

PODSTAWY PROGRAMOWANIA

Andrzej Marciniak

Politechnika Poznańska, Instytut Informatyki

Prezentacja multimedialna przygotowana za pomocą systemu Corel® Presentations X3
Copyright © 2003-2024 by Andrzej Marciniak

PODSTAWY PROGRAMOWANIA (komputerów w języku Delphi)

Andrzej Marciniak

Politechnika Poznańska, Instytut Informatyki

Prezentacja multimedialna przygotowana za pomocą systemu Corel® Presentations X3
Copyright © 2003-2024 by Andrzej Marciniak

Organizacja przedmiotu

Studia stacjonarne:

24 godziny wykładów
30 godzin laboratoriów

Wykłady w sali 6 CW, laboratoria w sali 144 CW.

Studia niestacjonarne:

12 godziny wykładów
20 godzin laboratoriów

Wykłady w sali L128 BT, laboratoria w sali 144 CW.

Plan wykładów

- **wstęp** (podstawowe pojęcia związane z programowaniem, przegląd języków programowania, sieci działań, ogólne zasady programowania zorientowanego obiektowo)
- **historia języka Delphi i ogólna charakterystyka pakietu Embarcadero Delphi**
- **posługiwanie się zintegrowanym systemem programowania**
- **przegląd konstrukcji języka Delphi** (program, moduł, biblioteka, funkcje i procedury, klasy i obiekty, typy danych, zmienne, instrukcje)
- **struktura programu i modułu,**
- **podstawowe elementy języka** (symbole podstawowe, słowa kluczowe i dyrektywy języka, identyfikatory, liczby, łańcuchy, literały logiczne, komentarze i separatory)

Plan wykładów

- **typy danych i ich opis** (definiowanie typów, typy proste, łańcuchowe, strukturalne, opisujące obiekty, wskaźnikowe, proceduralne, wariantowe, zagnieżdżone definicje typów, zgodność typów)
- **zmienne** (deklaracje zmiennych, zmienne ideksowane, rekordowe, obiektowe, dynamiczne, proceduralne, wariantowe, z początkową wartością, nakładanie zmiennych, literały stałe i zmienne)
- **wyrażenia** (rodzaje operatorów i ich priorytet, składnia wyrażenia, wyrażenia stałe, przeciążanie operatorów)
- **instrukcje** (proste, strukturalne, assemblerowa)
- **funkcje i procedury** (definicje funkcji i procedur, rodzaje parametrów, przeciążanie funkcji i procedur, wywoływanie funkcji i procedur, konwencje wywoływania)

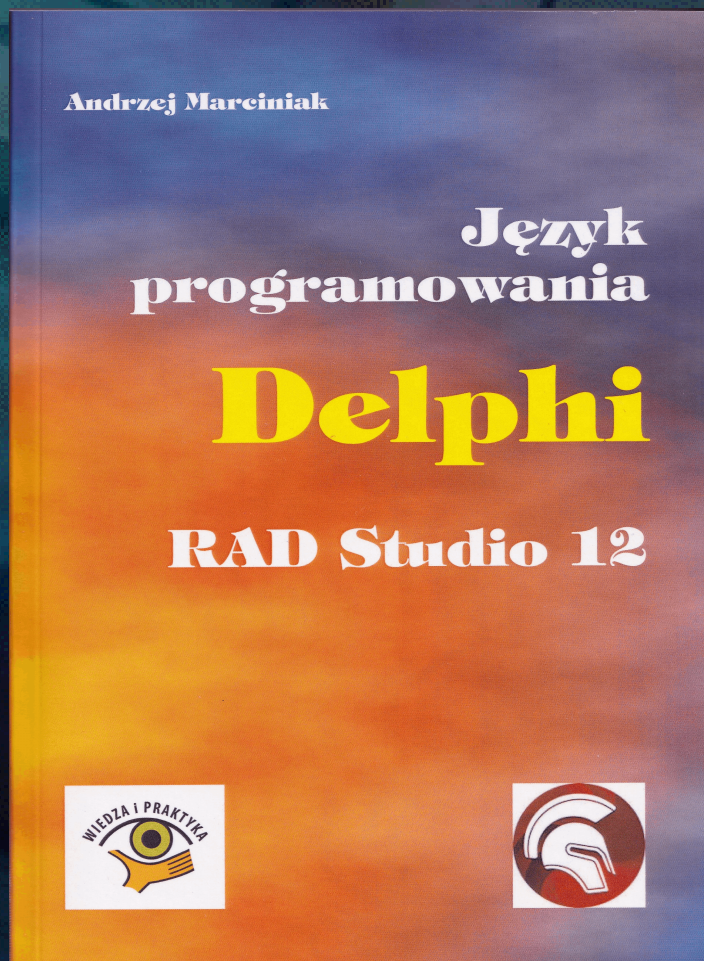
Plan wykładów

- przetwarzanie obiektów (konstruktory i destruktory, metody statyczne, wirtualne, dynamiczne i abstrakcyjne, obsługa wiadomości, własności)
- biblioteki łączone dynamicznie (tworzenie bibliotek, statyczne i dynamiczne pobieranie funkcji i procedur)
- pakiety
- przetwarzanie plików
- typy sparametryzowane
- podstawowe moduły biblioteki RTL (*System*, *System.SysUtils*, *System.Math*, wywołanie innego programu z programu paskalowego) i informacje o bibliotece VCL (m. in. okienka z komunikatami)

Plan wykładów

- przygotowanie wersji dystrybucyjnej programu
- tworzenie komponentów VCL i ich wykorzystanie
- temat dodatkowy 1: wielowątkowość (synchronizacja wątków, priorytety, oczekiwanie na zakończenie)
- temat dodatkowy 2: sprawdzanie obecności poprzedniego egzemplarza programu

Literatura



podręcznik podstawowy

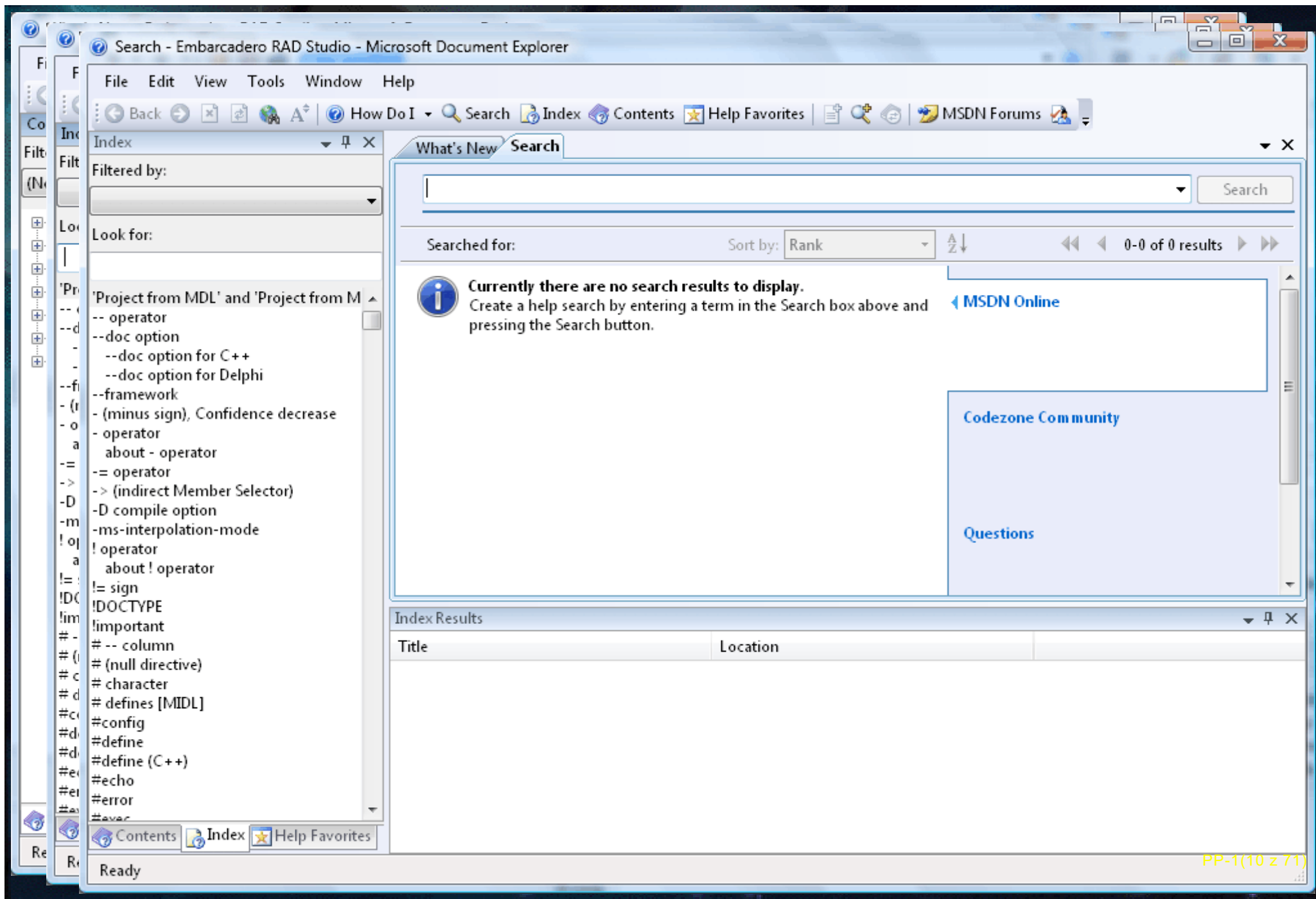
- znajduje się w Bibliotece Politechniki Poznańskiej
- do nabycia przez Internet
- szczegóły o książce:
www.cs.put.poznan.pl/amarciniak/Delphi12.html

Literatura



Literatura

- Andrzej Marciniak, *Język programowania Delphi – RAD Studio 12*, Wydawnictwo WIEDZA i PRAKTYKA, Warszawa 2024



Literatura

- Andrzej Marciniak, *Język programowania Delphi – RAD Studio 12*, Wydawnictwo WIEDZA i PRAKTYKA, Warszawa 2024
- System pomocy pakietu Embarcadero Delphi Professional

Literatura

Napisana przez dwukrotnych laureatów nagrody za najlepszą książkę o Delphi, przyznanej przez czytelników Delphi Informant.

„Steve i Xavier zrobili to znowu! Dali wyraz swemu kunsztownemu rzemiosłu, tworząc przewodnik pozwalający odkryć głębię i poczuć świeży oddech programowania w Delphi 6.”

David Intersimone (David I)
Vice President, Developer Relation
Borland

TOM 1



DELPHI 6

Vademecum profesjonalisty



Zawiera CD

Xavier Pacheco
Steve Teixeira

SAMS
PUBLISHING

Helion



Napisana przez dwukrotnych laureatów nagrody za najlepszą książkę o Delphi, przyznanej przez czytelników Delphi Informant.

„Steve i Xavier zrobili to znowu! Dali wyraz swemu kunsztownemu rzemiosłu, tworząc przewodnik pozwalający odkryć głębię i poczuć świeży oddech programowania w Delphi 6.”

David Intersimone (David I)
Vice President, Developer Relation
Borland

TOM 2



DELPHI 6

Vademecum profesjonalisty



Zawiera CD

Xavier Pacheco
Steve Teixeira

SAMS
PUBLISHING

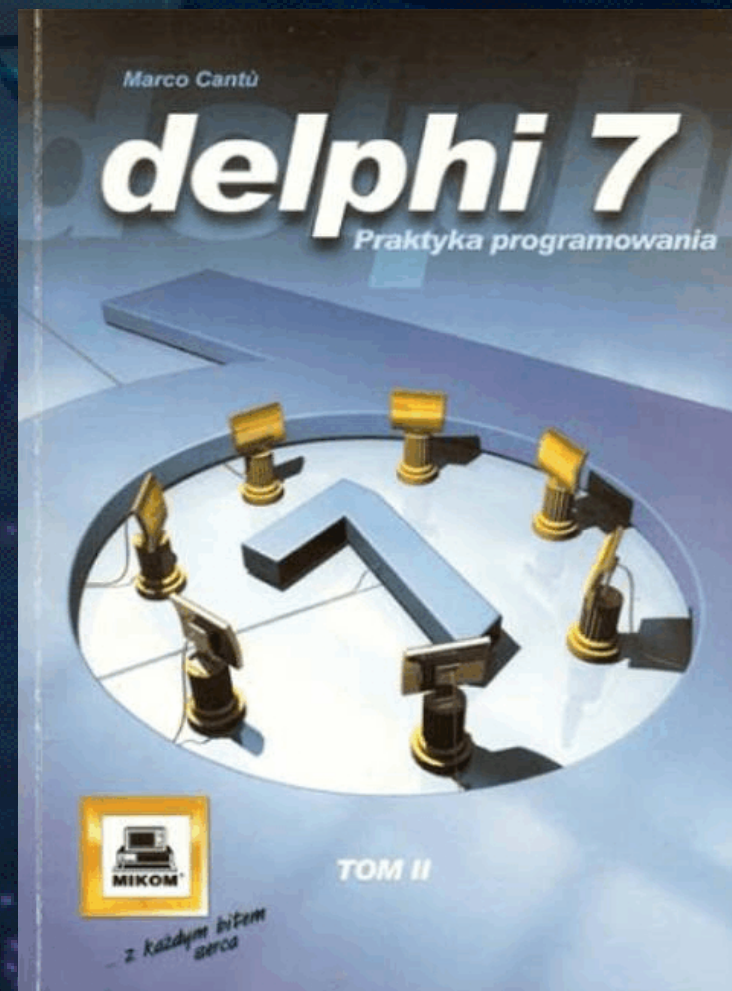
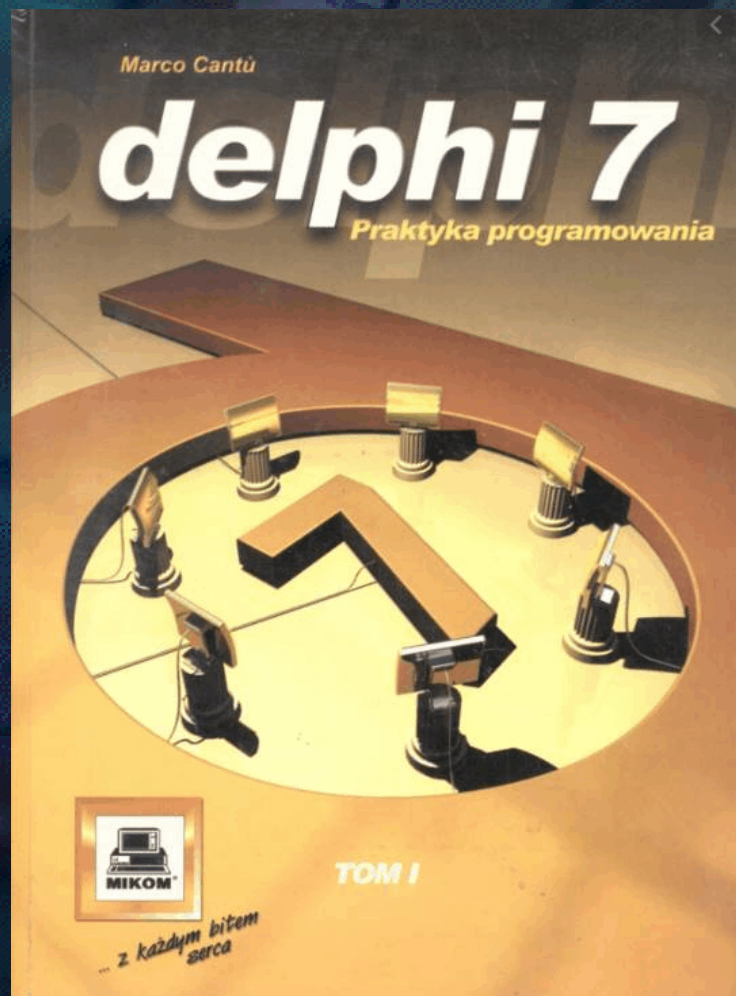
Helion



Literatura

- Andrzej Marciniak, *Język programowania Delphi – RAD Studio 12*, Wydawnictwo WIEDZA i PRAKTYKA, Warszawa 2024
- System pomocy pakietu Embarcadero Delphi Professional
- Xavier Pacheco, Steve Teixeira, *Delphi 6 (5) -Vademecum profesjonalisty*, tom 1, Wydawnictwo HELION, Gliwice

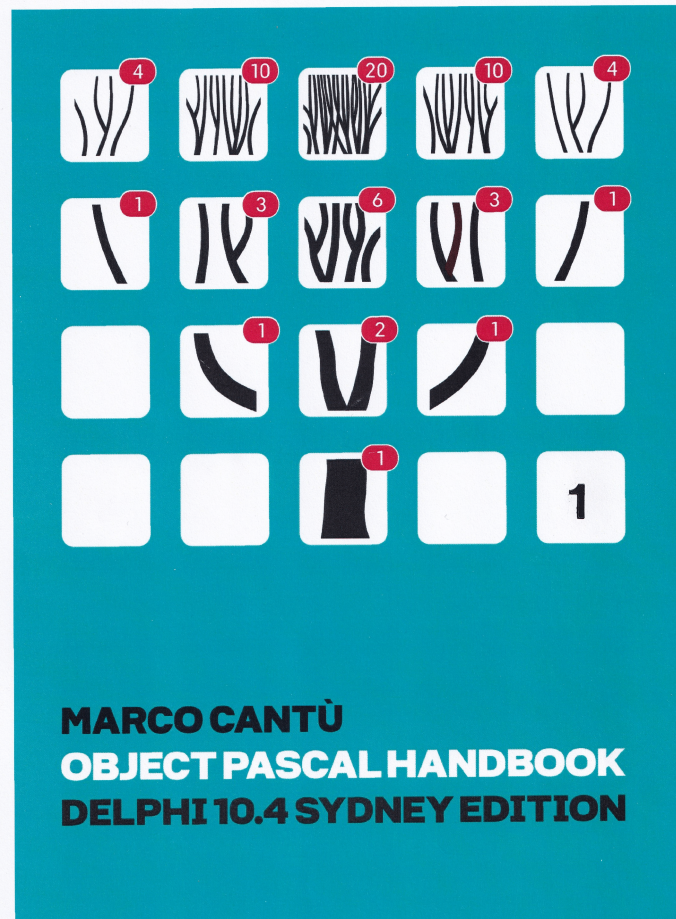
Literatura



Literatura

- Andrzej Marciniak, *Język programowania Delphi – RAD Studio 12*, Wydawnictwo WIEDZA i PRAKTYKA, Warszawa 2024
- System pomocy pakietu Embarcadero Delphi Professional
- Xavier Pacheco, Steve Teixeira, *Delphi 6 (5) -Vademecum profesjonalisty*, tom 1, Wydawnictwo HELION, Gliwice
- Marco Cantù, *Delphi 7 (6, 5) - praktyka programowania*, tom 1 i 2, Wydawnictwo MIKOM, Warszawa

Literatura



Literatura

- Andrzej Marciniak, *Język programowania Delphi – RAD Studio 12*, Wydawnictwo WIEDZA i PRAKTYKA, Warszawa 2024
- System pomocy pakietu Embarcadero Delphi Professional
- Xavier Pacheco, Steve Teixeira, *Delphi 6 (5) -Vademecum profesjonalisty*, tom 1, Wydawnictwo HELION, Gliwice
- Marco Cantù, *Delphi 7 (6, 5) - praktyka programowania*, tom 1 i 2, Wydawnictwo MIKOM, Warszawa
- Marco Cantù, *Object Pasdcal Handbook, Delphi 10.4 Sydney Edition*, <http://www.marcocantu.com/objectpascal>

Zaliczenie przedmiotu

- **laboratoria:** samodzielne wykonanie pięciu (czterech na studiach niestacjonarnych) zadań o zróżnicowanej tematyce i stopniu zaawansowania + sprawdzian (nie dotyczy studiów niestacjonarnych)
- **wykłady:** kolokwium zaliczeniowe

Podstawowe pojęcia związane z programowaniem

Programowanie - projektowanie, pisanie, modyfikowanie i testowanie programów.

(IBM Dictionary of Computing)

Dokładniej: jednoznaczne formułowanie zadań i sposobów ich rozwiązywania dla maszyny cyfrowej. Obejmuje następujące etapy:

- sformułowanie problemu,
- zbudowanie modelu logiczno-matematycznego,
- określenie metody rozwiązania (w tym warunków istnienia i jednoznaczności),
- opracowanie numerycznej metody rozwiązywania (jeśli konieczne),
- zbudowanie algorytmu obliczeń (schematu blokowego),
- zakodowanie algorytmu,
- wykonanie obliczeń (testów) na m. c.

Podstawowe pojęcia związane z programowaniem

Algorytm - uporządkowany zbiór reguł rozwiązania zadania w skończonej liczbie kroków.

(IBM Dictionary of Computing)

Program - algorytm zapisany w języku programowania.

Podstawowe pojęcia związane z programowaniem

Język programowania - język formalny (sztuczny) służący do zapisu programów komputerowych.

(IBM Dictionary of Computing)

Dokładniej: język formalny zapewniający komunikację człowieka z komputerem i przeznaczony do opisu danych oraz algorytmów ich przetwarzania w komputerze.

Każdy język programowania jest określony poprzez swoją syntaktykę oraz semantykę, tj. zbiór reguł określający, jakie rodzaje wyrażeń można wykorzystać w pisaniu programów i jakie jest ich znaczenie funkcjonalne.

Język programowania jest językiem sztucznym, opierającym się na języku etnicznym (np. angielskim) lub na wyrazach sztucznych o brzmieniu łatwym do zapamiętania.

Podstawowe pojęcia związane z programowaniem

Język ukierunkowany maszynowo (język niskiego poziomu; ang. *low-level language, computer-oriented language*) - język programowania zbliżony do języka wewnętrznego komputera, w którym jedna instrukcja odpowiada na ogół jednemu rozkazowi komputera.

Rozkaz - sekwencja określonych znaków, inicjująca i określająca w sposób częściowy lub całkowity przebieg operacji wykonywanej przez komputer. Rozkaz jest podstawowym elementem programu i składa się z części operacyjnej i adresowej.

Podstawowe pojęcia związane z programowaniem

Język wyższego rzędu (język wysokiego poziomu; ang. *high-level language*) - język programowania, którego pojęcia i struktury są dogodne dla człowieka (w procesie myślenia, dla zrozumienia algorytmu zapisanego w języku).

(IBM Dictionary of Computing)

Podstawowe pojęcia związane z programowaniem

Język uniwersalny - język programowania, który może być użyty do różnych zastosowań przetwarzania danych. Pierwszym językiem uniwersalnym był język PL/I (opracowany dla komputerów IBM serii 360).

Język specjalizowany - język przeznaczony do formułowania określonych zastosowań przetwarzania danych.

Przegląd języków programowania

ADA

Język wyższego rzędu, uniwersalny, zorientowany proceduralnie, opracowany i rozwijany pod auspicjami Departamentu Obrony USA. Zawiera programowanie strukturalne, środki do definiowania struktur danych z określaniem ich typów, środki do programowania zorientowanego obiektowo oraz umożliwia wielozadaniowość.

Augusta **Ada** Byron (1815-1852) - matematyczka i koleżanka Charlesa Babbage'a, który skonstruował pierwszy programowalny kalkulator. Niektóre zapiski Ady Byron dotyczące programowania tego kalkulatora przetrwały do czasów współczesnych, co spowodowało uznanie jej za pierwszą programistkę na świecie.

```
with Text_IO;
procedure Convert is
  package Int_IO is new
    Text_IO.Integer_IO(Integer);
  Fahr : Integer;
begin
  Text_IO.Put_Line("Enter fahrenheit");
  Int_IO.Get(Fahr);
  Text_IO.Put("Centigrade is ");
  Int_IO.Put((Fahr-32)*5/9);
  Text_IO.New_Line;
end Convert;
```

Przegląd języków programowania

Algol (Algorithmic Language)

Język wyższego rzędu, opracowany głównie w celu przedstawiania algorytmów i ich wymiany pomiędzy ludźmi, a następnie przystosowany do programowania maszyn cyfrowych (Algol 60, Algol 68). Powstał na początku lat 60-tych ubiegłego stulecia i szybko rozpowszechnił się w Europie (w USA mniej znany).

```
begin
  integer Fahr;
  Write ("Enter fahrenheit");
  Read (Fahr);
  Write ("Centigrade is ", (Fahr-32)*5/9)
end;
```

Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

minikomputer Hewlett-Packard HP 3000:

```
begin
  intrinsic read, print, binary, ascii;
  array buffer(0:17);
  array string(0:3);
  byte array b'string(*)=string;
  integer fahr, cent, len;
  move buffer:= "Enter fahrenheit ";
  print (buffer, -30, %320);
  len:=read (string, -4);
  fahr:= binary(b'string, len);
  cent:= (fahr-32)*5/9;
  len:=ascii(cent, 10, b'string);
```

Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

minikomputer Hewlett-Packard HP 3000:

```
move buffer:= "Centigrade is ";  
move buffer(14):= string, (-len);  
print (buffer,-32,%0);  
end
```


Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

Intel 8086 (IBM PC):

```
cseg      segment para public 'CODE'
          assume   cs:cseg,ds:cseg

start:
          jmp      start1
msgstr    db        'Enter fahrenheit '
crlf      db        13,10,'$'
nine      dw        9
five      dw        5
outstr    db        'Centigrade is $'
start1:   push     ds
          push     cs
          pop      ds
```

Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

Intel 8086 (IBM PC):

```
        mov     dx,offset cseg:msgstr
        mov     ah,9
        int     21h

sloop:
cent:   call    getnumb
        test    ax,ax
        je     exit
        push    ax
        mov     dx,offset cseg:outstr
        mov     ah,9
        int     21h
        pop     ax
```

Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

Intel 8086 (IBM PC):

```
        sub     ax, 32
        jns     c1
        push    ax
        mov     dl, '-'
        mov     ah, 6
        int     21h
        pop     ax
        neg     ax
c1:      mul     five
        div     nine
        call    putval
        mov     dx, offset cseg:crlf
```

Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

Intel 8086 (IBM PC):

```
        mov     ah, 9
        int     21h
        jmp     sloop
exit:    pop     ds
        mov     ah, 4ch
        int     21h
getnumb:
        xor     bx, bx
11p:    mov     dl, 0ffh
        mov     ah, 1
        int     21h
        cmp     al, '0'
```


Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

Intel 8086 (IBM PC):

```
je      llr
sub     al,'0'
jb      llr
cmp     al,'9'
ja      llr
xor     ah,ah
shl     bx,1
add     ax,bx
shl     bx,1
shl     bx,1
add     bx,ax
jmp     llp
```

Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

Intel 8086 (IBM PC):

```
l1r:    mov     dx,offset cseg:crlf
        mov     ah,9
        int     21h
        mov     ax,bx
        ret
putval: xor     bx,bx
        push    bx
        mov     bx,10
l1g:    xor     dx,dx
        div     bx
        add     dx,'0'
        push    dx
```

Przegląd języków programowania asemblery

Języki ukierunkowane maszynowo. Każda instrukcja jest zamieniana na jeden rozkaz maszynowy. Języki te są zależne od sprzętu, tzn. dla każdej serii procesorów (*CPU* - skr. ang. *central processing unit*) język jest inny. Z uwagi na zorientowanie maszynowe programy assemblerowe są trudne do tworzenia i analizowania. W językach tych pisze się zwykle oprogramowanie systemowe (systemy operacyjne, systemy zarządzania bazami danych itp.), choć współcześnie częściej używa się do tego celu języka C. Języki assemblerowe znacznie różnią się dla różnych klas komputerów (zestawów typu *mainframe*, minikomputerów i mikrokomputerów).

Intel 8086 (IBM PC):

```
test    ax,ax
jne     11g
bloop:  pop    dx
        test   dx,dx
        je     endx
        mov    ah,6
        int    21h
        jmp    bloop
endx:    ret
cseg    ends
end      start
```

Przegląd języków programowania

Basic (Beginners All purpose Symbolic Instruction Code)

Język programowania opracowany przez Johna Kemeny'ego i Thomasa Kurtza z Dartmouth College (USA) w połowie lat 60-tych ub. stulecia. Początkowo był używany jako język interakcyjny na komputerach typu *mainframe*, a później szeroko stosowany na komputerach różnych rozmiarów. Istnieją zarówno interpretery, jak i kompilatory tego języka. Jest używany głównie przez początkujących programistów (z uwagi na swą prostotę). Nie jest językiem strukturalnym.

```
10 INPUT "Enter fahrenheit "; FAHR
20 PRINT "Centigrade is ", (FAHR-32)*5/9
```


Przegląd języków programowania

C

Język wyższego rzędu, opracowany w latach 70-tych ub. stulecia w laboratorium Bella (USA). Od końca lat 80-tych ub. stulecia jest głównym językiem do pisania oprogramowania komercyjnego. Składa się z wielu funkcji wywołujących się wzajemnie w procesie przetwarzania. Główny program jest też funkcją zwaną *main*. Z każdym kompilatorem producenci dostarczają standardową bibliotekę funkcji. W porównaniu z innymi językami programowania język C wydaje się dość skomplikowany (jest to spowodowane jego dużą elastycznością).

```
main()  {  
    float fahr;  
    printf("Enter fahrenheit ");  
    scanf("%f", &fahr);  
    printf("\nCentigrade is %f", (fahr-32)*5/9);  
}
```

Przegląd języków programowania

C++

Wersja języka C zorientowana obiektowo, opracowana przez Bjarne'a Stroustrupa.

Przegląd języków programowania

Cobol (Common Business Oriented Language)

Język wyższego rzędu, używany głównie w biznesie na minikomputerach i komputerach typu *mainframe*. Powstał w 1960 roku z języka Flowmatic opracowanego w połowie lat 50-tych ub. stulecia. Język ten wymaga dużo pisania, ale dzięki temu programy są łatwo czytelne. Każdy program składa się z czterech sekcji (części): identyfikacyjnej, określającej typ komputera, określającej dane i części proceduralnej.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID.    EXAMPLE.  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE-COMPUTER.    IBM-370.  
OBJECT-COMPUTER.    IBM-370.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77 FAHR PICTURE 999.  
77 CENT PICTURE 999.  
PROCEDURE DIVISION.  
DISPLAY 'Enter fahrenheit ', UPON CONSOLE.
```

Przegląd języków programowania

Cobol (Common Business Oriented Language)

Język wyższego rzędu, używany głównie w biznesie na minikomputerach i komputerach typu *mainframe*. Powstał w 1960 roku z języka Flowmatic opracowanego w połowie lat 50-tych ub. stulecia. Język ten wymaga dużo pisania, ale dzięki temu programy są łatwo czytelne. Każdy program składa się z czterech sekcji (części): identyfikacyjnej, określającej typ komputera, określającej dane i części proceduralnej.

```
ACCEPT FAHR FROM CONSOLE.  
COMPUTE CENT=(FAHR-32)*5/9.  
DISPLAY 'Centigrade is ' CENT UPON CONSOLE.  
GOBACK.
```


Przegląd języków programowania

Fortran (Formula Translation)

Pierwszy język programowania wyższego rzędu i kompilator opracowany w 1954 roku przez IBM. Został zaprojektowany do obliczeń naukowych (łatwość zapisu wyrażeń matematycznych), ale jest też używany w biznesie.

```
WRITE(6,*) 'Enter fahrenheit '  
READ(5,*) XFAHR  
XCENT=(XFAHR-32)*5/9  
WRITE(6,*) 'Centigrade is ',XCENT  
STOP  
END
```

Przegląd języków programowania

Java

Język programowania zorientowany obiektowo, stworzony przez grupę roboczą pod kierunkiem Jamesa Goslinga z firmy Sun Microsystems. Java jest językiem tworzenia programów źródłowych kompilowanych do kodu bajtowego, czyli postaci wykonywanej przez maszynę wirtualną (maszyna taka musi być zainstalowana na komputerze). Język cechuje się silnym typowaniem. Jego podstawowe koncepcje zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią) oraz z języka C++ (duża część składni i słów kluczowych).

Javy nie należy mylić ze skryptowym językiem JavaScript, z którym wspólną ma jedynie składnię podstawowych instrukcji.

Przegląd języków programowania

Lisp (List Processing)

Język wyższego rzędu używany głównie w programowaniu nienumerycznym, w którym przetwarza się obiekty, a nie liczby. Powstał w 1960 roku, a jego twórcą był John McCarthy. Syntaktyka i struktura tego języka znacznie odbiega od języków Basic, Cobol i Pascal (np. w języku Lisp nie ma syntaktycznej różnicy pomiędzy danymi i instrukcjami). Język ten może być modyfikowany i rozszerzany przez programistę.

```
(defun convert ()  
  (format t "Enter fahrenheit ")  
  (let ((fahr (read)))  
    (format t "Centigrade is ~D"  
      (truncate (* (-fahr 32)  
        (/ 5 9))))))
```

Przegląd języków programowania

Logo

Język wyższego rzędu, łatwy w użyciu, o dużych możliwościach graficznych (część graficzną tego języka nazywa się grafiką żółwiową - ang. *turtle graphics*). Twórcą języka był w połowie lat 60-tych ub. stulecia Seymour Papert wraz z kolegami z MIT i pracownikami firmy Bolt Beranek & Newman.

```
convert
local [fahr]
print "|Enter fahrenheit |
make "fahr ReadWord
print "|Centigrade is |
print (:fahr-32)*5/9
end
```


Przegląd języków programowania

Pascal

Język wyższego rzędu opracowany przez Niklausa Wirtha w połowie lat 60-tych ub. stulecia i nazwany na cześć wybitnego matematyka francuskiego Blaise'a Pascala (1623-1662). Ma wiele cech języka Algol 60, a ponadto dość rozbudowane mechanizmy definiowania struktur danych (rekordy, pliki, dynamiczne przydzielanie pamięci) i operowania nimi oraz procedury wejścia-wyjścia. Język Pascal wywarł duży wpływ na konstrukcję innych języków, np. Ada i dBase. Pochodną języka Pascal jest też opracowany w 1974 roku przez Duńczyka P. Brinch-Hansena język Concurrent Pascal dla programowania współbieżnego.

```
program convert
var
fahr, cent : integer;
begin
  write('Enter fahrenheit ');
  readln(fahr);
  cent:=(fahr-32)*5/9;
  writeln('Centigrade is ',cent)
end.
```

Przegląd języków programowania

PHP (akronim od *PHP Hypertext Preprocessor*, dawniej skrót od *Personal Home Page*)

Skryptowy język programowania zorientowany obiektowo zaprojektowany do generowania stron internetowych w czasie rzeczywistym.

PHP jest najczęściej stosowany do tworzenia skryptów po stronie serwera WWW, ale może być on również używany do przetwarzania danych z poziomu wiersza poleceń, a nawet do pisania programów pracujących w trybie graficznym (np. za pomocą biblioteki GTK+, używając rozszerzenia PHP-GTK). Implementacja PHP wraz z serwerem WWW Apache oraz serwerem baz danych MySQL określana jest jako platforma AMP (w środowisku Linux – LAMP, w Windows – WAMP).

Przegląd języków programowania

PL/1 (Programming Language 1)

Język wyższego rzędu opracowany przez IBM w 1964 roku dla komputerów rodziny System/360. W pewnym sensie jest kombinacją języków Cobol i Fortran. Program jest zbudowany z modułów zwanych procedurami, przy czym zawsze występuje tzw. procedura główna. Odmianą tego języka jest PL/M (Programming Language for Microprocessors) opracowany przez firmę Intel dla mikroprocesorów tej firmy.

```
CONVERT: PROCEDURE OPTIONS (MAIN);  
  DECLARE (FAHR, CENT) DECIMAL FLOAT (15);  
  PUT EDIT ('ENTER FAHRENHEIT') (SKIP,A);  
  GET EDIT (FAHR);  
  CENT=(FAHR-32)*5/9;  
  PUT EDIT ('CENTIGRADE IS ',CENT) (SKIP,A,F(6,2));  
END CONVERT;
```

Przegląd języków programowania

Prolog (Programming in Logic)

Język wyższego rzędu używany głównie do tworzenia systemów sztucznej inteligencji (przetwarzanie języków naturalnych, systemy eksperckie itp.), opracowany we Francji w 1973 roku. Jest podobny do języka Lisp (symboliczna reprezentacja obiektów).

```
convert:-write('Enter fahrenheit'),  
         read(Fahr),  
         write('Centigrade is '),  
         Cent is (5*(Fahr-32))/9,  
         write(Cent),nl.
```


Przegląd języków programowania

Python

Język programowania wysokiego poziomu ogólnego przeznaczenia, o rozbudowanym pakiecie bibliotek standardowych, którego ideą przewodnią jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością.

Język Python wspiera różne paradygmaty programowania - jest obiektowy, imperatywny (ciąg wykonywanych po sobie instrukcji) oraz w mniejszym stopniu funkcyjny. Posiada w pełni dynamiczny system typów i automatyczne zarządzanie pamięcią. Interpreter języka są dostępne na wiele systemów operacyjnych.

Język Python jest rozwijany jako projekt *Open Source* zarządzany przez *Python Software Foundation*, która jest organizacją *nonprofit*.

(źródło: pl.wikipedia.org)

Przegląd języków programowania

Ruby

Język programowania interpretowany, w pełni obiektowy i dynamicznie typowany, stworzony w 1995 roku przez Yukihiro Matsumoto (pseudonim *Matz*).

Od roku 2003 język zdobywa lawinowo nowych zwolenników, głównie za sprawą popularnego *frameworku* do tworzenia aplikacji internetowych o nazwie *Ruby on Rails*.

(źródło: pl.wikipedia.org)

Przegląd języków programowania

Ranking popularności

Źródło: TIOBE Software –TioBE Index

Sep 2024	Sep 2023	Change	Programming Language	Ratings	Change
1	1		 Python	20.17%	+6.01%
2	3	▲	 C++	10.75%	+0.09%
3	4	▲	 Java	9.45%	-0.04%
4	2	▼	 C	8.89%	-2.38%
5	5		 C#	6.08%	-1.22%
6	6		 JavaScript	3.92%	+0.62%
7	7		 Visual Basic	2.70%	+0.48%
8	12	▲	 Go	2.35%	+1.16%
9	10	▲	 SQL	1.94%	+0.50%
10	11	▲	 Fortran	1.78%	+0.49%
11	15	▲	 Delphi/Object Pascal	1.77%	+0.75%
12	13	▲	 MATLAB	1.47%	+0.28%
13	8	▼	 PHP	1.46%	-0.09%
14	17	▲	 Rust	1.32%	+0.35%
15	18	▲	 R	1.20%	+0.23%
16	19	▲	 Ruby	1.13%	+0.18%
17	14	▼	 Scratch	1.11%	+0.03%
18	20	▲	 Kotlin	1.10%	+0.20%
19	21	▲	 COBOL	1.09%	+0.22%
20	16	▼	 Swift	1.08%	+0.09%

„It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written”.

Sieci działań

Sieć działań (schemat blokowy, ang. *flow chart*)

- graficzna reprezentacja sekwencji operacji w systemie informatycznym lub programie.

W systemie informatycznym schemat blokowy pokazuje jak dane przepływają od dokumentów źródłowych przez komputer do końcowych użytkowników, czyli dystrybucję danych (informacji).

Schematy blokowe programów pokazują sekwencję instrukcji w poszczególnych programach.

Na ogół w obu rodzajach schematów występują inne symbole graficzne.

Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Przetwarzanie

operacja lub grupa operacji, w wyniku których ulega zmianie wartość, postać lub miejsce zapisu danych



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

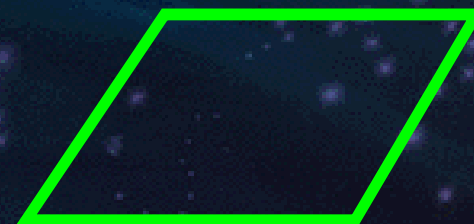
Nazwa operacji

Wyjaśnienie

Symbol

Wprowadzanie,
wyprowadzanie

wprowadzanie lub (i)
wyprowadzanie
danych



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Decyzja

operacja określająca
wybór jednej
z alternatywnych
dróg działania



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

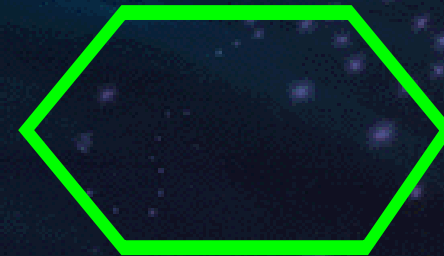
Nazwa operacji

Wyjaśnienie

Symbol

Przygotowanie

modyfikacja instrukcji
lub grupy instrukcji
powodująca zmianę
w przebiegu procesu
przetwarzania



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

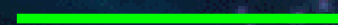
Nazwa operacji

Wyjaśnienie

Symbol

Droga
przepływu
danych

wież operacyjna między poszczególnymi operacjami procesu przetwarzania



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Droga
przepływu
danych
o wskazanym
kierunku

wskazanie kierunku
przepływu danych, gdy
kierunek ten nie jest
jednoznacznie zrozu-
miały



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Skrzyżowanie
dróg przepływu
danych bez
powiązania
między nimi



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Łączenie dróg
przepływu
danych



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Łącznik
międzystroni-
cowy

wejście lub wyjście
z wyodrębnionych
fragmentów schematu
znajdujących się na
różnych stronach



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Łącznik
stronicowy

wejście lub wyjście
z wyodrębnionych
fragmentów schematu
znajdujących się na
jednej stronie



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Początek,
koniec
lub przerwa

oznaczenie miejsca
rozpoczęcia, zakoń-
czenia lub przerwania
działania programu



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Działanie
równoległe

oznaczenie miejsca
rozpoczęcia lub zakoń-
czenia dwu lub więcej
operacji jednoczes-
nych



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

Nazwa operacji

Wyjaśnienie

Symbol

Komentarz

oznaczenie miejsca
na komentarz



Symbole stosowane w sieciach działań programów

(PN-75/E-01226)

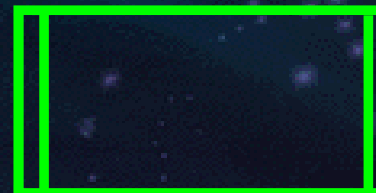
Nazwa operacji

Wyjaśnienie

Symbol

Proces
zdefiniowany

proces zdefiniowany
poza programem



Symbole stosowane w sieciach systemów informatycznych

ekran



taśma magnetyczna



drukarka



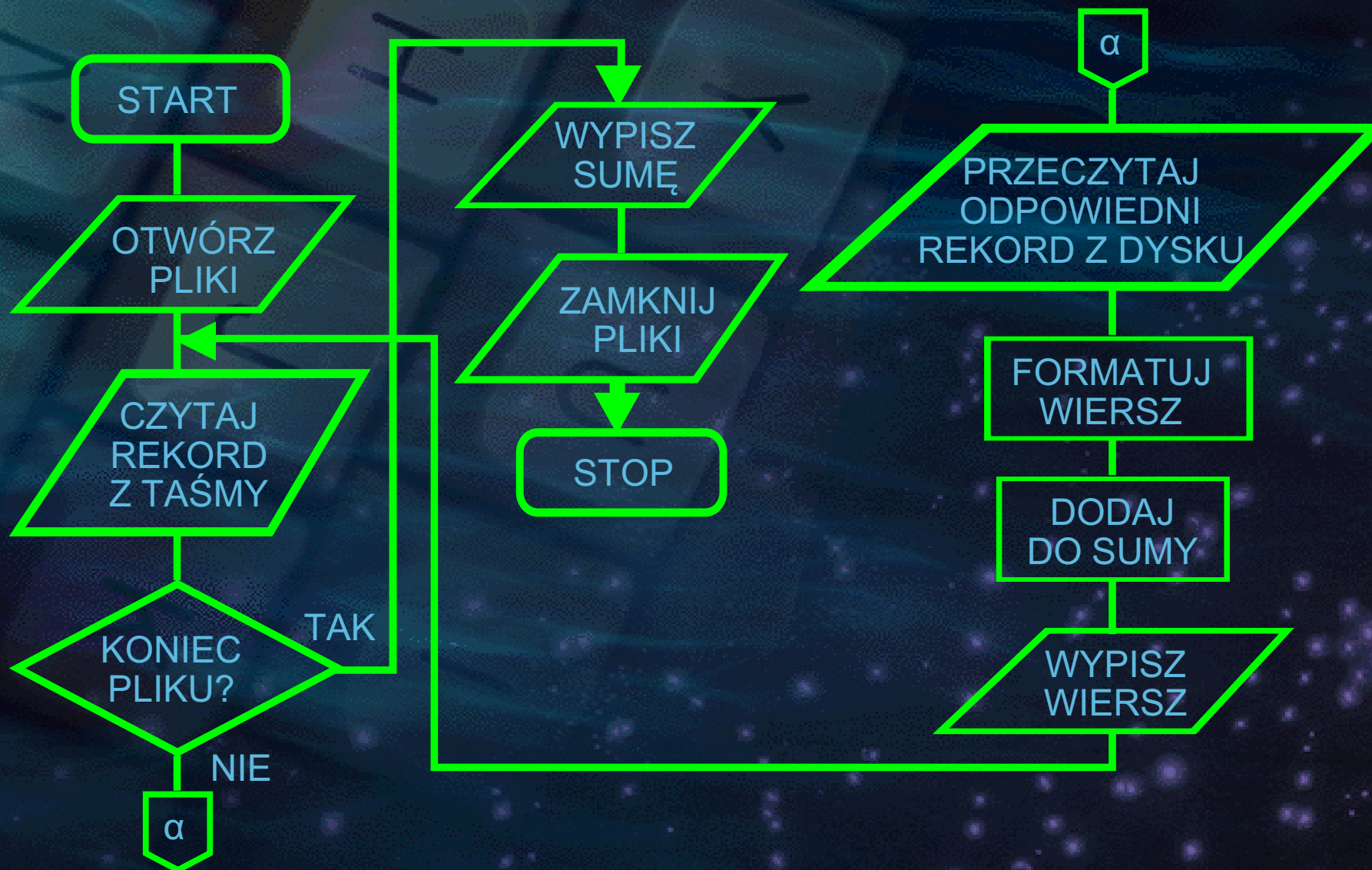
dysk magnetyczny



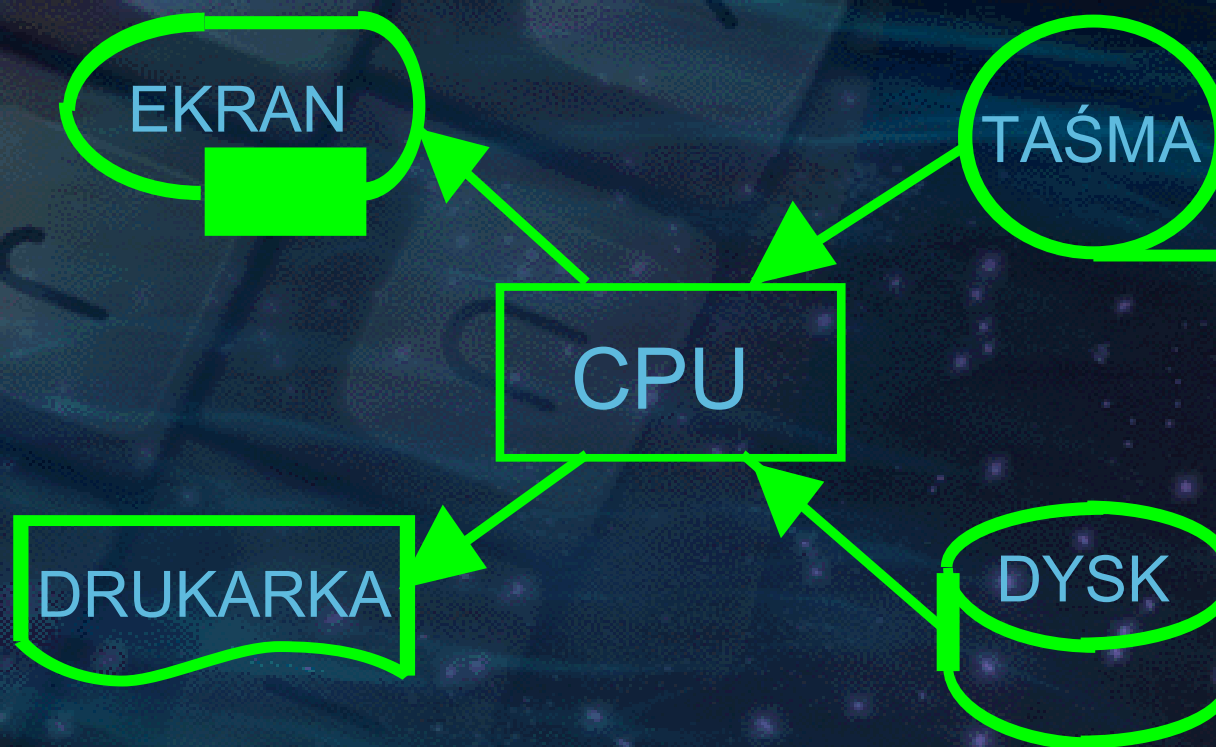
urządzenie do komunikacji
operatora (użytkownika)
z komputerem (np. klawiatura)



Przykład schematu blokowego programu



Przykład schematu blokowego systemu



Ten i następne wykłady są dostępne w formatach
PDF i EXE na stronie

www.cs.put.poznan.pl/amarciniak

podstrona  **Dydaktyka**



cdn...