstruktora `Init` obiektu klasy wskazywanej przez `P_Circle` i przełączenie wskaźnika zmiennej `F` na ten konkretny obiekt. Do czasu wykonania `New` zmienna `F` nie wskazuje na żaden obiekt i próby odwołania do tej zmiennej mogą zakończyć się dowolnie - nawet blokadą komputera.

UWAGA: F jest wskaźnikiem, `F^` - obiektem, dlatego też, aby poruszać obiektem zamiast `F.Move` musimy użyć instrukcji `F^.Move` .

Utworzony obiekt dynamiczny likwiduje się procedurą Dispose podając nazwę zmiennej wskaźnikowej oraz nazwę destruktora.

```
Dispose(F, Done);    { likwidacja obiektu dynamicznego }
```

Po wykonaniu tej instrukcji obiekt wskazywany dotąd przez F jest usuwany z pamięci i F nie wskazuje już na żaden konkretny obiekt.

Pełny tekst programu znajduje się w pliku FIG-4.PAS

```
Program Figures;
{---------------------------------------------
Ilustracja do tekstu o programowaniu obiektowym
            Fig-4.pas
       Witold Rudolf 2006

       Definicja klasy C_Figures
                       C_Point
                       C_Circle
                       C_Line
                       C_Triangle
                       C_Bar
                       C_Square
              oraz     C_Latawiec


Ilustracja do dynamicznych klas oraz do polimorfizmu
```

```
     Hierarchia klas:  C_Figures
                           |
                       C_Point  ---  C_Circle
                           |
                       C_Line   ---  C_Triangle
                          |                |
                       C_Bar            C_Latawiec
                          |
                       C_Square

-----------------------------------------------}
Uses Mouse,graph;

Type
 P_Figures = ^C_Figures;          {wskaznik do klasy obiektow}
 C_Figures=object                                {jest to klasa obiektow}
             p0:PointType;                       {kazda figura musi zawierac}
                                                 {co najmniej jeden punkt}
             Procedure Draw(color:Integer);virtual; {kreslenie figury}
             Procedure Show;                     {metoda narysowania}
             Procedure Hide;                     {metoda zmazania}
             Procedure Move;                     {metoda poruszania}

             Constructor Init;                   {metoda tworzaca obiekt}
             Destructor Done;                    {metoda niszczaca obiekt}
           end;                                  {koniec definicji klasy obiektow}


 P_Point = ^C_Point;              {wskaznik do klasy obiektow}
 C_Point=object(C_Figures)                       {klasa potomna klasy C_Figures}
           Procedure Draw(color:Integer);virtual;   {kreslenie punktu}
           Constructor Init(x,y:Integer);
         end;                                    {koniec definicji klasy obiektow}

 P_Circle = ^C_Circle;            {wskaznik do klasy obiektow}
 C_Circle=object(C_Point)
            r:Integer;
            Procedure Draw(color:Integer);virtual;
            Constructor Init(x,y,promien:Integer);
          end;

 P_Line = ^C_Line;                {wskaznik do klasy obiektow}
 C_Line=object(C_Point)
             dx, dy:Integer;
             Procedure Draw(color:Integer);virtual;
             Constructor Init(x,y,px,py:Integer);
            end;

 P_Triangle = ^C_Triangle;        {wskaznik do klasy obiektow}
 C_Triangle=object(C_Line)
             dx1, dy1: Integer;
             Procedure Draw(color:Integer);virtual;
             Constructor Init(x,y,ox1,oy1,ox2,oy2:Integer);
            end;

 P_Bar = ^C_Bar;                  {wskaznik do klasy obiektow}
 C_Bar=object(C_Line)
             Procedure Draw(color:Integer);virtual;
             Constructor Init(x,y,bx,by:Integer);
            end;

 P_Square = ^C_Square;            {wskaznik do klasy obiektow}
 C_Square=object(C_Bar)
             Procedure Draw(color:Integer);virtual;
             Constructor Init(x,y,bx:Integer);
            end;

 P_Latawiec = ^C_Latawiec;        {wskaznik do klasy obiektow}
```

```
 C_Latawiec=object(C_Triangle)
             Procedure Draw(color:Integer);virtual;
             Constructor Init(x, y, ox, oy:Integer);
            end;

{-----------DEFINICJA METOD DLA KLASY C_FIGURES------------}
Procedure C_Figures.Draw;
begin {-----------------------C_Figures.Draw}
end;  {-----------------------C_Figures.Draw}

Procedure C_Figures.Show;
begin {-----------------------C_Figures.Show}
 Draw(white);
end;  {-----------------------C_Figures.Show}

Procedure C_Figures.Hide;
begin {-----------------------C_Figures.Hide}
 Draw(black);
end;  {-----------------------C_Figures.Hide}

Procedure C_Figures.Move;
begin {-----------------------C_Figures.Move}
 Mouse.Show;
 While not Rightbutton do     {dopoki nie nacisnieto prawego guzika}
 begin
  if Leftbutton then          {jesli nacisnieto lewy guzik, to wtedy}
  begin
   Hide;                      {usun ostatni obraz obiektu}
   p0.x:=Xpos;                {wsp. x punktu p0 odpowiada wsp. x myszy}
   p0.y:=Ypos;                {wsp. y punktu p0 odpowiada wsp. y myszy}
   Show;                      {wyswietl obiekt na nowej pozycji}
  end
 end;
 Mouse.Hide
end;  {-----------------------C_Figures.Move}

Constructor C_Figures.Init;
begin {-----------------------C_Figures.Init}
 Show;
end;  {-----------------------C_Figures.Init}

Destructor C_Figures.Done;
begin {-----------------------C_Figures.Done}
 Hide;
end;  {-----------------------C_Figures.Done}

{-----------KONIEC DEFINICJI METOD DLA KLASY C_FIGURES------------}

{---------------DEFINICJA METOD DLA KLASY C_POINT----------------}
Procedure C_Point.Draw;
{zapala na ekranie punkt p0 w kolorze COLOR}
begin {-----------------------C_Point.Draw}
 PutPixel(p0.x,p0.y,color);
end;  {-----------------------C_Point.Draw}

Constructor C_Point.Init;
begin {-----------------------C_Point.Init}
 p0.x:=x;
 p0.y:=y;
 Show
end;  {-----------------------C_Point.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_POINT--------------}


{---------------DEFINICJA METOD DLA KLASY C_Circle---------------}
Procedure C_Circle.Draw;
{zapala na ekranie okrag w kolorze COLOR}
begin {-----------------------C_Circle.Draw}
```

```
 SetColor(Color);
 circle(p0.x,p0.y,r);
end;  {-----------------------C_Circle.Draw}

Constructor C_Circle.Init;
begin {-----------------------C_Circle.Init}
 p0.x:=x;
 p0.y:=y;
 r:=promien;
 Show
end;  {-----------------------C_Circle.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Circle--------------}


{----------------DEFINICJA METOD DLA KLASY C_Line----------------}
Procedure C_Line.Draw;
{zapala na ekranie odcinek w kolorze COLOR}
begin {-----------------------C_Line.Draw}
 SetColor(color);
 line(p0.x,p0.y,p0.x+dx,p0.y+dy);
end;  {-----------------------C_Line.Draw}

Constructor C_Line.Init;
begin {-----------------------C_Line.Init}
 p0.x:=x;
 p0.y:=y;
 dx:=px;
 dy:=py;
 Show
end;  {-----------------------C_Line.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Line--------------}


{----------------DEFINICJA METOD DLA KLASY C_Triangle----------------}
Procedure C_Triangle.Draw;
{zapala na ekranie trojkat w kolorze COLOR}
begin {-----------------------C_Triangle.Draw}
 SetColor(color);
 Line(p0.x, p0.y, p0.x+dx, p0.y+dy);
 Line(p0.x, p0.y, p0.x+dx1, p0.y+dy1);
 Line(p0.x+dx, p0.y+dy, p0.x+dx1, p0.y+dy1);
end;  {-----------------------C_Triangle.Draw}

Constructor C_Triangle.Init;
begin {-----------------------C_Triangle.Init}
 dx1:=ox2;
 dy1:=oy2;
 Inherited Init(x, y, ox1, oy1);
end;  {-----------------------C_Triangle.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Triangle--------------}

{----------------DEFINICJA METOD DLA KLASY C_Bar----------------}
Procedure C_Bar.Draw;
{zapala na ekranie prostokat w kolorze COLOR}
begin {-----------------------C_Bar.Draw}
 SetFillStyle(SolidFill, color);
 Bar(p0.x,p0.y,p0.x-dx,p0.y-dy);
end;  {-----------------------C_Bar.Draw}

Constructor C_Bar.Init;
begin {-----------------------C_Bar.Init}
 Inherited Init(x, y, bx, by)
end;  {-----------------------C_Bar.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Bar--------------}

{----------------DEFINICJA METOD DLA KLASY C_Square----------------}
```

```
Procedure C_Square.Draw;
{zapala na ekranie kwadrat w kolorze COLOR}
begin {-----------------------C_Square.Draw}
 SetFillStyle(SolidFill, color);
 Bar(p0.x, p0.y, p0.x-dx, p0.y-dy);
end;  {-----------------------C_Square.Draw}

Constructor C_Square.Init;
begin {-----------------------C_Square.Init}
 Inherited Init(x, y, bx, bx)
end;  {-----------------------C_Square.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Square--------------}

{---------------DEFINICJA METOD DLA KLASY C_Latawiec----------------}
Procedure C_Latawiec.Draw;
{zapala na ekranie latawiec w kolorze COLOR}
begin {-----------------------C_LatawiecTriangle.Draw}
 SetColor(color);
 Line(p0.x,p0.y,p0.x-dx,p0.y+dy);
 Line(p0.x,p0.y,p0.x+dx,p0.y+dy);

 Line(p0.x-dx,p0.y+dy,p0.x,p0.y+(3*dy));
 Line(p0.x+dx,p0.y+dy,p0.x,p0.y+(3*dy));
end;  {-----------------------C_Latawiec.Draw}

Constructor C_Latawiec.Init;
begin {-----------------------C_Latawiec.Init}
 C_Line.Init(x, y, ox, oy)
end;  {-----------------------C_Latawiec.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Latawiec--------------}

var
 F     : P_Figures;   {zmienna-konkretny obiekt klasy potomnej do C_Figures}
 dr,gr : Integer; {zmienne inicjujace tryb graficzny}

BEGIN
 dr:=detect;gr:=0;
 InitGraph(dr,gr,''); {zainicjowanie trybu graficznego}


 F := New(P_Bar, Init(100,100,50,70));
                                  {utworzenie obiektu dynamicznego}
                                  {wystarczaja zmiany w tym miejscu}
                                  {dla uaktywnienia innego obiektu}

 F^.Move;                         {poruszanie obiektem}
 Dispose(F, Done);                {likwidacja obiektu dynamicznego}

 CloseGraph;              {wylaczenie trybu graficznego}
END.
```

Podsumowanie

Jako podsumowanie proponujemy tekst programu z pliku FIG-5.PAS, który jest po prostu poprzednim przykładem wzbogaconym o prostą ofertę umożliwiającą wybranie figury, która ma zostać wyświetlona.

W praktyce programistycznej najczęściej wykorzystuje się klasy obiektów dynamicznych ze względu na ich pamięciooszczędność - obiekt istnieje tylko wtedy, gdy jest konieczny.

Co najciekawsze, zdefiniowanie kolejnych obiektów potomnych nie powoduje istotnego wzrostu kodu programu - klasa `C_Latawiec` to tylko ok. 300 bajtów kodu maszynowego. Podobnie skonstruowanie kolejnych klas - przy odpowiednim zaprojektowaniu klas nadrzędnych - jest o wiele prostsze. Stąd tak duża popularność programowania obiektowego w konstruowaniu złożonego oprogramowania. Po prostu, jeżeli mamy obiekt, który prawie dobrze realizuje potrzebne nam funkcje, tworzymy obiekt potomny dodając nowe atrybuty i funkcje, nie ryzykując przy tym zaburzenia pracy już działających, ba, często nie znając kodu źródłowego!. W przypadku programowania strukturalnego konieczne było posiadanie dostępu do kodu źródłowego i zmodyfikowanie go do własnych potrzeb, co często było trudne i powodowało błędne działanie poprawnej dotychczas procedury.

```
Program Figures;
{-----------------------------------------------
Ilustracja do tekstu o programowaniu obiektowym
            Fig-5.pas
       Witold Rudolf 2006

       Definicja klasy C_Figures
                       C_Point
                       C_Circle
                       C_Line
                       C_Triangle
                       C_Bar
                       C_Square
              oraz     C_Latawiec


Ilustracja do dynamicznych klas oraz do polimorfizmu
z wyborem obiektu z prostego menu

     Hierarchia klas:  C_Figures
                           |
                       C_Point  ---  C_Circle
                           |
                       C_Line   ---  C_Triangle
                          |               |
                       C_Bar          C_Latawiec
                          |
                       C_Square

-----------------------------------------------}
Uses Mouse,graph;



Type
 P_Figures = ^C_Figures;         {wskaznik do klasy obiektow}
 C_Figures=object                                {jest to klasa obiektow}
             p0:PointType;                       {kazda figura musi zawierac}
                                                 {co najmniej jeden punkt}
```

```
              Procedure Draw(color:Integer);virtual; {kreslenie figury}
              Procedure Show;                        {metoda narysowania}
              Procedure Hide;                        {metoda zmazania}
              Procedure Move;                        {metoda poruszania}

              Constructor Init;                      {metoda tworzaca obiekt}
              Destructor Done;                       {metoda niszczaca obiekt}
            end;                                     {koniec definicji klasy obiektow}


  P_Point = ^C_Point;             {wskaznik do klasy obiektow}
  C_Point=object(C_Figures)                          {klasa potomna klasy C_Figures}
            Procedure Draw(color:Integer);virtual;   {kreslenie punktu}
            Constructor Init(x,y:Integer);
          end;                                       {koniec definicji klasy obiektow}

  P_Circle = ^C_Circle;           {wskaznik do klasy obiektow}
  C_Circle=object(C_Point)
             r:Integer;
             Procedure Draw(color:Integer);virtual;
             Constructor Init(x,y,promien:Integer);
           end;

  P_Line = ^C_Line;               {wskaznik do klasy obiektow}
  C_Line=object(C_Point)
              dx, dy:Integer;
              Procedure Draw(color:Integer);virtual;
              Constructor Init(x,y,px,py:Integer);
             end;

  P_Triangle = ^C_Triangle;       {wskaznik do klasy obiektow}
  C_Triangle=object(C_Line)
              dx1, dy1: Integer;
              Procedure Draw(color:Integer);virtual;
              Constructor Init(x,y,ox1,oy1,ox2,oy2:Integer);
             end;

  P_Bar = ^C_Bar;                 {wskaznik do klasy obiektow}
  C_Bar=object(C_Line)
              Procedure Draw(color:Integer);virtual;
              Constructor Init(x,y,bx,by:Integer);
             end;

  P_Square = ^C_Square;           {wskaznik do klasy obiektow}
  C_Square=object(C_Bar)
              Procedure Draw(color:Integer);virtual;
              Constructor Init(x,y,bx:Integer);
             end;

  P_Latawiec = ^C_Latawiec;       {wskaznik do klasy obiektow}
  C_Latawiec=object(C_Triangle)
              Procedure Draw(color:Integer);virtual;
              Constructor Init(x, y, ox, oy:Integer);
             end;

{-----------DEFINICJA METOD DLA KLASY C_FIGURES------------}
Procedure C_Figures.Draw;
begin {----------------------C_Figures.Draw}
end;  {----------------------C_Figures.Draw}



Procedure C_Figures.Show;
begin {----------------------C_Figures.Show}
 Draw(white);
end;  {----------------------C_Figures.Show}
```

```
Procedure C_Figures.Hide;
begin {----------------------C_Figures.Hide}
 Draw(black);
end;  {----------------------C_Figures.Hide}

Procedure C_Figures.Move;
begin {----------------------C_Figures.Move}
 Mouse.Show;
 While not Rightbutton do     {dopoki nie nacisnieto prawego guzika}
 begin
  if Leftbutton then          {jesli nacisnieto lewy guzik, to wtedy}
  begin
   Hide;                      {usun ostatni obraz obiektu}
   p0.x:=Xpos;                {wsp. x punktu p0 odpowiada wsp. x myszy}
   p0.y:=Ypos;                {wsp. y punktu p0 odpowiada wsp. y myszy}
   Show;                      {wyswietl obiekt na nowej pozycji}
  end
 end;
 Mouse.Hide
end;  {----------------------C_Figures.Move}

Constructor C_Figures.Init;
begin {----------------------C_Figures.Init}
 Show;
end;  {----------------------C_Figures.Init}

Destructor C_Figures.Done;
begin {----------------------C_Figures.Done}
 Hide;
end;  {----------------------C_Figures.Done}

{----------KONIEC DEFINICJI METOD DLA KLASY C_FIGURES-----------}

{----------------DEFINICJA METOD DLA KLASY C_POINT----------------}
Procedure C_Point.Draw;
{zapala na ekranie punkt p0 w kolorze COLOR}
begin {----------------------C_Point.Draw}
 PutPixel(p0.x,p0.y,color);
end;  {----------------------C_Point.Draw}

Constructor C_Point.Init;
begin {----------------------C_Point.Init}
 p0.x:=x;
 p0.y:=y;
 Show
end;  {----------------------C_Point.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_POINT--------------}


{----------------DEFINICJA METOD DLA KLASY C_Circle----------------}
Procedure C_Circle.Draw;
{zapala na ekranie okrag w kolorze COLOR}
begin {----------------------C_Circle.Draw}
 SetColor(Color);
 circle(p0.x,p0.y,r);
end;  {----------------------C_Circle.Draw}

Constructor C_Circle.Init;
begin {----------------------C_Circle.Init}
 p0.x:=x;
 p0.y:=y;
 r:=promien;
 Show
end;  {----------------------C_Circle.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Circle--------------}


{----------------DEFINICJA METOD DLA KLASY C_Line----------------}
```

```
Procedure C_Line.Draw;
{zapala na ekranie odcinek w kolorze COLOR}
begin {-----------------------C_Line.Draw}
 SetColor(color);
 line(p0.x,p0.y,p0.x+dx,p0.y+dy);
end;  {-----------------------C_Line.Draw}

Constructor C_Line.Init;
begin {-----------------------C_Line.Init}
 p0.x:=x;
 p0.y:=y;
 dx:=px;
 dy:=py;
 Show
end;  {-----------------------C_Line.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Line--------------}


{----------------DEFINICJA METOD DLA KLASY C_Triangle----------------}
Procedure C_Triangle.Draw;
{zapala na ekranie trojkat w kolorze COLOR}
begin {-----------------------C_Triangle.Draw}
 SetColor(color);
 Line(p0.x, p0.y, p0.x+dx, p0.y+dy);
 Line(p0.x, p0.y, p0.x+dx1, p0.y+dy1);
 Line(p0.x+dx, p0.y+dy, p0.x+dx1, p0.y+dy1);
end;  {-----------------------C_Triangle.Draw}

Constructor C_Triangle.Init;
begin {-----------------------C_Triangle.Init}
 dx1:=ox2;
 dy1:=oy2;
 Inherited Init(x, y, ox1, oy1);
end;  {-----------------------C_Triangle.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Triangle--------------}

{----------------DEFINICJA METOD DLA KLASY C_Bar----------------}
Procedure C_Bar.Draw;
{zapala na ekranie prostokat w kolorze COLOR}
begin {-----------------------C_Bar.Draw}
 SetFillStyle(SolidFill, color);
 Bar(p0.x,p0.y,p0.x+dx,p0.y+dy);
end;  {-----------------------C_Bar.Draw}

Constructor C_Bar.Init;
begin {-----------------------C_Bar.Init}
 Inherited Init(x, y, bx, by)
end;  {-----------------------C_Bar.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Bar--------------}

{----------------DEFINICJA METOD DLA KLASY C_Square----------------}
Procedure C_Square.Draw;
{zapala na ekranie kwadrat w kolorze COLOR}
begin {-----------------------C_Square.Draw}
 SetFillStyle(SolidFill, color);
 Bar(p0.x, p0.y, p0.x+dx, p0.y+dy);
end;  {-----------------------C_Square.Draw}

Constructor C_Square.Init;
begin {-----------------------C_Square.Init}
 Inherited Init(x, y, bx, bx)
end;  {-----------------------C_Square.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Square--------------}

{----------------DEFINICJA METOD DLA KLASY C_Latawiec----------------}
Procedure C_Latawiec.Draw;
{zapala na ekranie latawiec w kolorze COLOR}
begin {-----------------------C_LatawiecTriangle.Draw}
```

```
 SetColor(color);
 Line(p0.x,p0.y,p0.x-dx,p0.y+dy);
 Line(p0.x,p0.y,p0.x+dx,p0.y+dy);

 Line(p0.x-dx,p0.y+dy,p0.x,p0.y+(3*dy));
 Line(p0.x+dx,p0.y+dy,p0.x,p0.y+(3*dy));
end;  {----------------------C_Latawiec.Draw}

Constructor C_Latawiec.Init;
begin {----------------------C_Latawiec.Init}
 C_Line.Init(x, y, ox, oy)
end;  {----------------------C_Latawiec.Init}
{-------------KONIEC DEFINICJi METOD DLA KLASY C_Latawiec--------------}

var
 F     : P_Figures;   {zmienna-konkretny obiekt klasy potomnej do C_Figures}
 dr,gr : Integer; {zmienne inicjujace tryb graficzny}
 wybor: Integer;  {dla ustalania rodzaju figury}

BEGIN
  { wybieranie figury }
  WriteLn('Kreslenie figur');
  WriteLn('Figure mozna:');
  WriteLn('- przesunac kursorem myszy po nacisnieciu Lewego przycisku');
  Writeln('- wylaczyc po nacisnieciu Prawego przycisku');
  WriteLn(' -------- podaj numer figury -------');
  Writeln('1. Punkt     ');
  Writeln('2. Okrag     ');
  Writeln('3. Linia     ');
  Writeln('4. Trojkat   ');
  Writeln('5. Prostokat ');
  Writeln('6. Kwadrat   ');
  Writeln('7. Latawiec  ');
  Writeln('------------- nacisnij numer i ENTER');
  readLn(wybor);


 dr:=detect;gr:=0;
 InitGraph(dr,gr,''); {zainicjowanie trybu graficznego}

 {inicjowanie wybranej figury - utworzenie obiektu dynamicznego}
 case wybor of
   1 : F := New(P_Point, Init(100, 100));
   2 : F := New(P_Circle, Init(100, 100, 60));
   3 : F := New(P_Line, Init(100, 100, 90, 80));
   4 : F := New(P_Triangle, Init(100, 100, 30, 80, 90, 70));
   5 : F := New(P_Bar, Init(100, 100, 90, 50));
   6 : F := New(P_Square, Init(100, 100, 80));
   7 : F := New(P_Latawiec, Init(100, 100, 30, 50));
 end;

 F^.Move;                          {poruszanie obiektem}
 Dispose(F, Done);                 {likwidacja obiektu dynamicznego}

 CloseGraph;              {wylaczenie trybu graficznego}
END.
```

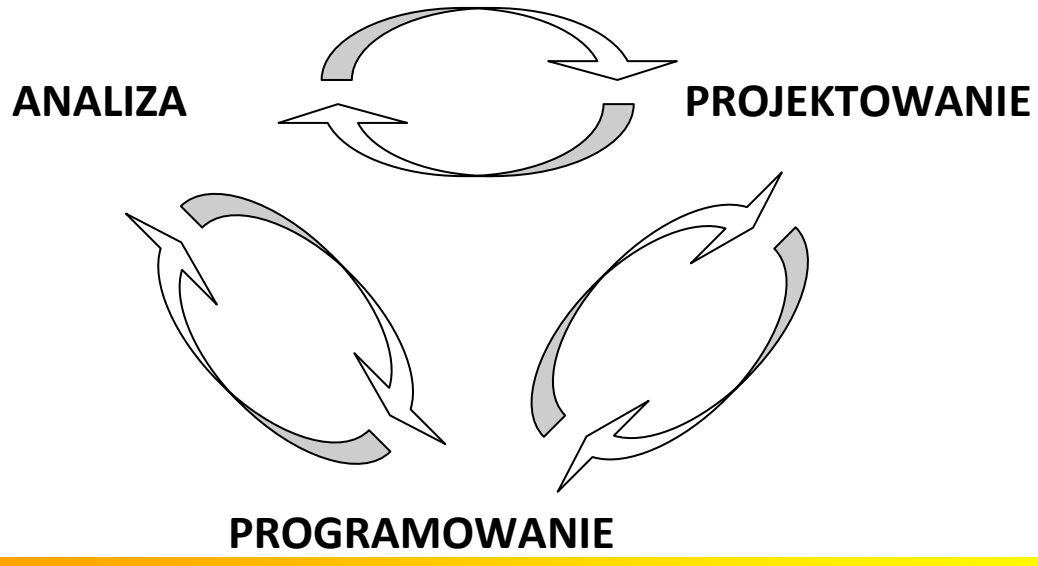
Język C# w praktyce szkolnej

mgr Marian Mysior

Wprowadzenie

Visual Studio .NET integruje wiele środowisk programistycznych w postaci jednolitego interfejsu (*IDE*). Udostępnia środowisko, z którego korzystać mogą różne języki programowania. Microsoft dostosował do platformy *.NET*, od lat znany i popularny, język *Visual Basic*, zmieniając go generalnie. *VB .NET* jest w pełni obiektowym językiem programowania. Dla użytkowników języków rodziny *C* stworzono na platformie *.NET* nowy język *C#* („*C sharp*"). Jest to nowoczesny, obiektowy język programowania, używający wielu konstrukcji języka *C++*, ale bez zbędnych „naleciałości historycznych". Język *Visual C++* również obecny jest na tej platformie. Istnieje przecież wiele wcześniej napisanych, przydatnych bibliotek obiektowych dla niego.

Tworzenie aplikacji można podzielić na trzy zasadnicze fazy: analizę, projektowanie i programowanie (kodowanie). Podział ten ma szczególne znaczenie dydaktyczne. Kłopoty z prawidłowym działaniem aplikacji mają często źródło w powierzchownej bądź błędnej analizie problemu, w złym (nieprzemyślanym) projekcie. Emocjonalne podejście do programowania przez wielu programistów powoduje niedocenianie dwóch pierwszych faz, czego efektem jest program błędny lub gorszy od oczekiwań. Efektywne tworzenie aplikacji powinno składać się nie z jednorazowej analizy, projektowania i programowania, lecz z równoczesnego rozwijania wszystkich tych faz. Aby udoskonalać i poprawiać swój produkt należy wielokrotnie powracać po fazie programowania do ponownej analizy lub do projektowania. Jest to tzw. *model piłki baseballowej*.



Finalnym zapisem algorytmu rozwiązania problemu jest kod źródłowy (skrypt) zapisany w języku programowania. Dalej następuje proces kompilacji i uruchamiania programu. Często używanymi narzędziami są tutaj środowiska zintegrowane w postaci różnych wersji *Pascala* (*Delphi)* lub *C++* (*Builder*). Zamiast nich użyjmy języków platformy *.NET*. Dobrym wyborem może być *Visual C#*. Jedną z zalet środowiska *.NET* jest to, że wybór języka jest rzeczą drugorzędną. Integracja wielu języków w jednym środowisku powoduje, że nie musimy tracić czasu na uczenie się od początku nowych języków, aby tworzyć, uruchamiać i rozwijać aplikacje. *Visual C#* i *Visual Basic* na platformie *.NET* dają nam podobne możliwości. Użycie zmiennych, klas, ich właściwości i metod jest prawie identyczne w obydwu językach. Środowisko *.NET* (tzw. *Framework*) oferuje wspólne typy danych (*Common Type System*) i biblioteki klas (*.NET Class Framework*).

Potrzeby obecnych użytkowników oprogramowania wymagają, aby aplikacje dysponowały graficznymi interfejsami (były „okienkowe”). To z kolei rodzi wymagania względem środowisk programistycznych, które musza być efektywne. Archaicznym podejściem do nauki programowania jest wykorzystywanie narzędzi, które nie zapewniają wspomagania graficznego w tworzeniu interfejsów, nie preferują programowania obsługującego zdarzenia i programowania obiektowego. Te pojęcia nie muszą oznaczać „wyższego wtajemniczenia” w sztukę programowania. Nie należy klasyfikować je, jako problemy zaawansowane. Od nich musimy zaczynać nie tylko programowanie, ale już analizę problemu i projektowanie rozwiązania. Języki platformy *.NET* umożliwiają jak nigdy dotąd tworzenie programów obiektowych, sterowanych zdarzeniami i silnie wspomaganych graficznie. Od samego początku wykorzystujemy tutaj (przynajmniej pasywnie) właściwości i metody obiektów, nawet nie wspominając o nich. W sytuacjach, gdy interfejs graficzny jest zbędny lub przeszkadza w „zgłębieniu istoty problemu” można zbudować aplikację konsolową.

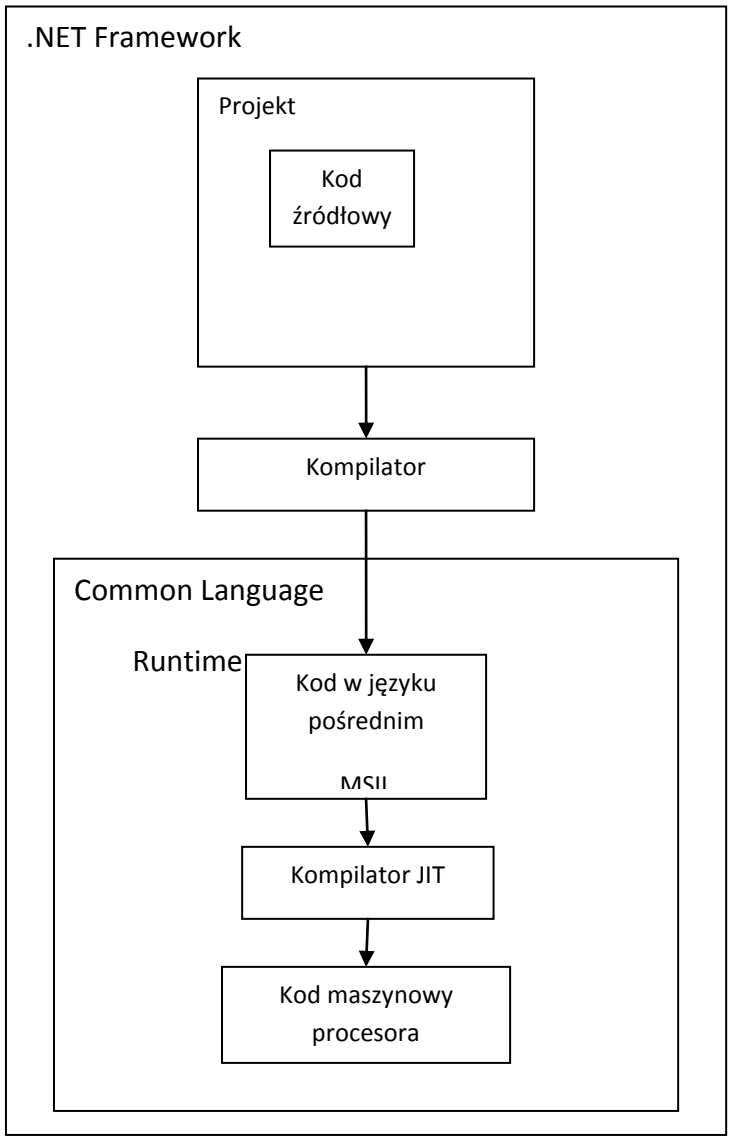
Platforma *.NET* to nie tylko języki programowania. W wielu sytuacjach pełnią one rolę pomocniczą, służąc do zapisywania poleceń. Znając podstawy programowania w *C#* lub *Visual Basic* można budować i zarządzać bazami danych. W *Visual Studio .NET* znajdziemy *Server Explorer*, za pomocą którego możemy tworzyć tabele baz danych i łączyć je relacjami. Zbiór klas *ADO .NET* umożliwia z kolei zarządzanie bazami w tradycyjnym środowisku połączonym jak i w bardziej efektywnym i nowoczesnym środowisku rozłączonym. Do wykonania szeregu zadań, jak zapytania i modyfikacje relacyjnych baz danych wykorzy-

stywać możemy popularny język SQL. Dane zapisywać możemy w coraz bardziej popularnym, niezależnym od platformy formacie *XML*.

Platforma *.NET* oferuje również (może przede wszystkim) narzędzia do budowania aplikacji internetowych i zarządzania zasobami rozproszonymi. Biblioteka *ASP .NET* umożliwia tworzenie formularzy *WWW*. Usługi *Web Services* umożliwiają tworzenie nowoczesnych aplikacji rozproszonych.

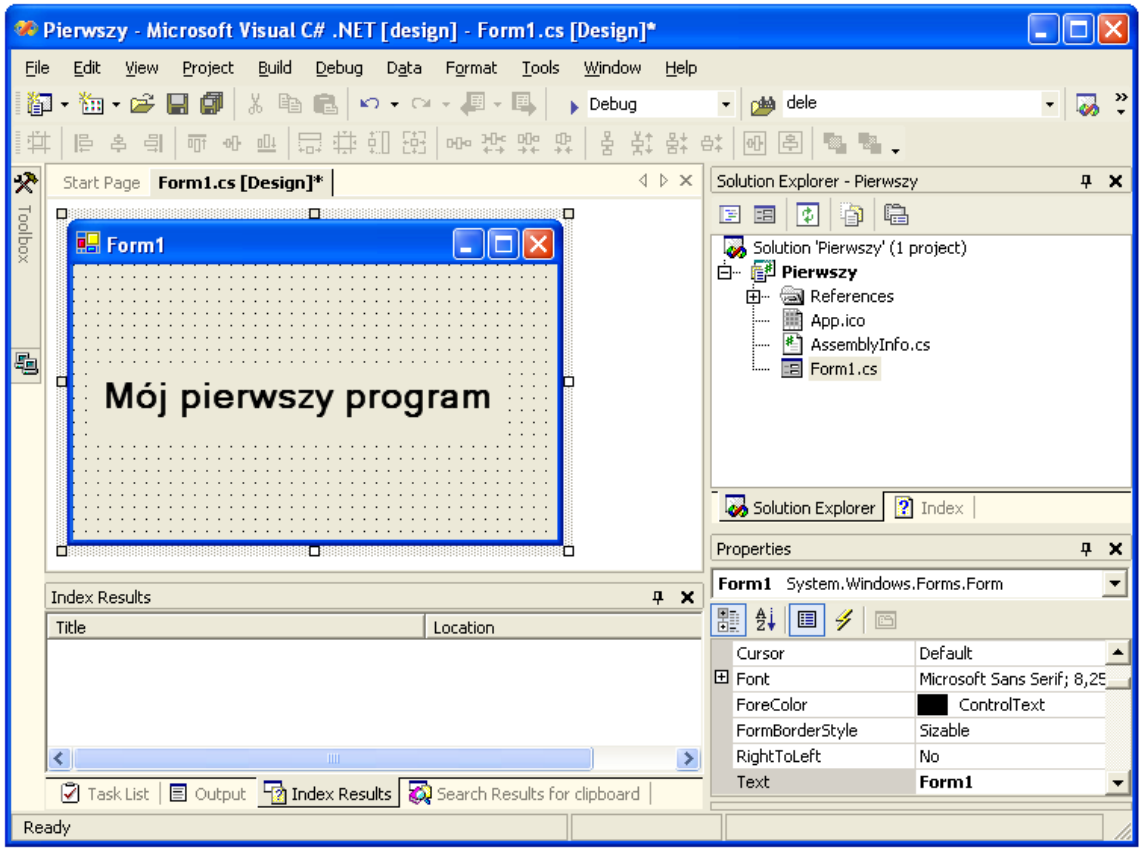
1. Środowisko programowania

Język C# jest narzędziem programistycznym zaprojektowanym do użycia na platformie *.NET Framework* firmy *Microsoft*. Platforma ta stanowi środowisko uruchomieniowe, bibliotekę klas, kompilatory i narzędzia pomocnicze. Programy napisane w środowisku *.NET Framework* nie są kompilowane od razu do kodu maszynowego danego procesora, lecz do kodu pośredniego *MSIL* (*Microsoft Intermediate Language*). Uruchomienie programu wymaga środowiska *CLR* (*Common Language Runtime*), które jest częścią *.NET Framework*. Podczas pierwszego uruchomienia programu na maszynie docelowej kompilator *Just In Time* tworzy kod maszynowy specyficzny dla danego procesora. W uproszczeniu przedstawia to poniższy schemat.

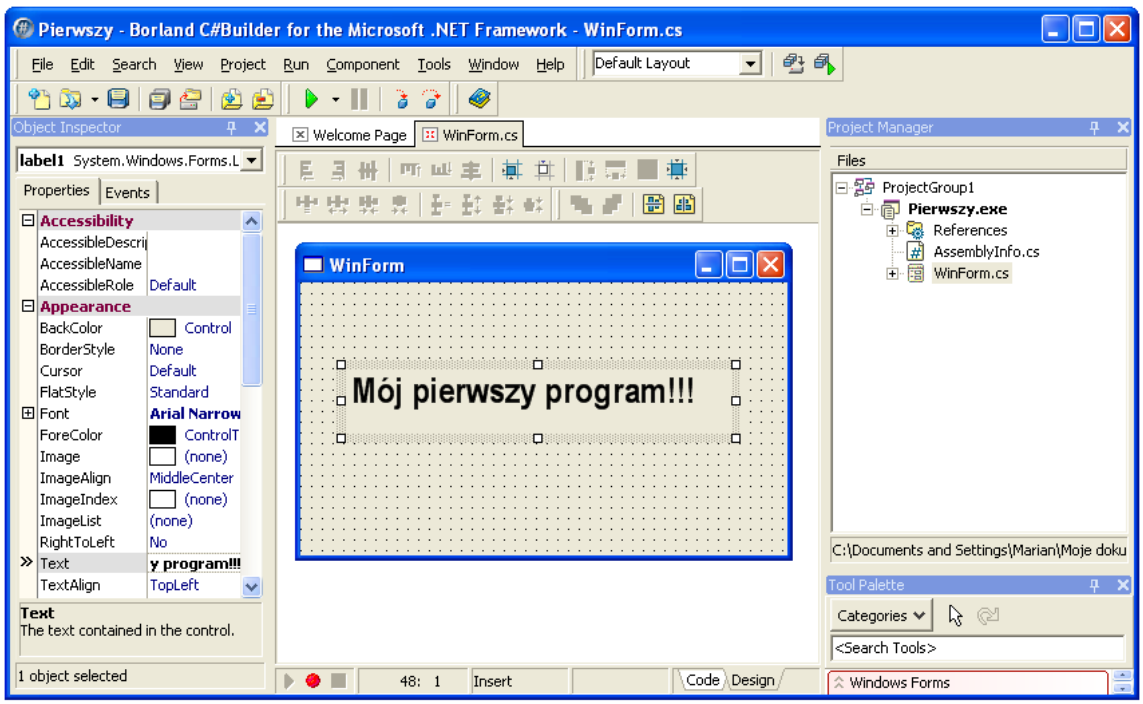


We wszystkich systemach Windows, wcześniejszych niż Windows XP niezbędne jest zainstalowanie pakietu *.NET Framework*, dostępnego bezpłatnie na stronach internetowych Microsoftu. Środowisko to wystarcza jako minimum, do pisania, kompilowania i uruchomiania programów w języku C# (zawiera kompilator wierszowy tego języka – *csc.exe*). Aby jednak efektywnie tworzyć kod źródłowy (korzystać z graficznego wspomagania i wielu innych narzędzi wspierających), kompilować go i wykrywać błędy, warto

zaopatrzyć się w środowiska zintegrowane *Visual Studio .NET* (*Microsoft Visual C# .NET*) lub *Borland C# Builder*.



Rysunek 1. Środowisko zintegrowane *Visual Studio .NET*



Rysunek 2. Środowisko zintegrowane *Borland C# Builder*

Środowiska zintegrowane zwalniają programistę z „ręcznego” pisania całego kodu, oferując graficzne wspomaganie podczas wybierania typu aplikacji i wstawiania poszczególnych komponentów. Znacząco przyspiesza to cały proces tworzenia programu i zwalnia programistę pamięciowego opanowywania wielu szczegółów.

Kod automatyczny programu typu *Windows Application*:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace Pusty
{
	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class Form1 : System.Windows.Forms.Form
	{
		/// <summary>
		/// Required designer variable.
		/// </summary>
		private System.ComponentModel.Container components = null;

		public Form1()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
```

```
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.Size = new System.Drawing.Size(300,300);
            this.Text = "Form1";
        }
        #endregion

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.Run(new Form1());
        }
    }
}
```

Szkielet aplikacji, tworzony automatycznie, może być częściowo niezrozumiały dla początkującego programisty, jednakże szczegółowa jego znajomość nie jest konieczna na tym etapie nauki programowania. Warto zwrócić uwagę na niektóre charakterystyczne miejsca kodu:

`using` – deklaracja użycia przestrzeni nazw. Pozwala uniknąć pisania kwalifikowanych (wieloczłonowych) nazw;

`namespace` – definicja własnej przestrzeni nazw, zapisanej w pliku;

`public class Form1 : System.Windows.Forms.Form` – definicja klasy formularza;

`#region` – dyrektywa początku obszaru (regionu), tutaj automatycznie wypełnianego podczas umieszczania elementów (formantów - kontrolek) na formularzu;

`#endregion` - dyrektywa końca obszaru;

`static void Main()` – specjalna metoda (funkcja), od której rozpoczyna się wykonanie programu;

`Application.Run(new Form1())` – uruchomienie standardowej aplikacji Windows z utworzeniem obiektu formularza;

`//` - komentarz wierszowy (znaki pomijane, do końca wiersza, przez kompilator);

`///` - komentarz XML.

W języku C#, podobnie jak w innych językach rodziny C, wielkość użytych liter ma znaczenie, np. *Main* i *main* nie są tymi samymi nazwami.

Oprócz aplikacji Windows (okienkowych) możemy tworzyć programy działające w oknie konsoli poleceń. Kod tworzony automatycznie jest wtedy prostszy.

```
using System;

namespace Konsola
{
	/// <summary>
	/// Summary description for Class1.
	/// </summary>
	class Class1
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
```

```
                //
                // TODO: Add code to start application here
                //
            }
        }
}
```

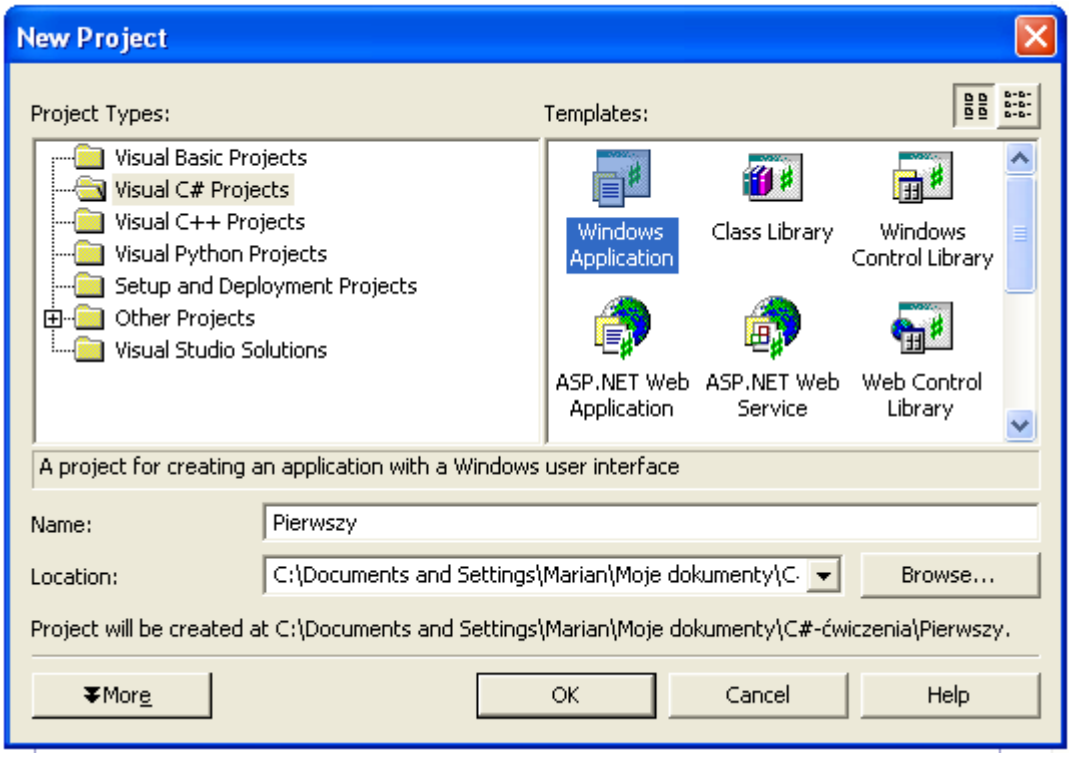
Wygląd takiej aplikacji jest jednak dość ascetyczny. Nie będziemy więc używać jej w naszych przykładach. Aplikacje konsolowe warto wykorzystywać, w przypadkach gdzie nieistotny jest tzw. graficzny interfejs użytkownika.

2. Pierwszy program

Ćwiczenia realizować będziemy wykorzystując środowisko zintegrowane *Visual Studio .NET*.

Ćwiczenie 1. Zaczynamy

Uruchomimy *Visual Studio .NET* i klikniemy przycisk *New Project*.



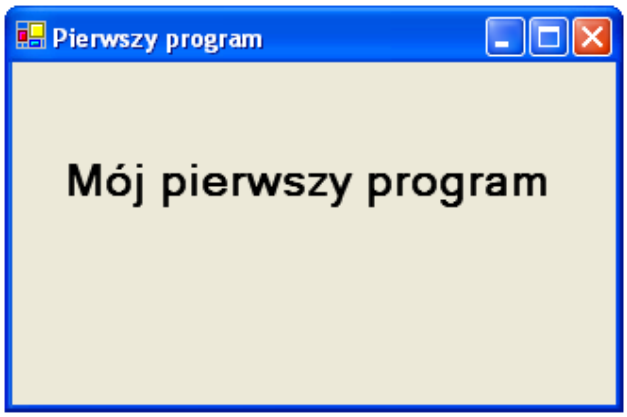
Rysunek 3. Okno dialogowe *New Project*

- W oknie dialogowym *New Project* wybierzemy typ projektu (język programowania) – *Visual C# Project*, szablon (*Templates*) – *Windows Application*, określimy własną nazwę projektu (zostanie automatycznie utworzony katalog o podanej nazwie) i wybierzemy lokalizację na dysku.

Po otwarciu projektu zobaczymy projekt formularza i kilka innych okien (*Rysunek1*). Projekt formularza będzie oknem naszej aplikacji.

- W prawym dolnym narożniku ekranu znajdziemy okno właściwości (*Properties*), w którym wybierzemy właściwość formularza: *Text*. Wpiszemy tam „Pierwszy program". Będzie to tytuł naszego formularza.
- Na formularzu umieścimy napis (etykietę): „Mój pierwszy program". Robimy to wykorzystując przybornik (*Toolbox*), zawierający różne kontrolki (inaczej formanty). Po rozwinięciu przybornika wybierzemy kontrolkę *Label* i naniesiemy ją na formularz (podobnie jak rysuje się elementy graficzne w edytorze graficznym). W oknie właściwości, które automatycznie wyświetla właściwości wybranego formantu. Odszukamy pozycję *Text* i wpiszemy swój tekst.
- Możemy jeszcze zadbać o jego wygląd. Właściwość *Font* pozwala nam wybrać atrybuty czcionki (rodzaj, wielkość, kolor itp.). Po kliknięciu na tej właściwości najlepiej wybrać przycisk […]. Otwiera on okno dialogowe *Czcionka* (podobne jak w edytorach tekstu).

Nasz pierwszy program jest gotowy. Utworzyliśmy go korzystając jedynie z narzędzi graficznych, nie pisząc kodu „ręcznie". Możemy teraz wypróbować nasz produkt, uruchamiając go ze środowiska zintegrowanego. W tym celu wybieramy przycisk *Start* (▶) z paska narzędzi. Alternatywne sposoby to *Debug/Start*, lub klawisz F5, *Debug/Start Without Debugging* lub *Ctrl +F5*. Warto polecić ostatni sposób pomijający śledzenie programu (spowalniające kompilację i uruchomienie). Program można oczywiście uruchamiać spoza środowiska zintegrowanego. Odnajdziemy go w swoim projekcie w odpowiednim podkatalogu.



Rysunek 4. Pierwszy program w działaniu

3. Wprowadzanie danych, formatowanie wyników

Tylko najprostsze programy tworzymy bazują całkowicie na kodzie pisanym automatycznie. Program rozwiązujący nawet nieskomplikowane zadanie matematyczne wymaga napisania odpowiednich formuł, podania danych i wyprowadzenia wyników.

Ćwiczenie 2. Obliczamy objętość kuli

Analiza problemu:

Zastosujemy wzór na obliczanie objętości kuli $v = 4/3\ \pi\ r^3$.

Projekt:

- Objętość kuli zależy od jej promienia, więc na formularzu umieścimy następujące kontrolki:
  - Etykietę (*Label*) z napisem „Podaj promień:”
  - Pole tekstowe (*TextBox*), w którym wymażemy napis domyślny z właściwości *Text*.
- Wstawimy drugą etykietę (usuniemy napis domyślny), do wyprowadzenia wyników.
- Obliczenia zostaną uruchomione po podaniu promienia i naciśnięciu odpowiedniego przycisku. Przycisk (*Buton*) stawimy z przybornika na projekt formularza. Jego właściwość *Text* nadamy wartość „Oblicz”.
- Uruchomimy edycję metody obsługi zdarzenia „przy kliknięciu”, klikając dwukrotnie na projekcie przycisku. W ten sposób zaczynamy ręczne wprowadzanie kodu. Nagłówek tej metody tworzony zostanie automatycznie i nie należy zmieniać go.

Programowanie (pisanie kodu):

```
private void button1_Click(object sender, System.EventArgs e)
{
      double r, v;
      r = Double.Parse(textBox1.Text);
      v = 4.0/3* Math.PI*r*r*r;
      label2.Text = "Objętość kuli wynosi: " + v.ToString("N4");
}
```

Prześledźmy wprowadzony kod.

Słowem kluczowym `double` deklarujemy użycie dwóch zmiennych `r` i `v`.

Język *C#* wymaga deklarowania typu zmiennych (danych) przed ich użyciem.

Tabela prezentuje dopuszczalne typy danych.

Typ C#	Typ .NET Framework	Zakres	Rodzaj – rozmiar
`bool`	`System.Boolean`	true, false	Logiczny, 32 bity
`byte`	`System.Byte`	0 do 255	Bez znaku, liczba całkowita, 8 bitów
`sbyte`	`System.SByte`	-128 do 127	Ze znakiem, liczba całkowita, 8 bitów
`char`	`System.Char`	U+0000 do U+ffff	Znaki Unicode, 16 bitów
`decimal`	`System.Decimal`	1.0×10^{-28} do 7.9×10^{28} Precyzja: 28-29 cyfr znaczących	Liczba stałopozycyjna (rzeczywista), 96 bitów
`double`	`System.Double`	$\pm 5.0 \times 10^{-324}$ do $\pm 1.7 \times 10^{308}$ Precyzja: 15-16 cyfr	Liczba zmiennopozycyjna (rzeczywista), 64 bity
`float`	`System.Single`	$\pm 1.5 \times 10^{-45}$ do $\pm 3.4 \times 10^{38}$ Precyzja: 7 cyfr	Liczba zmiennopozycyjna (rzeczywista), 32 bity
`int`	`System.Int32`	-2,147,483,648 do 2,147,483,647	Ze znakiem, liczba całkowita, 32 bity
`uint`	`System.UInt32`	0 do 4,294,967,295	Bez znaku, liczba całkowita, 32 bity
`long`	`System.Int64`	-9,223,372,036,854,775,808 do	Ze znakiem, liczba całkowita,

		9,223,372,036,854,775,807	64 bity
ulong	System.UInt64	0 do 18,446,744,073,709,551,615	Bez znaku, liczba całkowita, 64 bity
object	System.Object		Odwołanie do obiektu klasy, 32 bity
short	System.Int16	-32,768 do 32,767	Ze znakiem, liczba całkowita, 16 bitów
ushort	System.UInt16	0 do 65,535	Bez znaku, liczba całkowita, 16 bitów
string	System.String	Do 2 miliardów znaków	Łańcuch znakowy w kodzie Unicode

Platforma .NET Framework wymaga używania zunifikowanych typów danych. Zastosowanie typu `double` jest równoznaczne z użyciem `System.Double`.

Dane wprowadzane do okna edycyjnego `TextBox` są zawsze łańcuchem znaków (właściwość `Text`). Należy przekształcić je w liczbę właściwego typu. Wykonujemy to używając metody `Parse`:

```
r = Double.Parse(textBox1.Text);
```

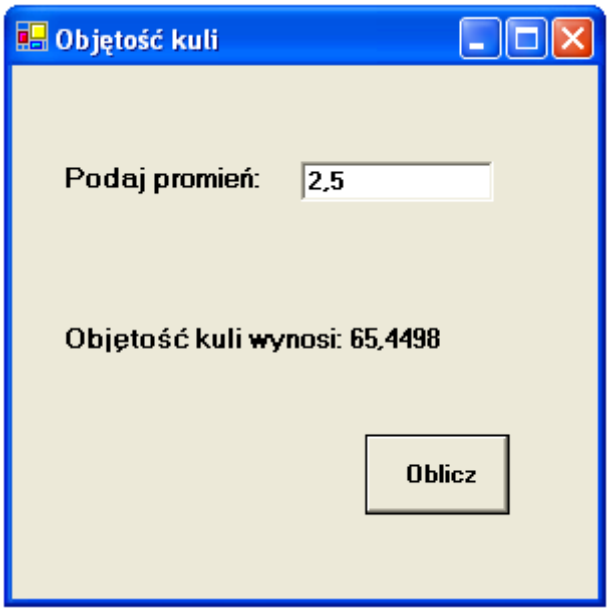
Obliczając objętość należy zwrócić uwagę, że operator "/" oznacza w *C#* dzielenie całkowitoliczbowe (pomija część ułamkową), jeżeli licznik i mianownik są liczbami całkowitymi. W naszym przypadku należy, zamiast dzielenia „4/3" - napisać 4.0/3 (4.0 nie jest już zapisem liczby całkowitej) lub użyć przyrostka „d" (jak double): „4d/3".

Liczba π jest wartością stałą umieszczoną w klasie `Math`. Dostęp do niej uzyskamy pisząc `Math.PI`.

Dane wyprowadzamy do etykiety (`Label`) na formularzu. Właściwość `Text` etykiety wymaga podania łańcucha znaków. Wartość `v` musimy więc skonwertować do postaci łańcucha. Zrobimy to używając metody `ToString("N4")`. Argument w nawiasie podaje format liczby i ilość miejsc po

przecinku dziesiętnym. Ponieważ chcemy podać tekst objaśniający wyprowadzaną wartość, wpisujemy łańcuch informacyjny i dołączamy do niego `v.ToString("N4")`:

```
label2.Text = "Objętość kuli wynosi: " + v.ToString("N4");
```



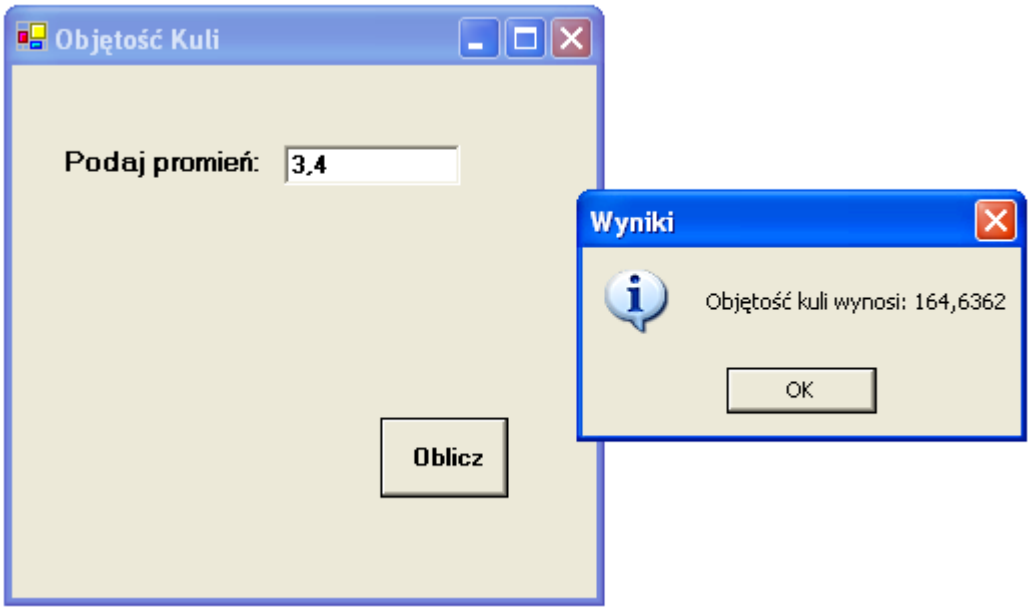
Rysunek 5. Kula

Ćwiczenie 3. Inny sposób wyprowadzenia wyników

Wyniki i komunikaty możemy wyprowadzić do standardowego okna `MessageBox`.

```
private void button1_Click(object sender, System.EventArgs e)
{
      double r, v;
      r = Double.Parse(textBox1.Text);
      v = 4.0/3* Math.PI*r*r*r;
      MessageBox.Show("Objętość kuli wynosi: " + v.ToString("N4"),
        "Wyniki", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Metoda `Show` klasy `MessageBox` służy do wyprowadzania różnego rodzaju komunikatów w formie standardowego okna.



Rysunek 6. MessageBox

Metoda Show może być używana z różną ilością argumentów. Jej bardziej rozbudowana postać to:

```
MessageBox.Show(komunikat, tytuł, MessageBoxButtons, MessageBoxIcon,
   MessageBoxDefaultButton);
```

Gdzie:

`komunikat` – komunikat umieszczany w oknie,

`tytuł` – tytuł okna,

`MessageBoxButtons` – rodzaje przycisków,

`MessageBoxIcon` – rodzaje ikon (znaków graficznych w oknie),

`MessageBoxDefaultButton` – domyślny przycisk.

Trzy ostatnie argumenty są wyliczeniami. W C# wyliczenia używamy łącznie z nazwą typu, np. `MessageBoxIcon.Information`.

`MessageBoxButtons` określa ilość i rodzaje przycisków. Może przyjmować następujące wartości:

Nazwa	Opis
OK.	Wyświetl tylko przycisk *OK*.
OKCancel	Wyświetl przyciski *OK* i *Anuluj*
AbortRetryIgnore	Wyświetl przyciski *Przerwij, Ponów i Ignoruj*
YesNoCancel	Wyświetl przyciski *Tak, Nie i Anuluj*.
YesNo	Wyświetl przyciski Ta*k i Nie*.
RetryCancel	Wyświetl przyciski *Ponów i Anuluj*.

Wartości `MessageBoxIcon`:

Nazwa	Opis
Error, Hand lub Stop	Wyświetl ikonę *Komunikat krytyczny*.
Question	Wyświetl ikonę *Pytanie ostrzegawcze*.
Exclamation lub Warning	Wyświetl ikonę *Komunikat ostrzegawczy*.
Information lub Asterisk	Wyświetl ikonę *Komunikat informacyjny*
None	Brak ikony

Wartości `MessageBoxDefaultButton`:

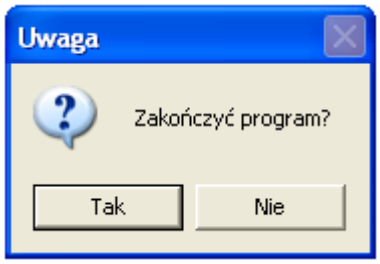
Nazwa	Opis
Button1	Domyślnym przyciskiem jest przycisk pierwszy.
Button2	Domyślnym przyciskiem jest przycisk drugi.
Button3	Domyślnym przyciskiem jest przycisk trzeci.

Metoda `Show` zwraca określone wartości, zależne od wciśniętego przycisku. Można to wykorzystać do zadania pytania w oknie `MessageBox`.

Ćwiczenie 4. Pytanie w oknie `MessageBox`

Formularz z poprzedniego ćwiczenia uzupełnimy o przycisk „Zamknij”. Oprogramujemy go następująco:

```
private void button2_Click(object sender, System.EventArgse)
{
      if (MessageBox.Show("Zakończyć program?", "Uwaga",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultBut-
            ton.Button1)==DialogResult.Yes)
          this.Close();
}
```



Rysunek 7. MessageBox Tak/Nie

Pełna lista zwracanych wartości `DialogResult`:

Stan	Opis
OK.	OK
Cancel	Anuluj
Abort	Przerwij
Retry	Ponów
Ignore	Ignoruj
Yes	Tak
No	Nie
None	Nic nie zwraca

Laptop na bank

mgr Janusz Podolak, mgr inż. Wojciech Błaszczyk

Konkurs „Laptop na Bank", organizowany przez Wojewódzki Ośrodek Doskonalenia Informatycznego i Politechnicznego w Opolu w Opolu i Oddział Okręgowy Narodowego Banku Polskiego w Opolu, dobiegł końca.

Laureatami konkursu zostali:

Pierwsze miejsce:	**Piotr Kulessa**	ZSO Kluczbork	opiekun Elżbieta Sowa
Drugie miejsce:	**Rafał Hirsz**	ZSO Strzelce Opolskie	opiekun Paweł Pagacz
Trzecie miejsce:	**Kamil Lentner**	ZS Praszka	opiekun Marek Morawiec

Zwycięscy otrzymali nagrody ufundowane przez Narodowy Bank Polski:

- **Notebook FS V6535**
- **Palmtop Mio 360**
- **Cyfrowy odtwarzacz multimedialny MP3/MP4**

Opiekun zwycięscy otrzymał **Palmtop Asus A696**.

Konkurs skierowany był do uczniów szkół gimnazjalnych i ponadgimnazjalnych województwa opolskiego i spotkał się z ogromnym zainteresowaniem. Zgłoszonych zostało 750 uczniów spośród, których zostało wyłoniony 17 finalistów.

Pierwszy etap Konkursu został przeprowadzony w dniach 9-10 grudnia 2008. Uczestnicy rozwiązywali test wielokrotnego wyboru, umieszczony na platformie e-learnigowej WODIiP. Klucz dostępu do platformy otrzymali opiekunowie od organizatorów. Test był udostępniony do rozwiązywania od godziny 8:00 9 grudnia do godziny 16:00 10 grudnia, każdy z uczestników mógł do niego podejść jednokrotnie. W składał testu weszło 30 pytań o tematyce ekonomicznej i informatycznej przygotowanych przez organizatorów. Zarówno kolejność pytań jak i umieszczonych w nich odpowiedzi była losowa.

Część pierwsza etapu finałowego miała podobny charakter ale odbyła się już w pracowni Wojewódzkiego Ośrodka Doskonalenia Informatycznego i Politechnicznego w Opolu.

Do ścisłego finału przeszły trzy osoby, które uzyskały największą ilość punktów w części pierwszej. Ścisły finał odbył się z udziałem publiczności. Zawodnicy odpowiadali na pięć pytań, które miały ustalić kolejność finalistów. Publiczność miało możliwość obserwowania zmagań laureatów oraz obserwować pytania i wyniki, które były wyświetlane na tablicach interaktywnych. Po tej rundzie okazało się, że dwóch pierwszych zawodników ma tę samą ilość punktów. Zwycięzca został wyłoniony dopiero w wyniku dogrywki.

Sądzimy, że cele konkursu a w szczególności: rozwijanie zainteresowania uczniów edukacją ekonomiczną, przedsiębiorczością w połączeniu z technologią informacyjną oraz

poszukiwanie utalentowanej młodzieży i zapewnienie możliwości rywalizacji w wybranej dziedzinie wiedzy zostały spełnione. Zarówno uczestnicy jak opiekunowie pozytywnie wypowiadali się na temat idei i organizacji konkursu. Część opiekunów chce nawiązać z nami kontakt w celu pomocy w organizacji podobnych form na terenie swoich szkół. Platforma e-learnigowa okazała się tu doskonałym narzędziem.

Nasz konkurs był szeroko komentowany przez media.

Gazeta Wyborcza – Laptop na bank

Gazeta Wyborcza - Laptop nagrodą w konkursach ekonomicznych

Radio Opole - Konkurs "Laptop na bank" rozstrzygnięty

Nowa Trybuna Opolska - Uczniowie powalczą o laptopa

Nowa Trybuna Opolska - Laptop na bank - zwycięzcy konkursu NBP

WODIiP - Laureaci konkursu "Laptop na Bank"

W opinii uczestników, opiekunów i obserwatorów Konkurs był atrakcyjną propozycją dla uczniów, wywołał duże zaangażowanie młodzieży. Dodatkowo interaktywna forma przeprowadzenia finału wprowadziła gorące emocje wśród zebranych.

Pełne wersje programów

mgr Roland Zimek

Zbliża się koniec roku – czas porządków przedświątecznych. Proponuję zrobienie ich nie tylko w mieszkaniu, ale także na dysku twardym. Niestety zdarzy się czasami przypadkowe skasowanie ważnego pliku. Jeżeli został on także usunięty z kosza to odzyskanie go bez specjalistycznego programu będzie niemożliwe. Proponuję wydanie *PC Format 10/2008*, w którym znajdziemy program **EASEUS Photo Recovery 2.1.1**. Umożliwi on odzyskanie przypadkowo usuniętych zdjęć, filmów, muzyki i innych plików. Gdybyśmy przypadkowo stracili zdjęcia na karcie pamięci z aparatu cyfrowego, to z pomocą przyjdzie nam także **UndeleteMyFiles Pro 2.8** znajdujący się w wydaniu *11/2008 Komputer Świat*. Gdy uda nam się już odzyskać wszystkie zdjęcia, to **Auto Movie Creator v1.51** z tego samego wydania miesięcznika, przygotuje prezentacje ze zdjęć i filmów.

W listopadzie ukazał się pakiet 3 wydań miesięcznika *Komputer Świat*. W okazyjnej cenie otrzymujemy numery *8/2008*, *9/2008* i *10/2008* . Wśród ciekawych programów znajdziemy między innymi **Moje Gimnazjum Testy Humanistyczne** oraz **Moje Gimnazjum Testy Matematyczno-Przyrodnicze**. Te dwa programy okażą się dużą pomocą dla uczniów kończących w tym roku naukę w Gimnazjach.

Przygotowanie statycznych zrzutów ekranów nie nastręcza kłopotów. Problemy zaczynają się dopiero wtedy, gdy zamierzamy przechwycić ciąg poleceń wykonywanych na ekranie i zapisać je w postaci wideo. Wyręczy nas w tym program **BB Flashback Express 2** umieszczony w numerze *10/2008 CHIP*.

Wydanie *11/2008 PC World Komputer* zawiera program **PC Security Explorer 2** pozwalający zabezpieczyć nasz komputer i monitorować wiele procesów działający w tle. A dla miłośników tuningu systemu operacyjnego umieszczono na płycie Virtual Kit, będący zestawem funkcji optymalizujących wygląd naszego pulpitu i nie tylko.

NEXT 11/2008 zachęca nas między innymi bardzo dobrym programem **UltraISO 8.6.6** do kompleksowego zarządzania danymi zapisanymi w obrazach dysków.

PC Format 12/2008 kusi nas z kolei programem **Zoner Photo Studio 10 Xpress**. Przy jego pomocy łatwo posegregujemy zdjęcia z wakacji, a także dokonamy kompleksowej obróbki zdjęć, m.in. usuwając czerwone oczy, ustawimy poprawną jasność, kontrast czy nasycenie kolorów.

W wydaniu *12/2008 CHIP*, znajdziemy **Ashampoo Winoptimizer 2009**. Bardzo prosty i przyjemy program dla niezaawansowanych użytkowników komputera, umożliwiający kompleksową optymalizację systemu Windows i porządkowanie danych na dysku twardym.

Na nowy rok, *CHIP* w wydaniu *1/2009* proponuje między innymi program **Paragon Drive Copy 9.0 Specjal Edition** umożliwiający wykonanie obrazu wybranej partycji lub dysku. Odzyskanie tak zachowanego systemu z zainstalowanymi aplikacjami i danymi jest nieporównywanie szybsze i wygodniejsze niż ponowna ich instalacja.