Podstawy Programowania

Wykład I

Wprowadzenie

Robert Muszyński
Katedra Cybernetyki i Robotyki
Politechnika Wrocławska

email: robert.muszynski@pwr.edu.pl
pokój: 331 budynek C3 (Kompleks Wydziału Elektroniki...)
GPS: N 51°06'31.75" E 17°03'38.02"

– Skład FoilTEX –

Literatura

1. B. Kernighan, D. Ritchie, **Język ANSI C**, WNT, (1987), 1994, 2007
2. G. Glass, K. Ables, **Linux dla programistów i użytkowników**, Helion, 2007
3. B. W. Kernighan, R. Pike, **Lekcja programowania**, WNT, 2002
4. T. Cormen, C. Leiserson, R. Rivest, **Wprowadzenie do algorytmów**, WNT 1998, 2007
5. N. Wirth, **Algorytmy + struktury danych = programy**, WNT, 1980, 2004
6. S. Granneman, **Linux. Rozmówki**, Helion, 2006
7. D. Cameron, **GNU Emacs**, Helion, 2002

Literatura

1. B. Kernighan, D. Ritchie, **Język ANSI C**, WNT, (1987), 1994, 2007
2. G. Glass, K. Ables, **Linux dla programistów i użytkowników**, Helion, 2007
3. B. W. Kernighan, R. Pike, **Lekcja programowania**, WNT, 2002
4. T. Cormen, C. Leiserson, R. Rivest, **Wprowadzenie do algorytmów**, WNT 1998, 2007
5. N. Wirth, **Algorytmy + struktury danych = programy**, WNT, 1980, 2004
6. S. Granneman, **Linux. Rozmówki**, Helion, 2006
7. D. Cameron, **GNU Emacs**, Helion, 2002

Literatura

1. B. Kernighan, D. Ritchie, **Język ANSI C**, WNT, (1987), 1994, 2007
2. G. Glass, K. Ables, **Linux dla programistów i użytkowników**, Helion, 2007
3. B. W. Kernighan, R. Pike, **Lekcja programowania**, WNT, 2002
4. T. Cormen, C. Leiserson, R. Rivest, **Wprowadzenie do algorytmów**, WNT 1998, 2007
5. N. Wirth, **Algorytmy + struktury danych = programy**, WNT, 1980, 2004
6. S. Granneman, **Linux. Rozmówki**, Helion, 2006
7. D. Cameron, **GNU Emacs**, Helion, 2002

Literatura

1. B. Kernighan, D. Ritchie, **Język ANSI C**, WNT, (1987), 1994, 2007
2. G. Glass, K. Ables, **Linux dla programistów i użytkowników**, Helion, 2007
3. B. W. Kernighan, R. Pike, **Lekcja programowania**, WNT, 2002
4. T. Cormen, C. Leiserson, R. Rivest, **Wprowadzenie do algorytmów**, WNT 1998, 2007
5. N. Wirth, **Algorytmy + struktury danych = programy**, WNT, 1980, 2004
6. S. Granneman, **Linux. Rozmówki**, Helion, 2006
7. D. Cameron, **GNU Emacs**, Helion, 2002

W sieci

1. www.kcir.pwr.edu.pl — strona Katedry Cybernetyki i Robotyki
2. kcir.pwr.edu.pl/~mucha/PProg — strona tego kursu
3. diablo.kcir.pwr.edu.pl — strona studenckiego serwera diablo
4. diablo.kcir.pwr.edu.pl/pomoc — strona pomocy: studenci AiR studentom AiR

W sieci

1. www.kcir.pwr.edu.pl — strona Katedry Cybernetyki i Robotyki
2. kcir.pwr.edu.pl/~mucha/PProg — strona tego kursu
3. diablo.kcir.pwr.edu.pl — strona studenckiego serwera diablo
4. diablo.kcir.pwr.edu.pl/pomoc — strona pomocy: studenci AiR studentom AiR
5. www.kcir.pwr.edu.pl/~witold/info3/n1124.pdf — dokumentacja języka C standard ISO/IEC 9899
6. wazniak.mimuw.edu.pl/index.php?title=Wstęp_do_programowania — strona kursu Wstęp do programowania ze Studiów informatycznych

Zawartość tematyczna wykładu

- System operacyjny UNIX — narzędzia, powłoki, środowisko graficzne.
- Język C: gramatyka, kompilacja, struktury danych, wyrażenia i instrukcje.
- Algorytmy: metody konstruowania, weryfikacja poprawności.
- Operacje wejścia/wyjścia. Strumienie danych.
- Struktury sterujące programu: sekwencja, wybór warunkowy, iteracja.
- Funkcje, ich parametry. Moduły.
- Tablice. Przeszukiwanie i sortowanie.
- Reguły stylu programowania. Dokumentacja programu.
- Wskaźniki, zmienne dynamiczne.
- Struktury, listy, stosy, sterty, kolejki FIFO, priorytetowe.
- Drzewa.
- Efektywność programów.

Zawartość tematyczna wykładu

- System operacyjny UNIX — narzędzia, powłoki, środowisko graficzne.
- Język C: gramatyka, kompilacja, struktury danych, wyrażenia i instrukcje.
- Algorytmy: metody konstruowania, weryfikacja poprawności.
- Operacje wejścia/wyjścia. Strumienie danych.
- Struktury sterujące programu: sekwencja, wybór warunkowy, iteracja.
- Funkcje, ich parametry. Moduły.
- Tablice. Przeszukiwanie i sortowanie.
- Reguły stylu programowania. Dokumentacja programu.
- Wskaźniki, zmienne dynamiczne.
- Struktury, listy, stosy, sterty, kolejki FIFO, priorytetowe.
- Drzewa.
- Efektywność programów.

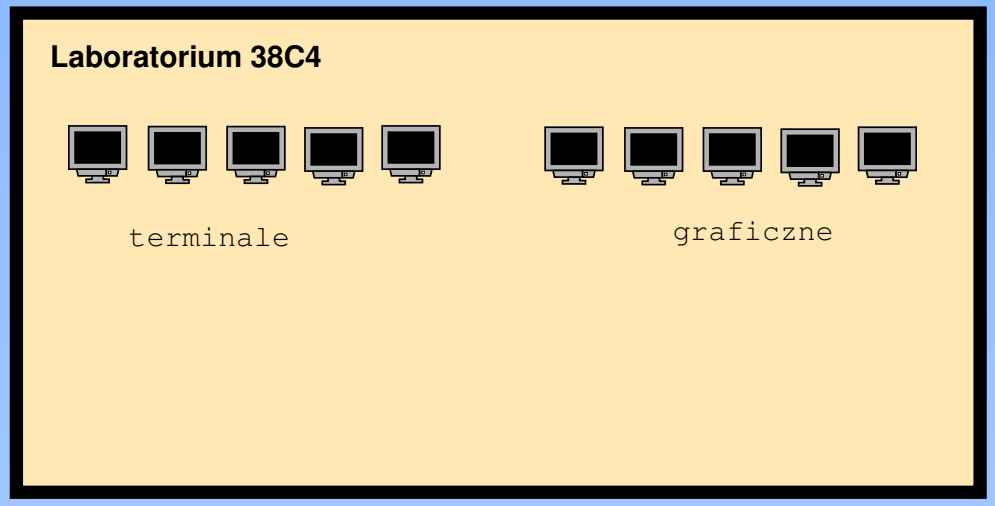
Zawartość tematyczna wykładu

- System operacyjny UNIX — narzędzia, powłoki, środowisko graficzne.
- Język C: gramatyka, kompilacja, struktury danych, wyrażenia i instrukcje.
- Algorytmy: metody konstruowania, weryfikacja poprawności.
- Operacje wejścia/wyjścia. Strumienie danych.
- Struktury sterujące programu: sekwencja, wybór warunkowy, iteracja.
- Funkcje, ich parametry. Moduły.
- Tablice. Przeszukiwanie i sortowanie.
- Reguły stylu programowania. Dokumentacja programu.
- Wskaźniki, zmienne dynamiczne.
- Struktury, listy, stosy, sterty, kolejki FIFO, priorytetowe.
- Drzewa.
- Efektywność programów.

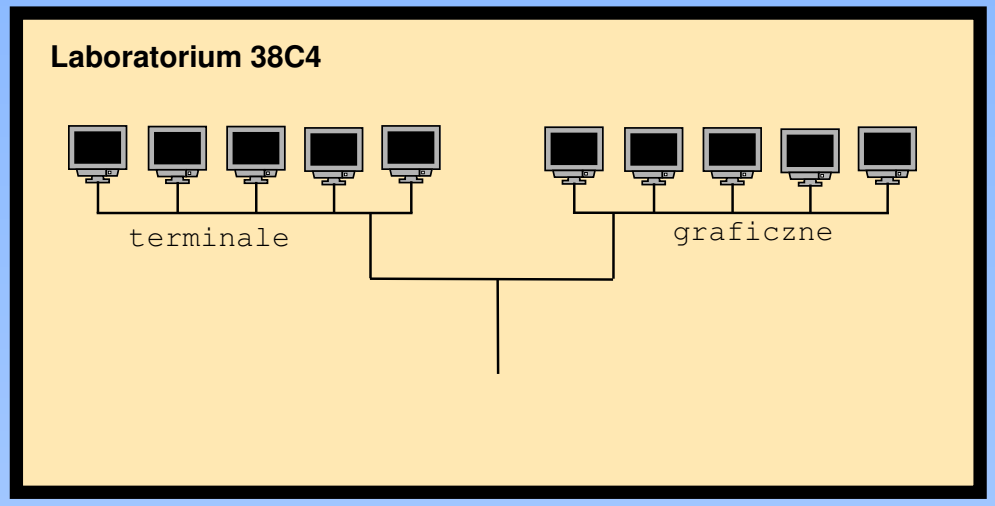
Zawartość tematyczna wykładu

- System operacyjny UNIX — narzędzia, powłoki, środowisko graficzne.
- Język C: gramatyka, kompilacja, struktury danych, wyrażenia i instrukcje.
- Algorytmy: metody konstruowania, weryfikacja poprawności.
- Operacje wejścia/wyjścia. Strumienie danych.
- Struktury sterujące programu: sekwencja, wybór warunkowy, iteracja.
- Funkcje, ich parametry. Moduły.
- Tablice. Przeszukiwanie i sortowanie.
- Reguły stylu programowania. Dokumentacja programu.
- Wskaźniki, zmienne dynamiczne.
- Struktury, listy, stosy, sterty, kolejki FIFO, priorytetowe.
- Drzewa.
- Efektywność programów.

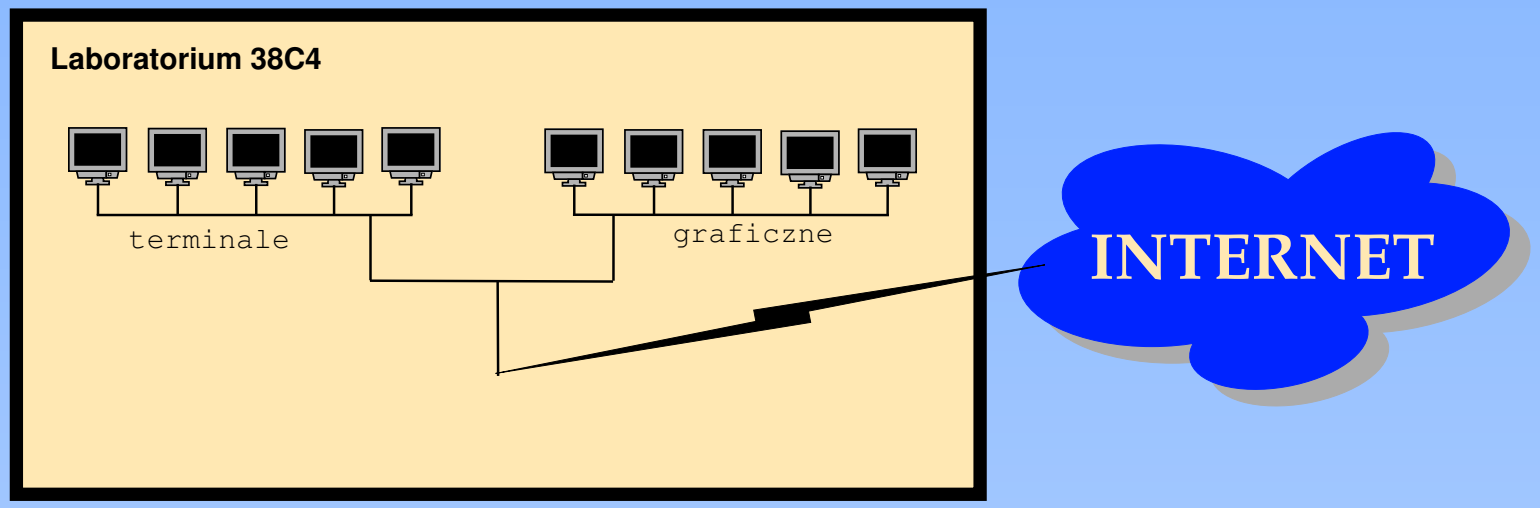
Środowisko pracy



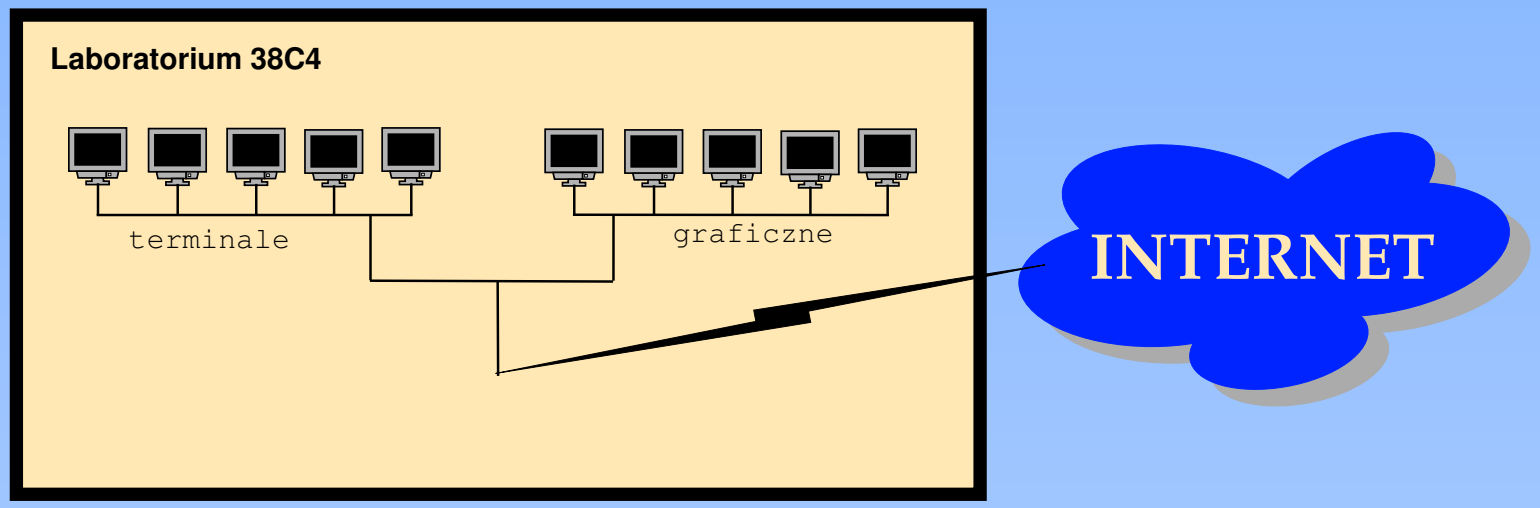
Środowisko pracy

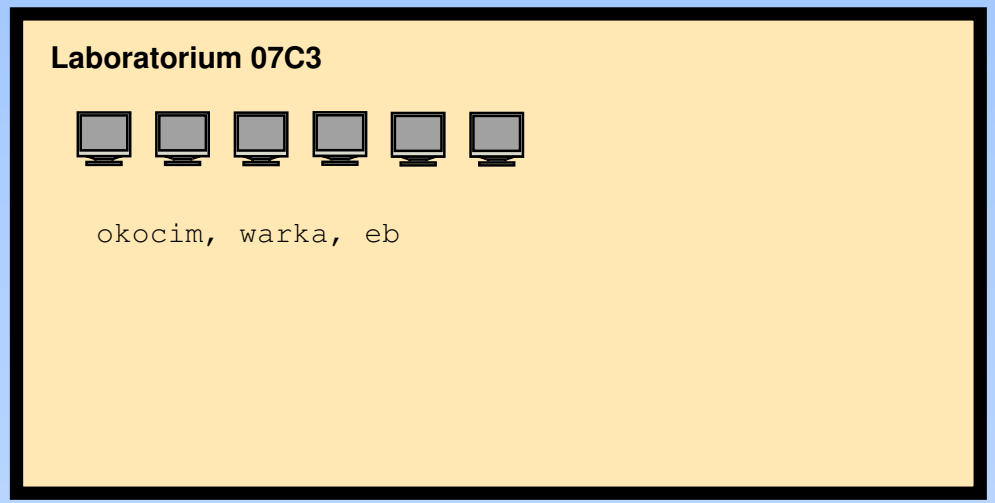


Środowisko pracy

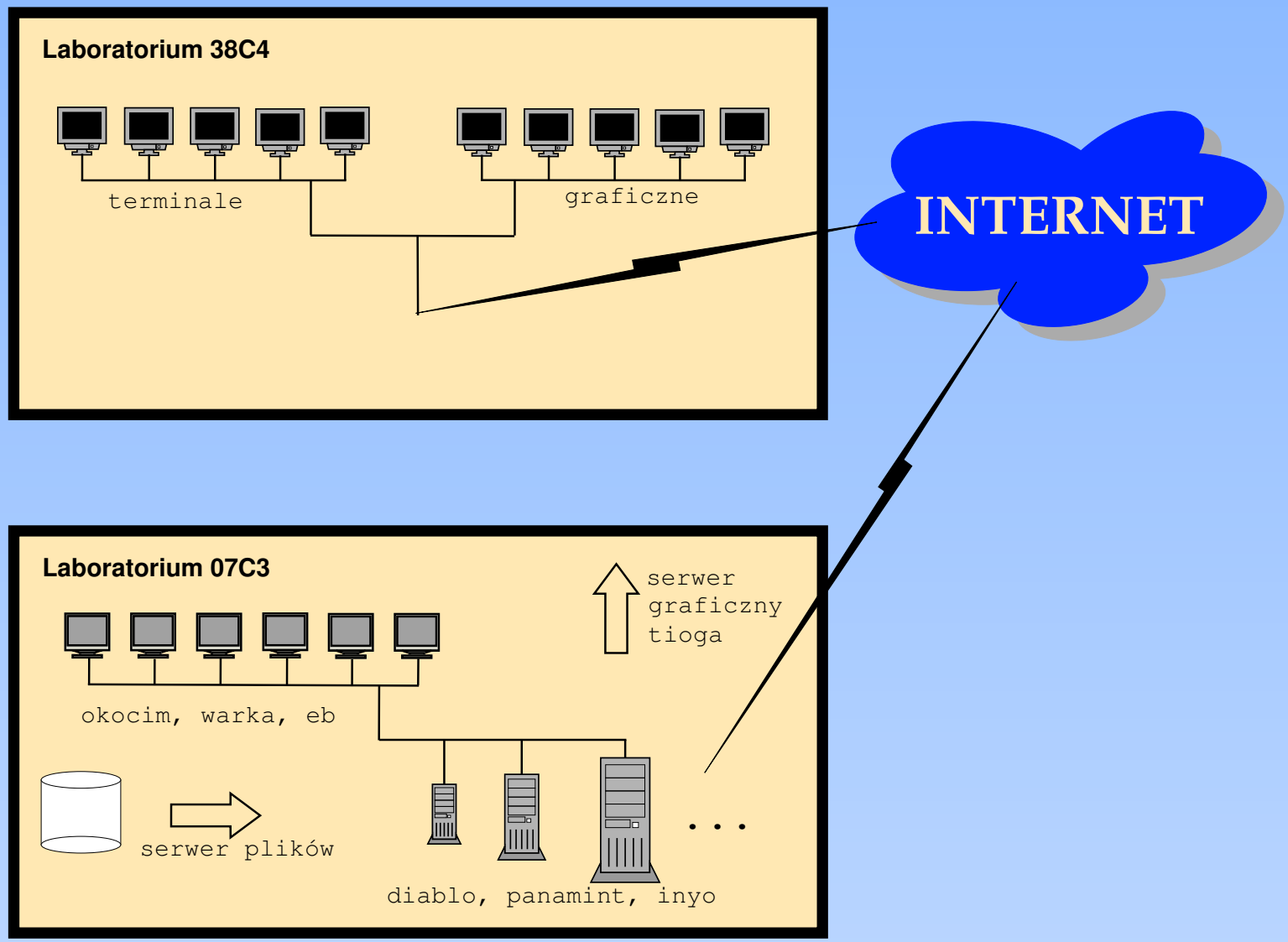


Środowisko pracy

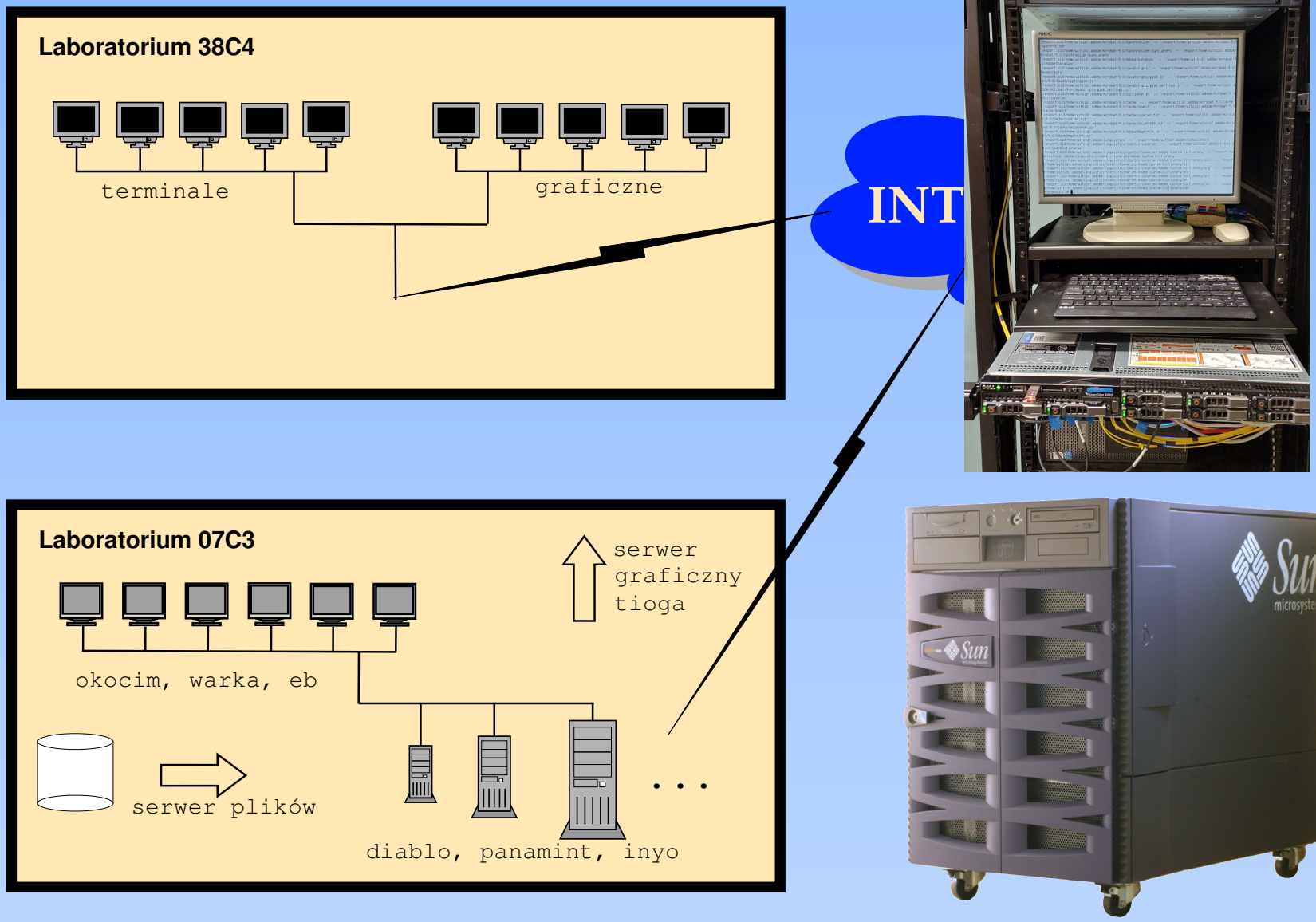




Środowisko pracy

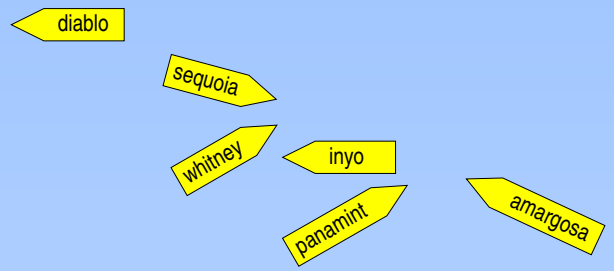


Środowisko pracy

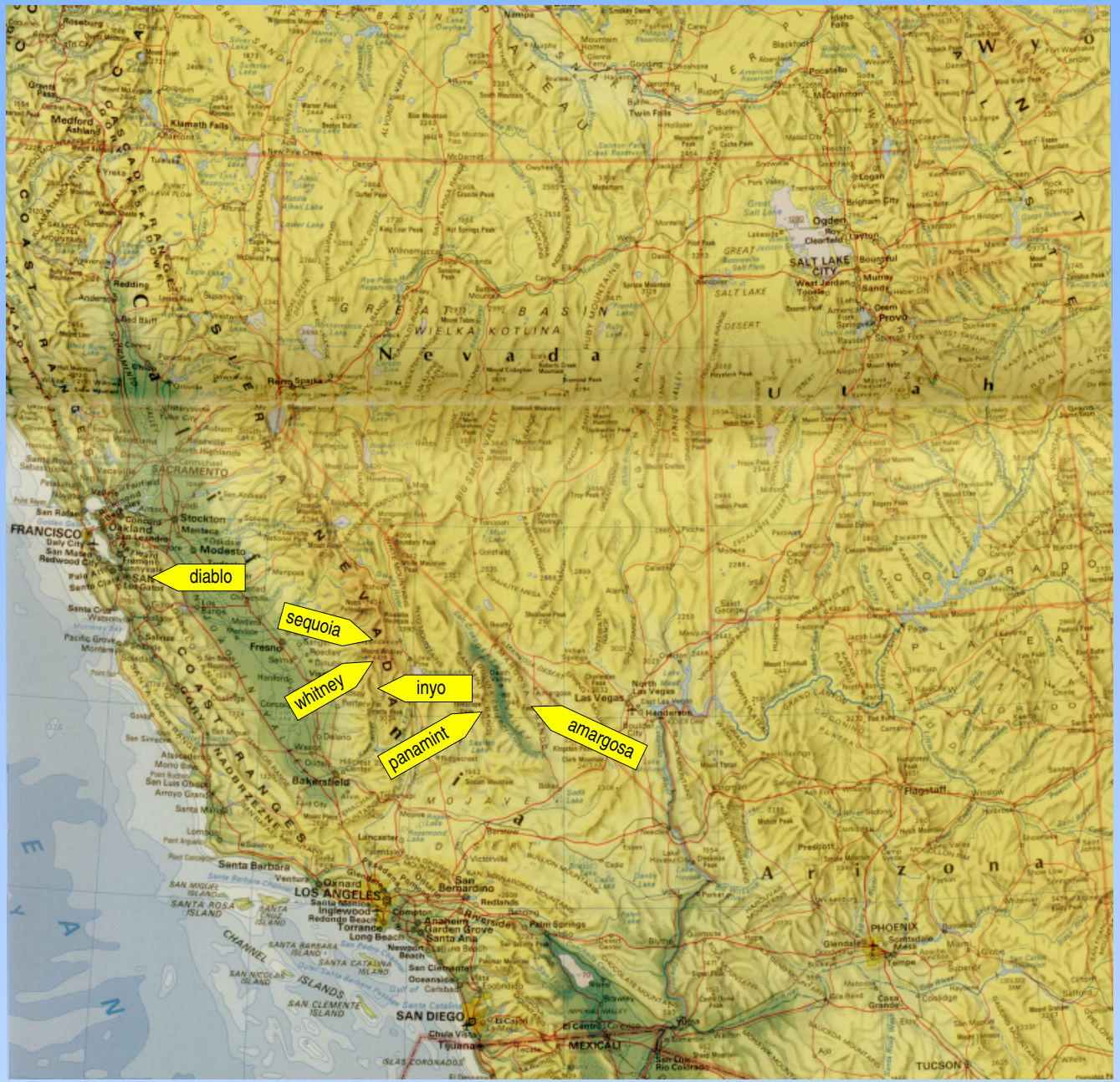


Środowisko pracy

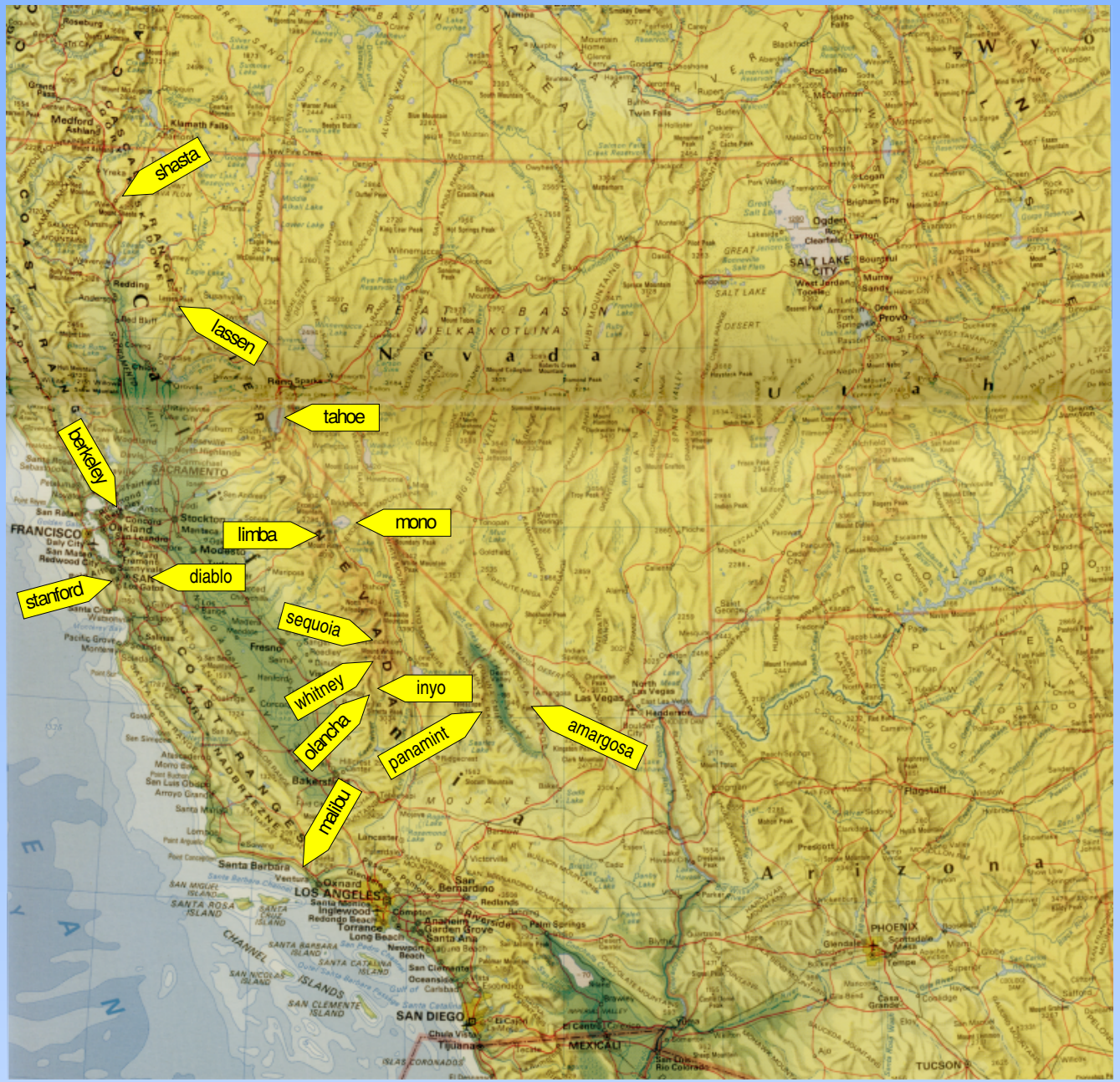
Topologia sieci (;-p)



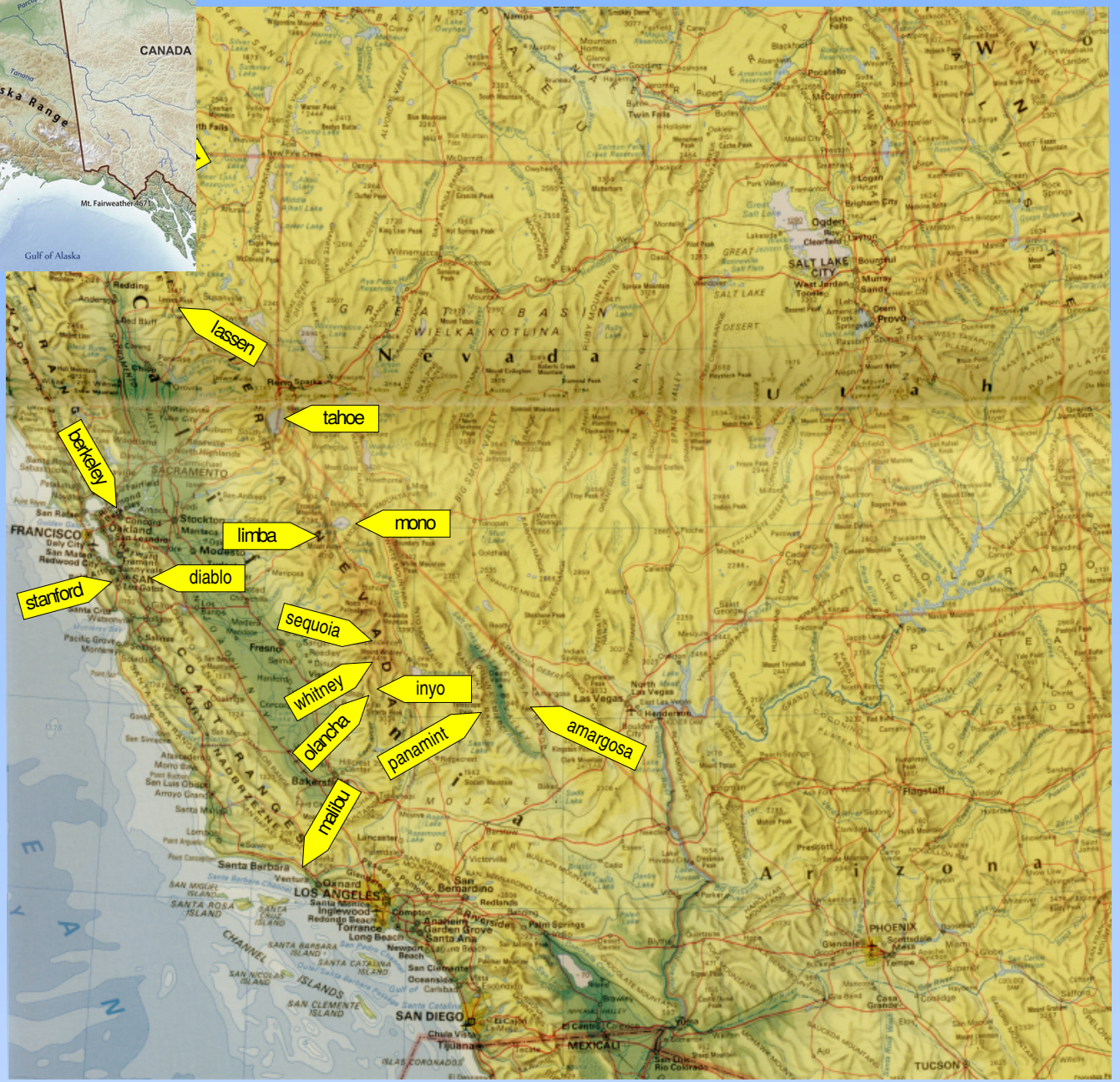
Topologia sieci (;-p)



Topologia sieci (;-p)



Topologia sieci (;-p)



Dostęp z terenu Kampusu PWr (i spoza niego)

- dowolne laboratorium komputerowe z dostępem do internetu

Dostęp z terenu Kampusu PWr (i spoza niego)

- dowolne laboratorium komputerowe z dostępem do internetu
- PWR-WiFi — otwarta sieć bezprzewodowa dostępna na terenach Politechniki Wrocławskiej (jednakże z zablokowanymi wieloma portami, w tym portem 22 (ssh, scp, sftp – remote login protocols))

Dostęp z terenu Kampusu PWr (i spoza niego)

- dowolne laboratorium komputerowe z dostępem do internetu
- PWR-WiFi — otwarta sieć bezprzewodowa dostępna na terenach Politechniki Wrocławskiej (jednakże z zablokowanymi wieloma portami, w tym portem 22 (ssh, scp, sftp – remote login protocols))
- eduroam — miedzynarodowa, zamknięta sieć bezprzewodowa dostępna na terenie większości uczelni (dla studentów i pracowników naukowych po uprzednim skonfigurowaniu i jednorazowym zalogowaniu z terenu uczelni macierzystej — więcej na `http://eduroam.pwr.edu.pl/`)

Dostęp z terenu Kampusu PWr (i spoza niego)

- dowolne laboratorium komputerowe z dostępem do internetu
- PWR-WiFi — otwarta sieć bezprzewodowa dostępna na terenach Politechniki Wrocławskiej (jednakże z zablokowanymi wieloma portami, w tym portem 22 (ssh, scp, sftp – remote login protocols))
- eduroam — miedzynarodowa, zamknięta sieć bezprzewodowa dostępna na terenie większości uczelni (dla studentów i pracowników naukowych po uprzednim skonfigurowaniu i jednorazowym zalogowaniu z terenu uczelni macierzystej — więcej na `http://eduroam.pwr.edu.pl/`)

Dostęp z terenu Kampusu PWr (i spoza niego)

- dowolne laboratorium komputerowe z dostępem do internetu
- PWR-WiFi — otwarta sieć bezprzewodowa dostępna na terenach Politechniki Wrocławskiej (jednakże z zablokowanymi wieloma portami, w tym portem 22 (ssh, scp, sftp – remote login protocols))
- eduroam — miedzynarodowa, zamknięta sieć bezprzewodowa dostępna na terenie większości uczelni (dla studentów i pracowników naukowych po uprzednim skonfigurowaniu i jednorazowym zalogowaniu z terenu uczelni macierzystej — więcej na `http://eduroam.pwr.edu.pl/`)
- dostęp przez dowolnego klienta ssh (ssh, putty), zdalny pulpit X2Go

Dostęp z terenu Kampusu PWr (i spoza niego)

- dowolne laboratorium komputerowe z dostępem do internetu
- PWR-WiFi — otwarta sieć bezprzewodowa dostępna na terenach Politechniki Wrocławskiej (jednakże z zablokowanymi wieloma portami, w tym portem 22 (ssh, scp, sftp – remote login protocols))
- eduroam — miedzynarodowa, zamknięta sieć bezprzewodowa dostępna na terenie większości uczelni (dla studentów i pracowników naukowych po uprzednim skonfigurowaniu i jednorazowym zalogowaniu z terenu uczelni macierzystej — więcej na `http://eduroam.pwr.edu.pl/`)
- dostęp przez dowolnego klienta ssh (ssh, putty), zdalny pulpit X2Go

---

- z wykorzystaniem wirtualnej sieci prywatnej PWr-VPN (więcej na `https://di.pwr.edu.pl/uslugi/siec/vpn`)

Podstawowe pojęcia

Komputer — zestaw odpowiednio dobranych urządzeń...

System operacyjny — program komputerowy, który zarządza sprzętem oraz aplikacjami komputera.

Podstawowe pojęcia

Komputer — zestaw odpowiednio dobranych urządzeń...

System operacyjny — program komputerowy, który zarządza sprzętem oraz aplikacjami komputera.

Program komputerowy — ciąg poleceń wykonywanych przez komputer w celu realizacji zadanego algorytmu.

Podstawowe pojęcia

Komputer — zestaw odpowiednio dobranych urządzeń...

System operacyjny — program komputerowy, który zarządza sprzętem oraz aplikacjami komputera.

Program komputerowy — ciąg poleceń wykonywanych przez komputer w celu realizacji zadanego algorytmu.

Algorytm — ciąg czynności prowadzących do rozwiązania zadania.

Podstawowe pojęcia

Komputer — zestaw odpowiednio dobranych urządzeń...

System operacyjny — program komputerowy, który zarządza sprzętem oraz aplikacjami komputera.

Program komputerowy — ciąg poleceń wykonywanych przez komputer w celu realizacji zadanego algorytmu.

Algorytm — ciąg czynności prowadzących do rozwiązania zadania.

Specyfikacja zadania — określenie dopuszczalnych danych wejściowych i oczekiwanych wyników jako funkcji danych wejściowych, charakteryzujące stawiany problem.

Problem — zadanie do rozwiązania.

Systemy operacyjne

- Windows, Mac OS, OpenVMS — przeznaczone dla jednej rodziny sprzętu
- UNIX, Linux — dostępne dla różnych platform sprzętowych

Systemy operacyjne

- Windows, Mac OS, OpenVMS — przeznaczone dla jednej rodziny sprzętu
- UNIX, Linux — dostępne dla różnych platform sprzętowych

pierwsza zaleta Linuksa

Systemy operacyjne

- Windows, Mac OS, OpenVMS — przeznaczone dla jednej rodziny sprzętu
- UNIX, Linux — dostępne dla różnych platform sprzętowych

pierwsza zaleta Linuksa

- Windows, Mac OS, UNIX — tworzone pod presją harmonogramów
- Linux — dzieło tysięcy doświadczonych programistów–ochotników

Systemy operacyjne

- Windows, Mac OS, OpenVMS — przeznaczone dla jednej rodziny sprzętu
- UNIX, Linux — dostępne dla różnych platform sprzętowych

pierwsza zaleta Linuksa

- Windows, Mac OS, UNIX — tworzone pod presją harmonogramów
- Linux — dzieło tysięcy doświadczonych programistów–ochotników

druga zaleta Linuksa

Systemy operacyjne

- Windows, Mac OS, OpenVMS — przeznaczone dla jednej rodziny sprzętu
- UNIX, Linux — dostępne dla różnych platform sprzętowych

pierwsza zaleta Linuksa

- Windows, Mac OS, UNIX — tworzone pod presją harmonogramów
- Linux — dzieło tysięcy doświadczonych programistów–ochotników

druga zaleta Linuksa

- Windows, Mac OS — pozwalają na pracę jednego użytkownika
- UNIX, Linux, OpenVMS — systemy wielodostępne

Systemy operacyjne

- Windows, Mac OS, OpenVMS — przeznaczone dla jednej rodziny sprzętu
- UNIX, Linux — dostępne dla różnych platform sprzętowych

pierwsza zaleta Linuksa

- Windows, Mac OS, UNIX — tworzone pod presją harmonogramów
- Linux — dzieło tysięcy doświadczonych programistów–ochotników

druga zaleta Linuksa

- Windows, Mac OS — pozwalają na pracę jednego użytkownika
- UNIX, Linux, OpenVMS — systemy wielodostępne

trzecia zaleta Linuksa

Systemy operacyjne cd.

- Windows, Mac OS, UNIX — oprogramowanie komercyjne
- Linux — tysiące narzędzi GNU, Open Source

czwarta zaleta Linuksa

- Linux „z wierzchu” wygląda dokładnie tak jak UNIX

piąta zaleta Linuksa

- Linux jest znacznie lepiej napisany niż UNIX

szósta zaleta Linuksa

- Linux pozwala...

siódma zaleta Linuksa

Krótka historia Linuksa

- na początku był MULTICS (1964)

Krótka historia Linuksa

- na początku był MULTICS (1964)
- 1969 — Ken Thomson tworzy w asemblerze system UNICS
- 1971 — wraz z Ritchiem przepisuje go w języku C — powstaje UNIX
- 1971 — Bell Lab udostępnia uniwersytetom kod źródłowy UNIX-a

Krótka historia Linuksa

- na początku był MULTICS (1964)
- 1969 — Ken Thomson tworzy w asemblerze system UNICS
- 1971 — wraz z Ritchiem przepisuje go w języku C — powstaje UNIX
- 1971 — Bell Lab udostępnia uniwersytetom kod źródłowy UNIX-a
- 1975–85 — powstaje BSD UNIX i UNIX System V
- później powstaje Solaris (Sun), HP-UX (Hewlett-Packard), AIX (IBM), IRIX (Silicon Graphics) — wojna o UNIX-a
- w tym czasie powstaje koncepcja oprogramowania GNU

Krótka historia Linuksa

- na początku był MULTICS (1964)
- 1969 — Ken Thomson tworzy w asemblerze system UNICS
- 1971 — wraz z Ritchiem przepisuje go w języku C — powstaje UNIX
- 1971 — Bell Lab udostępnia uniwersytetom kod źródłowy UNIX-a
- 1975–85 — powstaje BSD UNIX i UNIX System V
- później powstaje Solaris (Sun), HP-UX (Hewlett-Packard), AIX (IBM), IRIX (Silicon Graphics) — wojna o UNIX-a
- w tym czasie powstaje koncepcja oprogramowania GNU
- 1991 – Linus Torvalds z grupą przyjaciół zaczyna pracę nad jądrem systemu

Krótka historia Linuksa

- na początku był MULTICS (1964)
- 1969 — Ken Thomson tworzy w asemblerze system UNICS
- 1971 — wraz z Ritchiem przepisuje go w języku C — powstaje UNIX
- 1971 — Bell Lab udostępnia uniwersytetom kod źródłowy UNIX-a
- 1975–85 — powstaje BSD UNIX i UNIX System V
- później powstaje Solaris (Sun), HP-UX (Hewlett-Packard), AIX (IBM), IRIX (Silicon Graphics) — wojna o UNIX-a
- w tym czasie powstaje koncepcja oprogramowania GNU
- 1991 – Linus Torvalds z grupą przyjaciół zaczyna pracę nad jądrem systemu
- 1994 – udostępnia on system Linux 1.0 na zasadach GNU GPL

Krótka historia Linuksa

- na początku był MULTICS (1964)
- 1969 — Ken Thomson tworzy w asemblerze system UNICS
- 1971 — wraz z Ritchiem przepisuje go w języku C — powstaje UNIX
- 1971 — Bell Lab udostępnia uniwersytetom kod źródłowy UNIX-a
- 1975–85 — powstaje BSD UNIX i UNIX System V
- później powstaje Solaris (Sun), HP-UX (Hewlett-Packard), AIX (IBM), IRIX (Silicon Graphics) — wojna o UNIX-a
- w tym czasie powstaje koncepcja oprogramowania GNU
- 1991 – Linus Torvalds z grupą przyjaciół zaczyna pracę nad jądrem systemu
- 1994 – udostępnia on system Linux 1.0 na zasadach GNU GPL
- obecnie tysiące ochotników pracuje dostarczając wielu dystrybucji Linuksa: Debian, Ubuntu, Mint, Manjaro, SuSe, Fedora, Mandriva, Slackware, Raspbian, TurboLinux ;-)

Etapy pracy nad programem

- Specyfikacja zadania

Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem

Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu

Etapy pracy nad programem

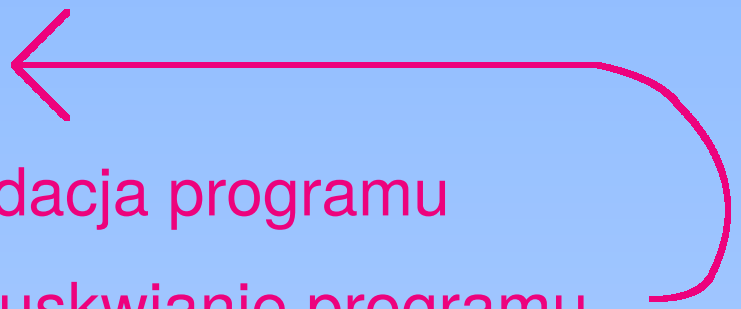
- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu

Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu

Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu

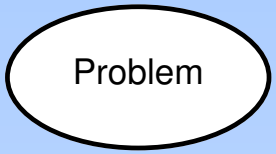


Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu
- Dokumentacja!!!

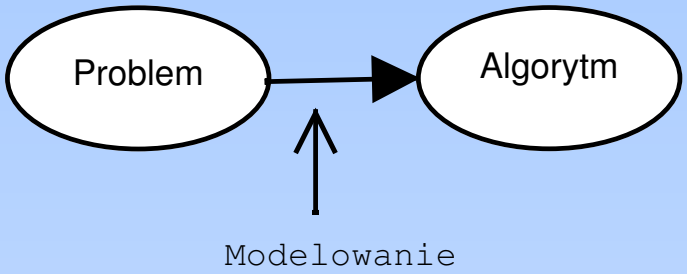
Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu
- Dokumentacja!!!



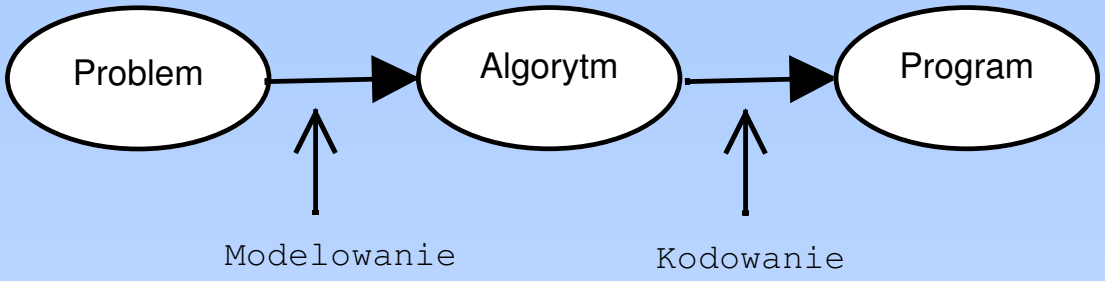
Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu
- Dokumentacja!!!



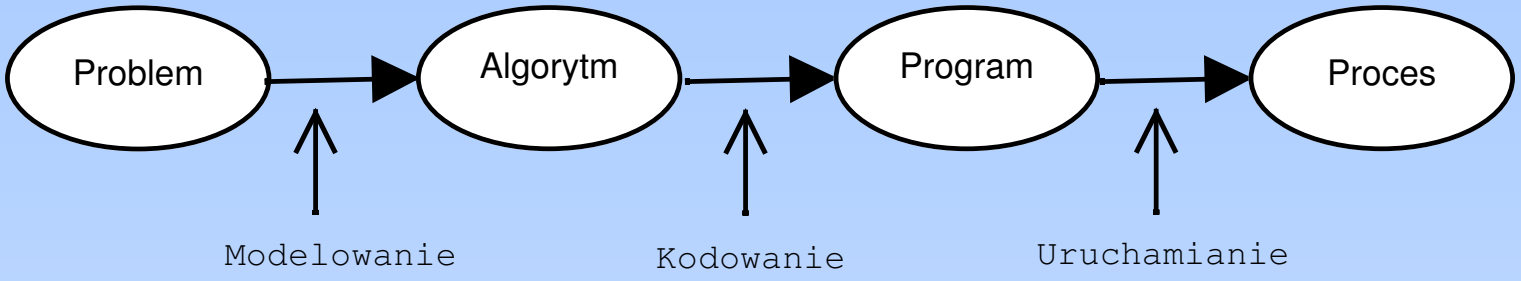
Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu
- Dokumentacja!!!



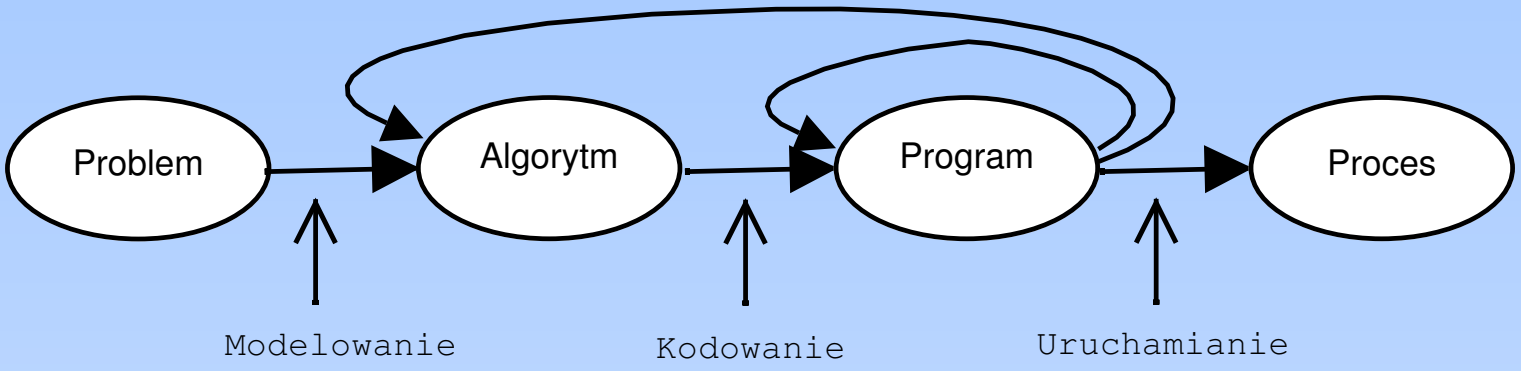
Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu
- Dokumentacja!!!

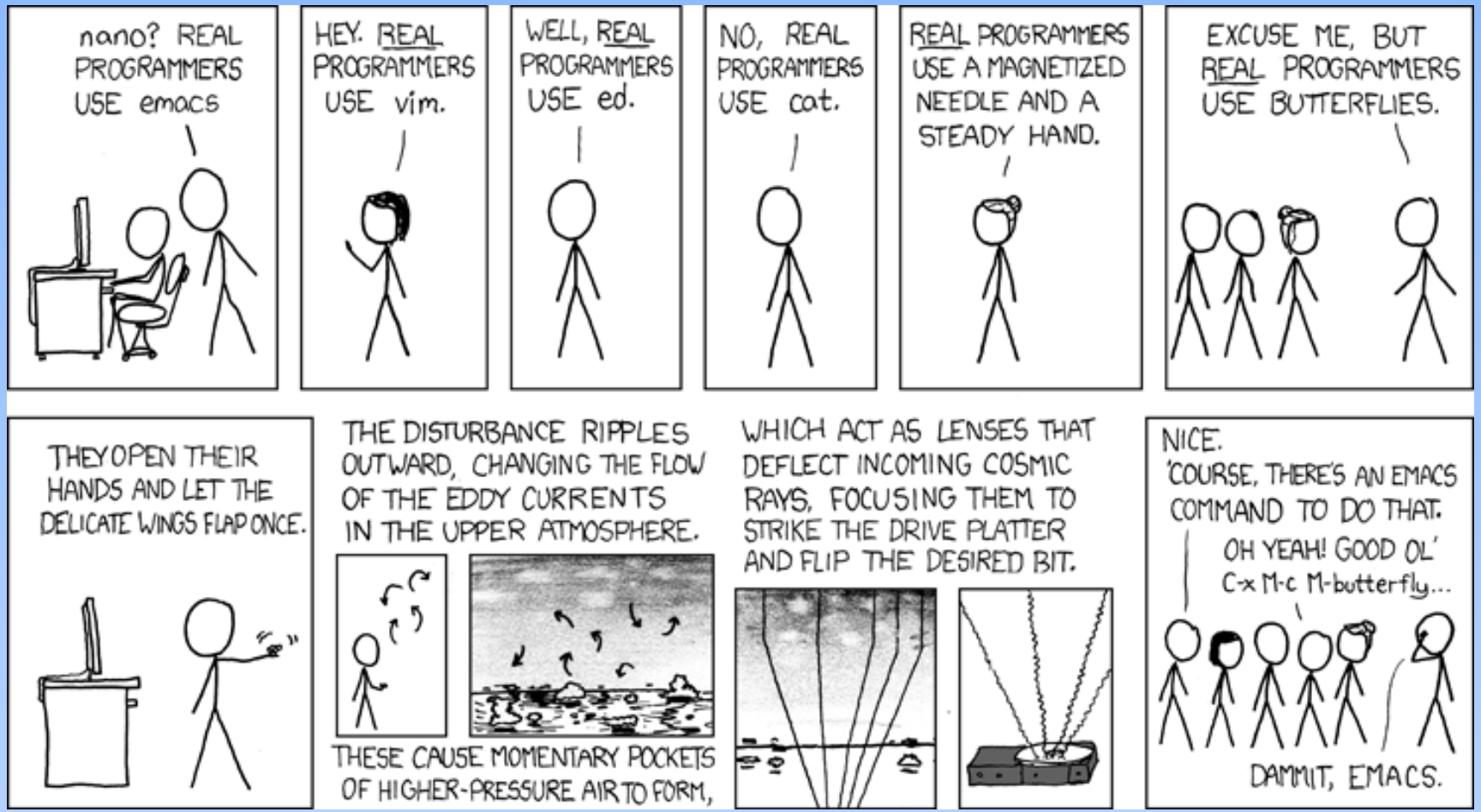


Etapy pracy nad programem

- Specyfikacja zadania
- Praca nad algorytmem
- Edycja programu
- Kompilacja i konsolidacja programu
- Uruchamianie i odpluskwianie programu
- Dokumentacja!!!



Edytor GNU Emacs (by Real Programmers)



Edytor GNU Emacs

Podstawowe komendy edytora GNU Emacs 94-9-1

Edytor GNU Emacs jest wywoływany komendą: `emacs nazwa_pliku`.

Przemieszczanie kursora

Ctrl-B, Ctrl-N, Ctrl-P, Ctrl-F — Przemieszczanie kursora o jeden znak (←↓↑→)

ESC F — Słowo do przodu
ESC B — Słowo do tyłu
ESC < — Kursor na początek pliku
ESC > — Kursor na koniec pliku
Ctrl-A — Kursor na początek linii
Ctrl-E — Kursor na koniec linii
Ctrl-V — Przejście do następnej strony
ESC V — Przejście do poprzedniej strony
ESC X goto-line — Przejście do linii o numerze podanym przez użytkownika

Kasowanie znaków

DEL — Kasowanie znaku przed kursorem
Ctrl-D — Kasowanie znaku na pozycji kursora
Ctrl-K — Kasowanie znaków od kursora do końca linii
ESC DEL — Kasowanie słowa przed kursorem
ESC D — Kasowanie słowa za kursorem

Poszukiwanie i zamiana ciągu znaków

Ctrl-S *ciąg_znaków* ESC — Przyrostowe poszukiwanie w przód
Ctrl-R *ciąg_znaków* ESC — Przyrostowe poszukiwanie w tył
ESC % *ciąg_znaków* RETURN *nowy_ciąg* RETURN — Warunkowa zamiana ciągu znaków:

Operacje na regionach (region - obszar między markerem a kursorem)

Ctrl-@ — Ustawienie markera (lub Ctrl-SPACJA)
Ctrl-W — Skasowanie regionu i zapis jego zawartości do bufora
ESC W — Zapis regionu do bufora bez jego kasowania
Ctrl-Y — Skopiowanie zawartości regionu w miejsce położenia kursora

Edytor GNU Emacs cd.

Operacje na oknach

Ctrl-X 2 — Otwarcie drugiego okna
Ctrl-X O — Przejście do drugiego okna
Ctrl-X 0 — Zamknięcie aktywnego okna
Ctrl-X 1 — Zamknięcie wszystkich okien oprócz aktywnego

Operacje na plikach i buforach plikowych

Ctrl-X Ctrl-S — Zapis bufora plikowego na dysk
Ctrl-X Ctrl-F — Otwarcie/utworzenie nowego pliku
Ctrl-X Ctrl-W — Utworzenie i zapis do pliku o nowej nazwie
Ctrl-X I — Wstawienie zawartości innego pliku do aktywnego bufora plikowego

Pomoc **Ctrl-H — Klawisz pomocy**

Ctrl-H T — wyświetla samouczek
ESC X describe-function — wyświetla opis funkcji o podanej nazwie (Ctrl-H F)
ESC X describe-key — wyświetla nazwę i opis funkcji przypisanej do klawisza (Ctrl-H K)
ESC X apropos — wyświetla nazwy funkcji ze słowem kluczowym (Ctrl-H A)

Inne ważne komendy **Ctrl-G — Przerwanie komendy**

Ctrl-L — Odświeżenie ekranu
Ctrl-X U — Odtworzenie zawartości bufora sprzed ostatniej zmiany
ESC ! — Uruchomienie shella w buforze edytora
Ctrl-X Ctrl-C — Zakończenie pracy z edytorem

Istnieje wiele samouczków edytora GNU Emacs w sieci www, na przykład tutaj jest jakiś.

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Komentarze

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji
  - Nagłówek funkcji

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji
  - ⋆ Deklaracja zmiennych

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                         /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji
  - ⋆ Część operacyjna

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji
  - Część operacyjna
    - Wczytanie danych

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji
  - ⋆ Część operacyjna
    - ∗ Kontrola poprawności danych

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji
  - ⋆ Część operacyjna
    - ∗ Część obliczeniowa

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Definicja funkcji
  - ⋆ Część operacyjna
    - ∗ Wyświetlenie wyników

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */
  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                        /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);
      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Dołaczenie prototypów funkcji bibliotecznych

Przykładowy program w C

```
/* kompilacja:
        Sun C: cc -Xc trojmian.c -lm
        GNU C: gcc -pedantic -Wall trojmian.c -lm */
#include <stdio.h>
#include <math.h>
int main()
{
  float a, b, c;                    /* parametry rownania */
  float delta, sqrtdelta, x1, x2;               /* wyniki */

  printf("Program rozwiazuje rownanie kwadratowe.\n");
  printf("Podaj wspolczynnik a, b, c:\n");
  scanf("%f%f%f", &a, &b, &c);
  if (a == 0.0)           /* przypadek rownania liniowego */
    printf("To nie jest rownanie kwadratowe.\n");
  else {
    delta  = (b*b) - (4.0*a*c);
    if (delta < 0.0)      /* kontrola istnienia rozwiazan */
      printf("Brak rozwiazan rzeczywistych.\n");
    else {                         /* rozwiazanie rownania */
      sqrtdelta = (float)sqrt( (double)delta );
      x1 = (-b - sqrtdelta) / (2*a);
      x2 = (-b + sqrtdelta) / (2*a);

      printf("Rozwiazaniem rownania sa pierwiastki:\n");
      printf("  x1 = %f\n  x2 = %f\n", x1, x2);
    }
  }
  return 0;
}
```

- Komentarze
- Dołaczenie prototypów funkcji bibliotecznych
- Definicja funkcji
  - Nagłówek funkcji
  - Deklaracja zmiennych
  - Część operacyjna
    - Wczytanie danych
    - Kontrola poprawności danych
    - Część obliczeniowa
    - Wyświetlenie wyników

Diagram algorytmu

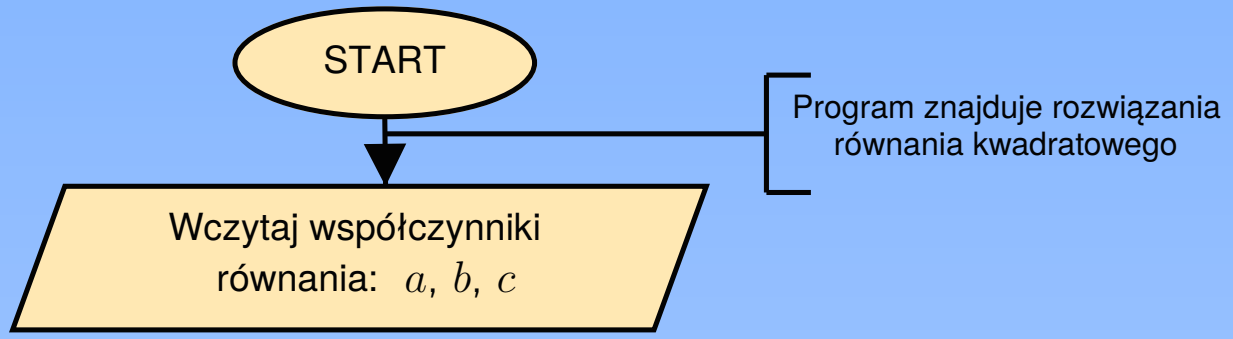


Diagram algorytmu

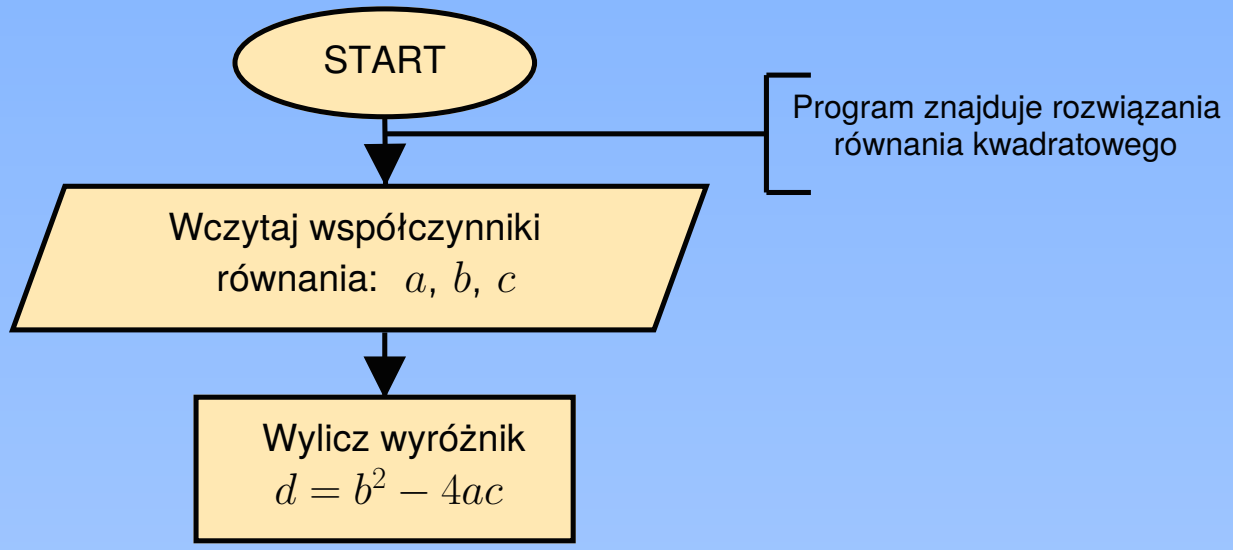


Diagram algorytmu

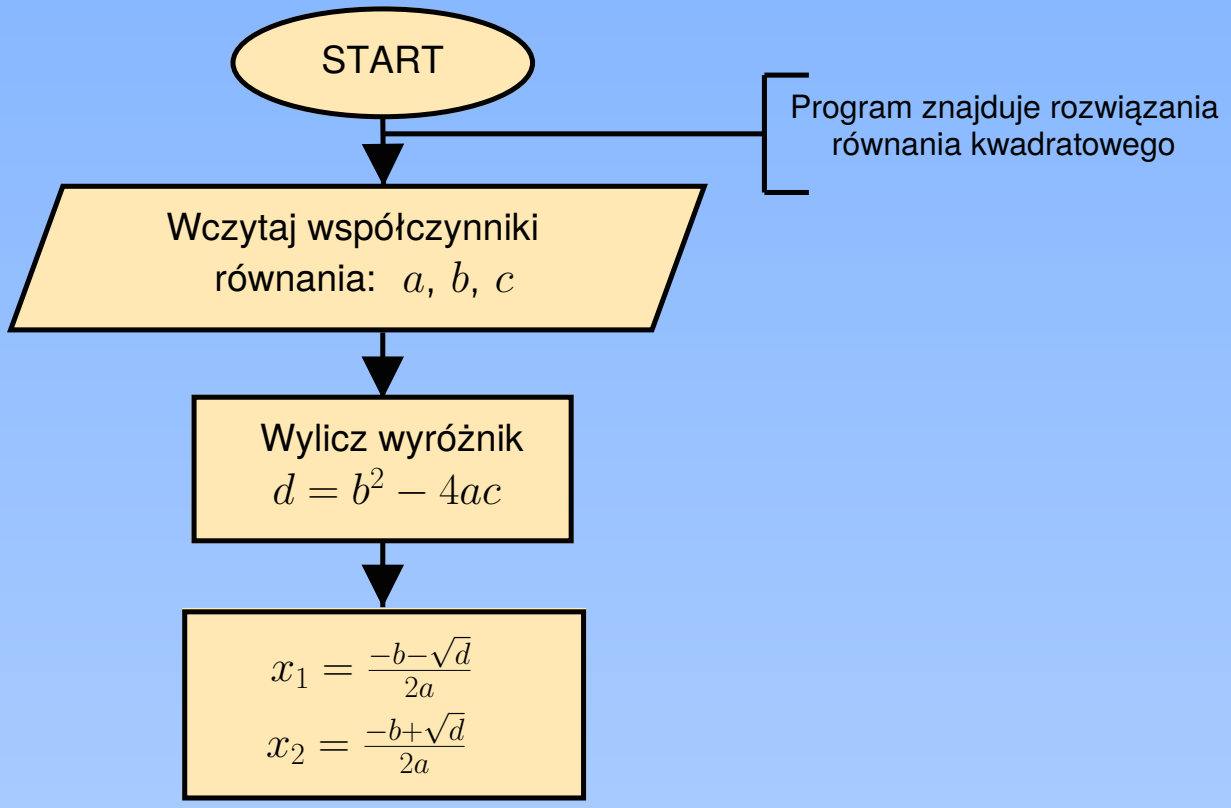
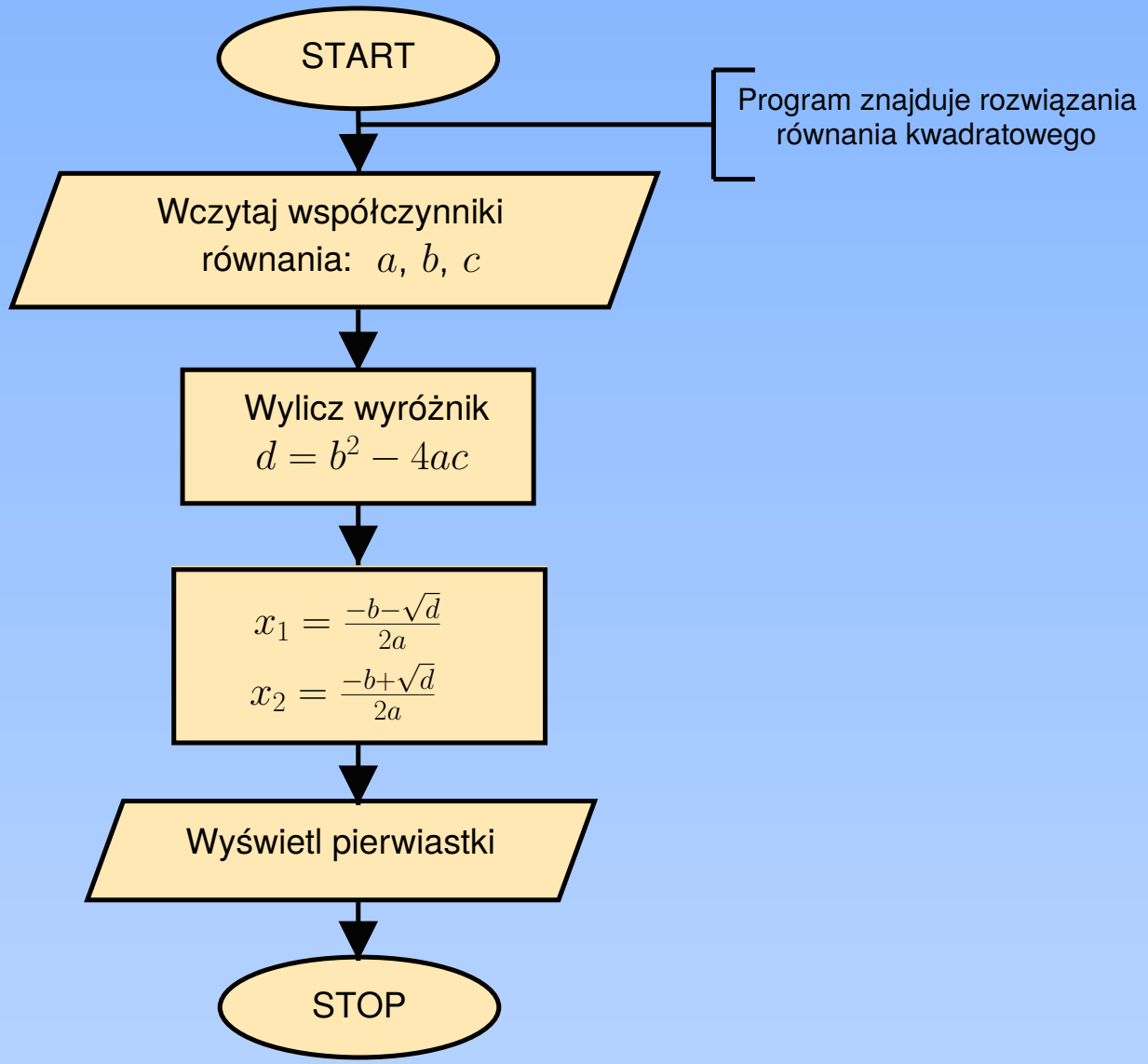
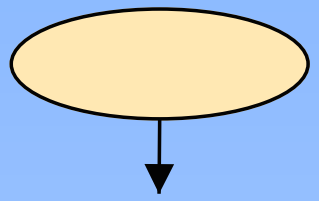


Diagram algorytmu



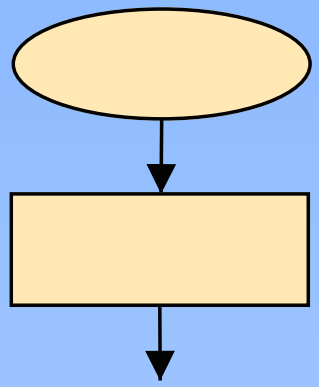
Diagramy algorytmów (schematy blokowe)



blok terminatora

Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)

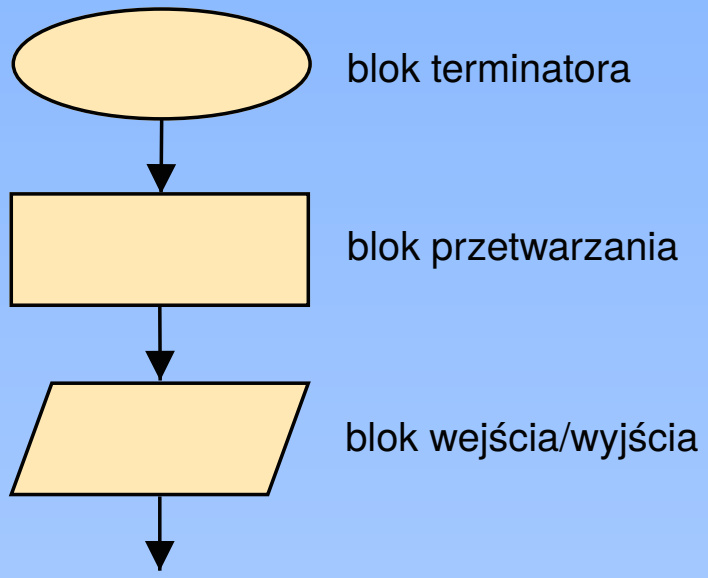


blok terminatora

blok przetwarzania

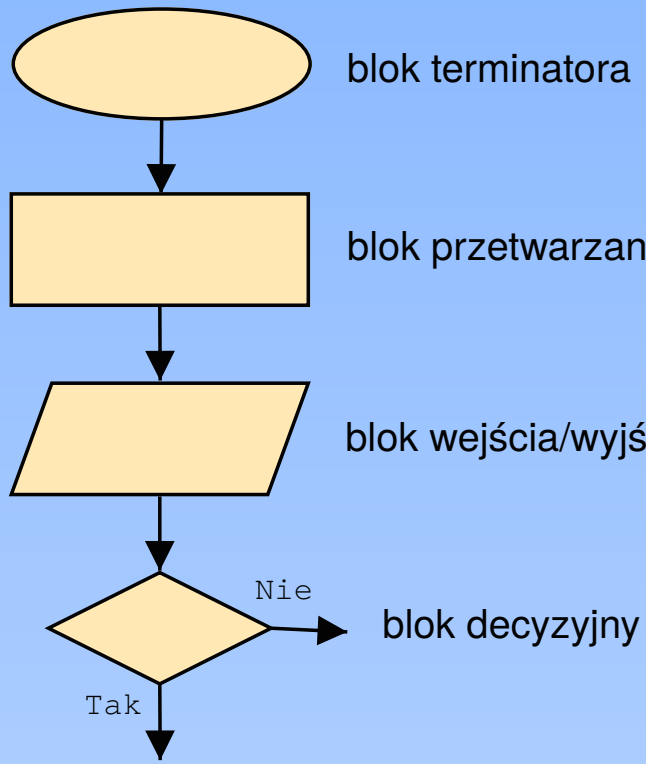
Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)



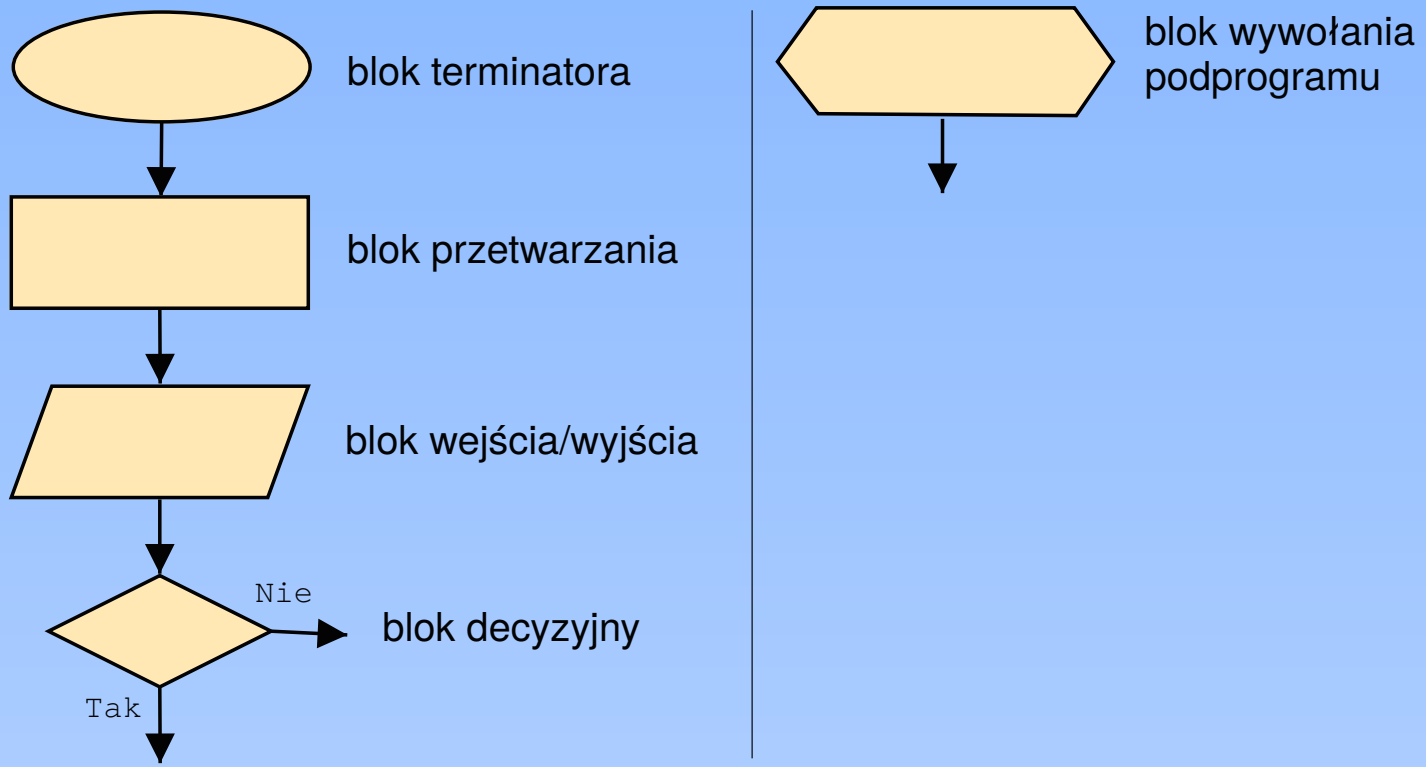
Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)



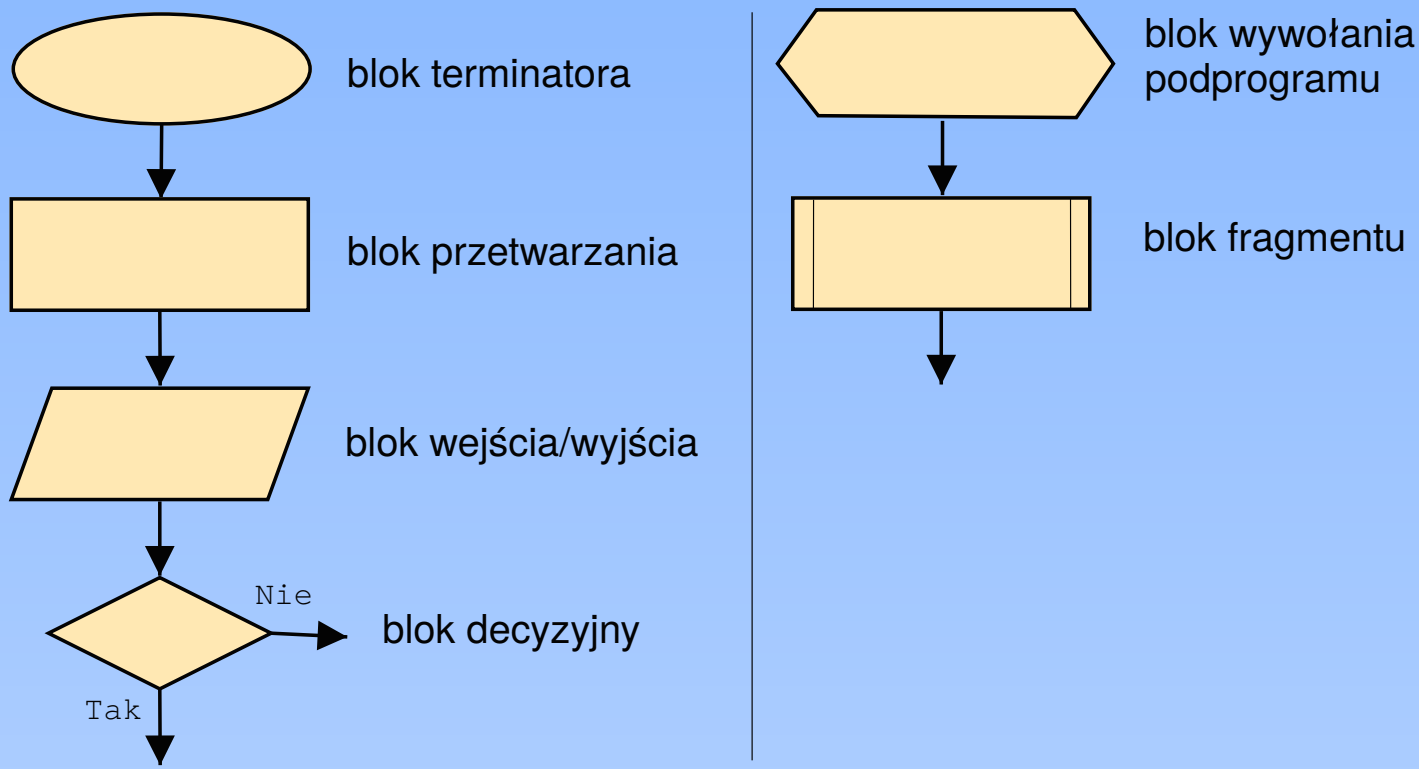
Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)



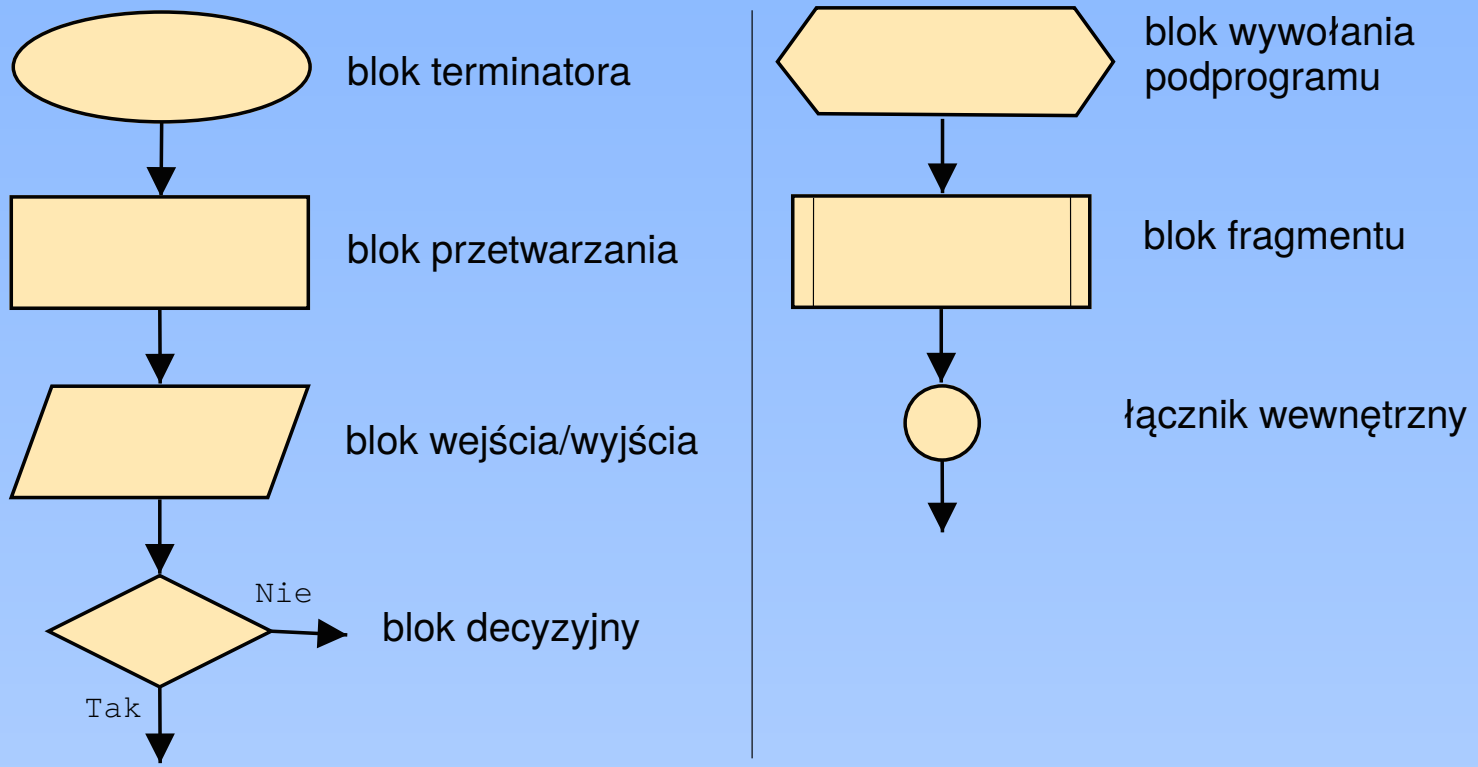
Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)



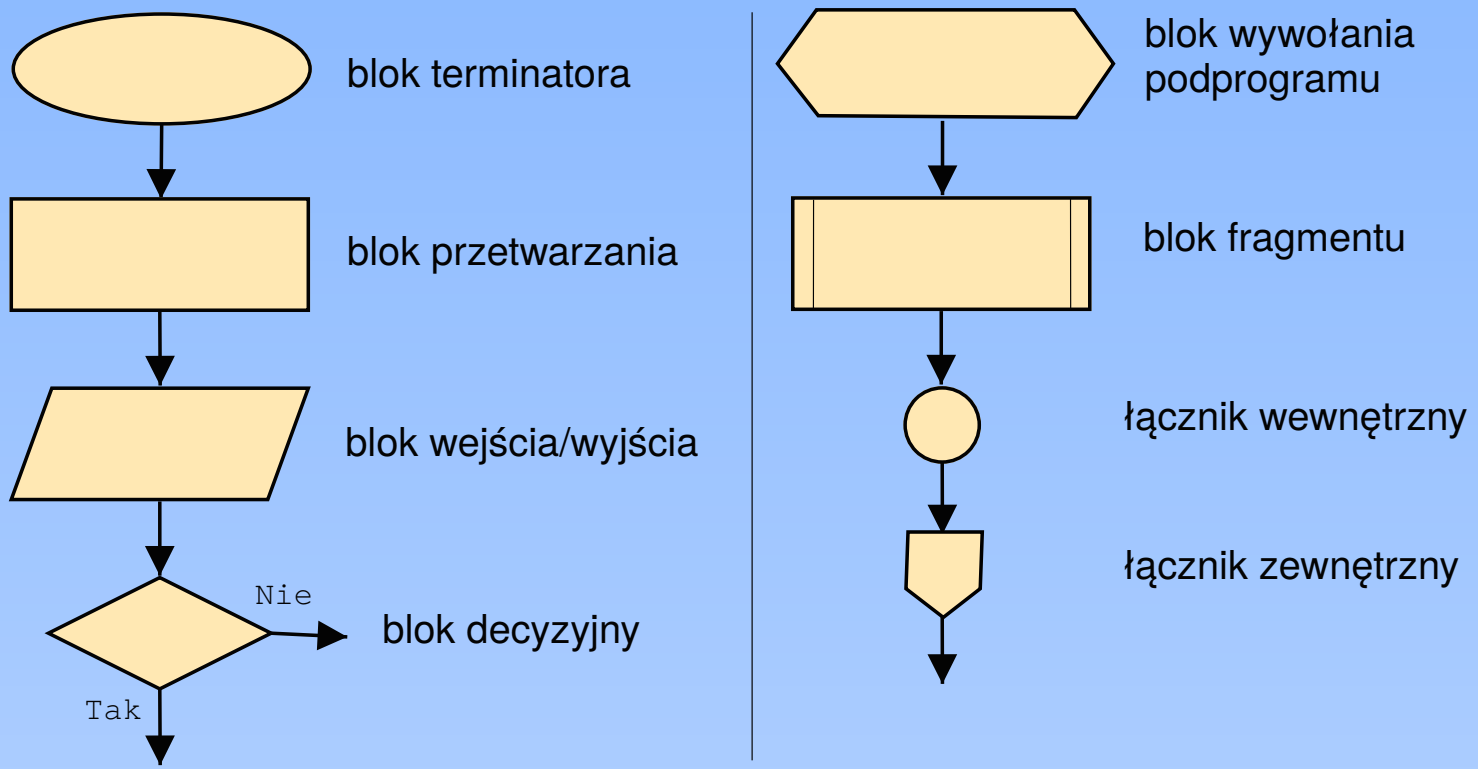
Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)



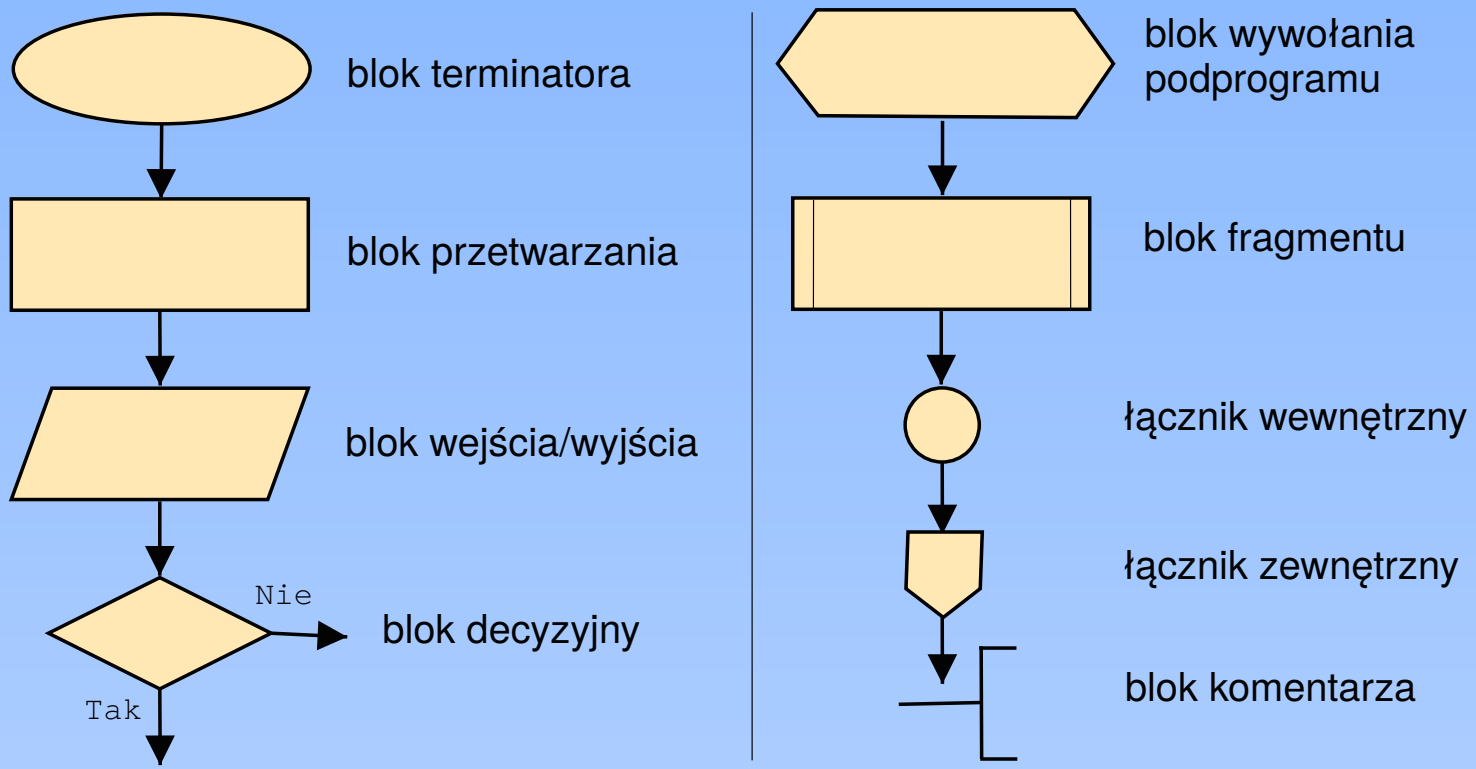
Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)



Elementy składowe schematów blokowych algorytmów.

Diagramy algorytmów (schematy blokowe)



Elementy składowe schematów blokowych algorytmów.

Diagramy czynności – język UML

Czynność (Activity)

Wybrane elementy składowe diagramów czynności UML.

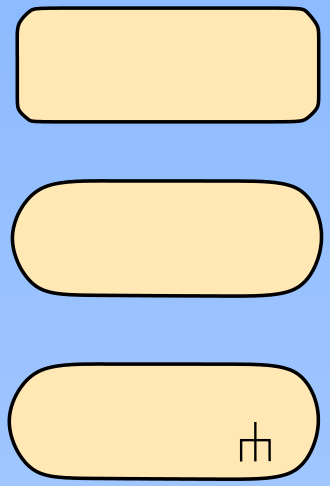
Diagramy czynności – język UML

Czynność (Activity)

Akcja (Action)

Wybrane elementy składowe diagramów czynności UML.

Diagramy czynności – język UML



Wybrane elementy składowe diagramów czynności UML.

Diagramy czynności – język UML

Czynność (Activity)

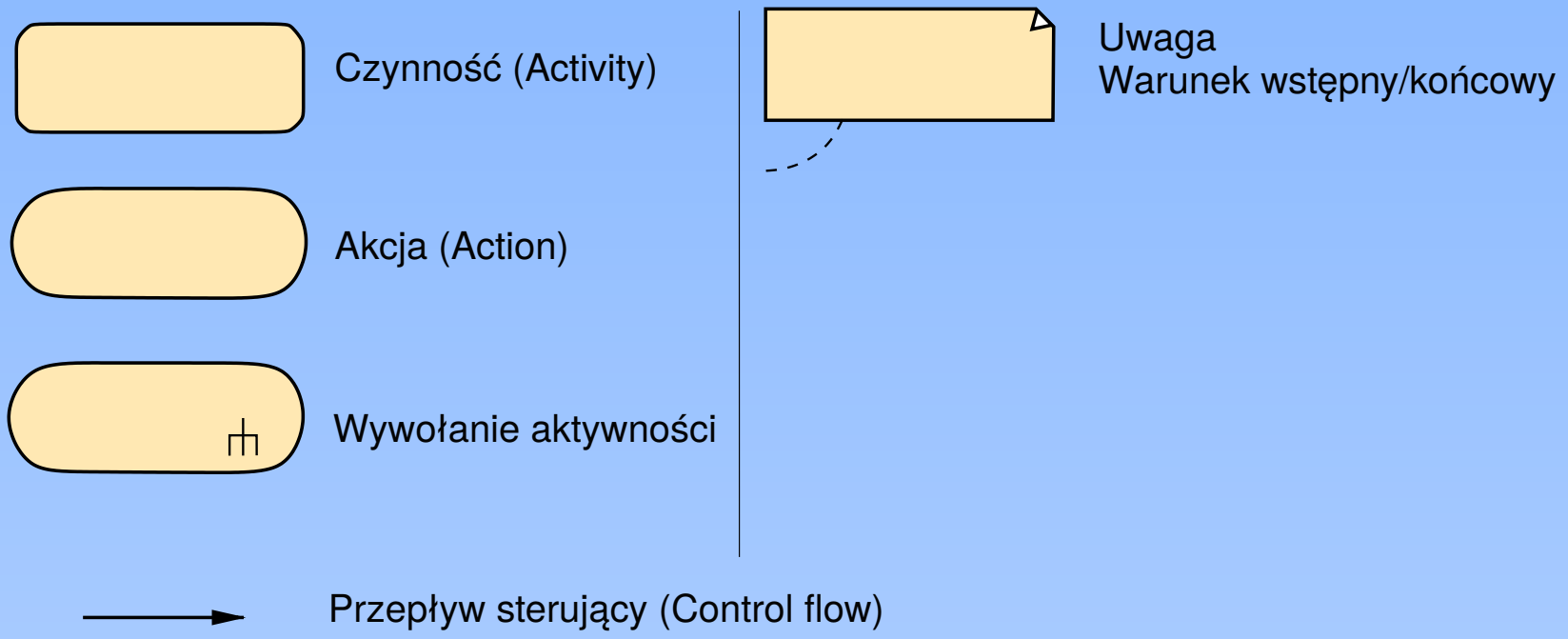
Akcja (Action)

Wywołanie aktywności

Przepływ sterujący (Control flow)

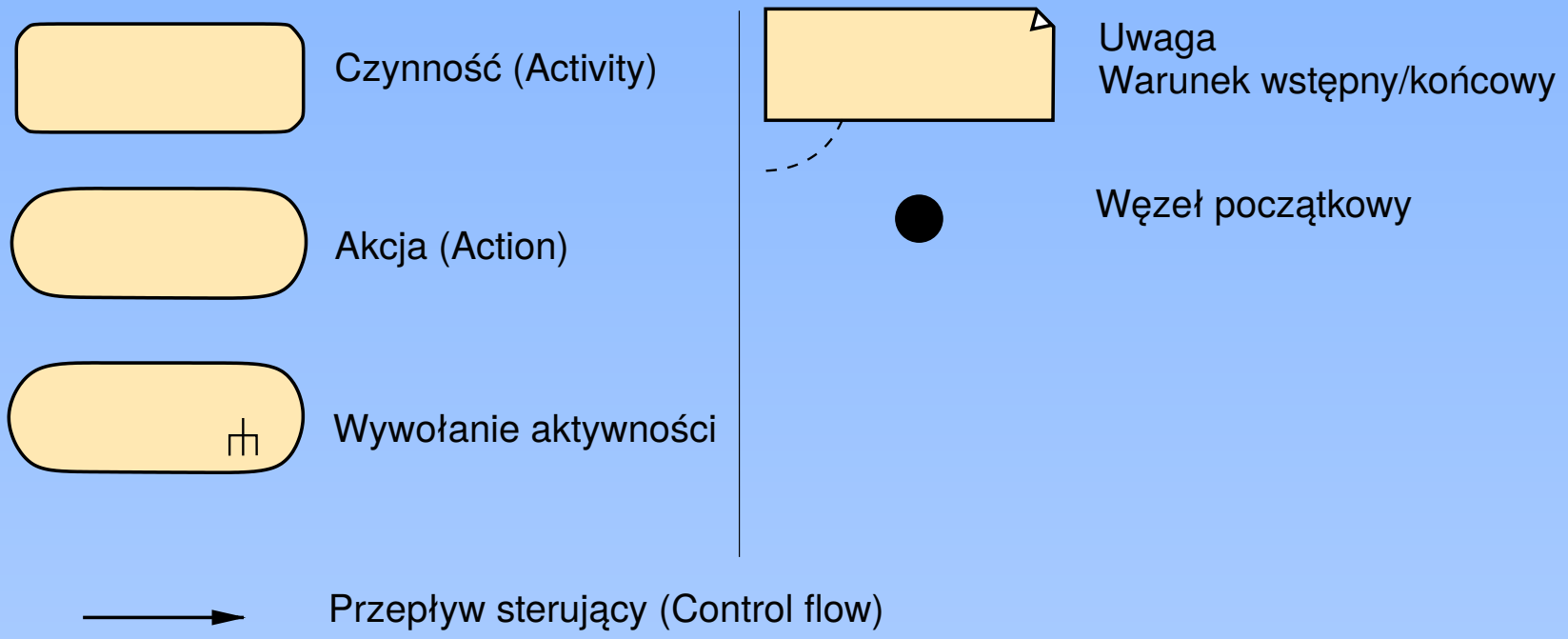
Wybrane elementy składowe diagramów czynności UML.

Diagramy czynności – język UML



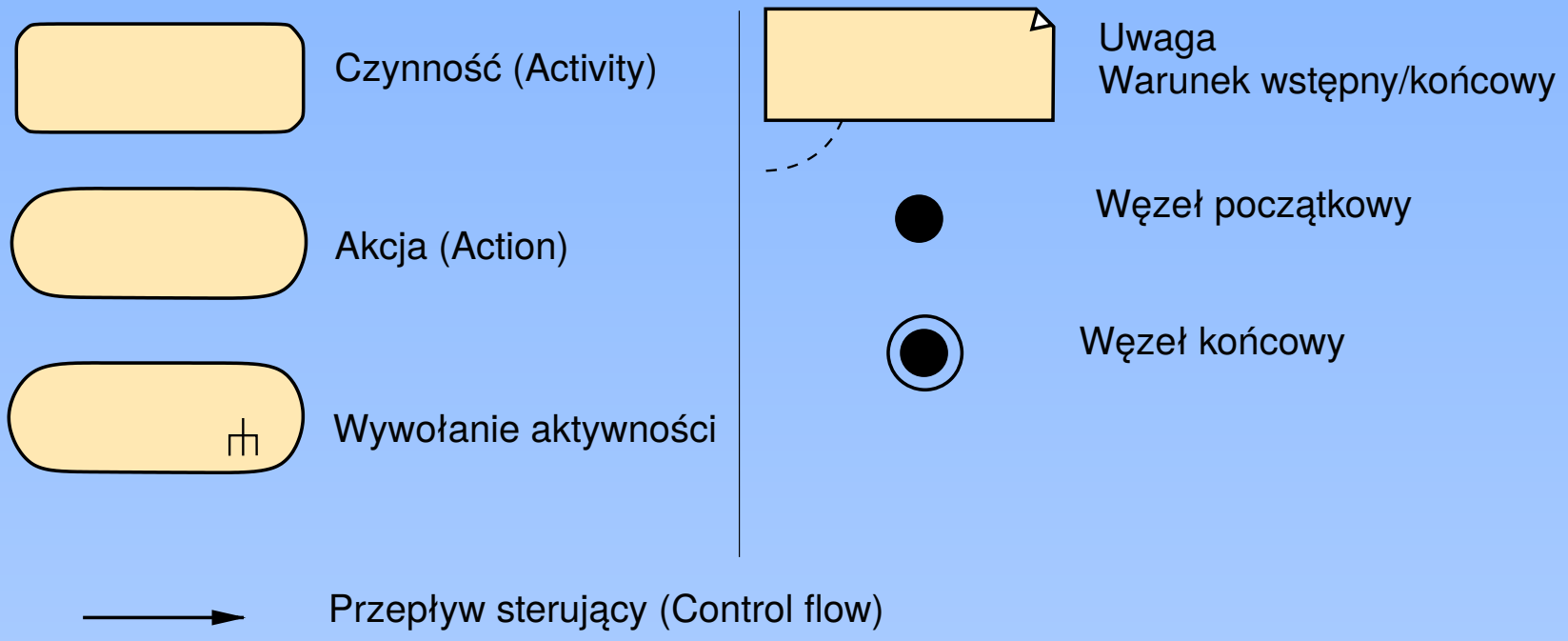
Wybrane elementy składowe diagramów czynności UML.

Diagramy czynności – język UML



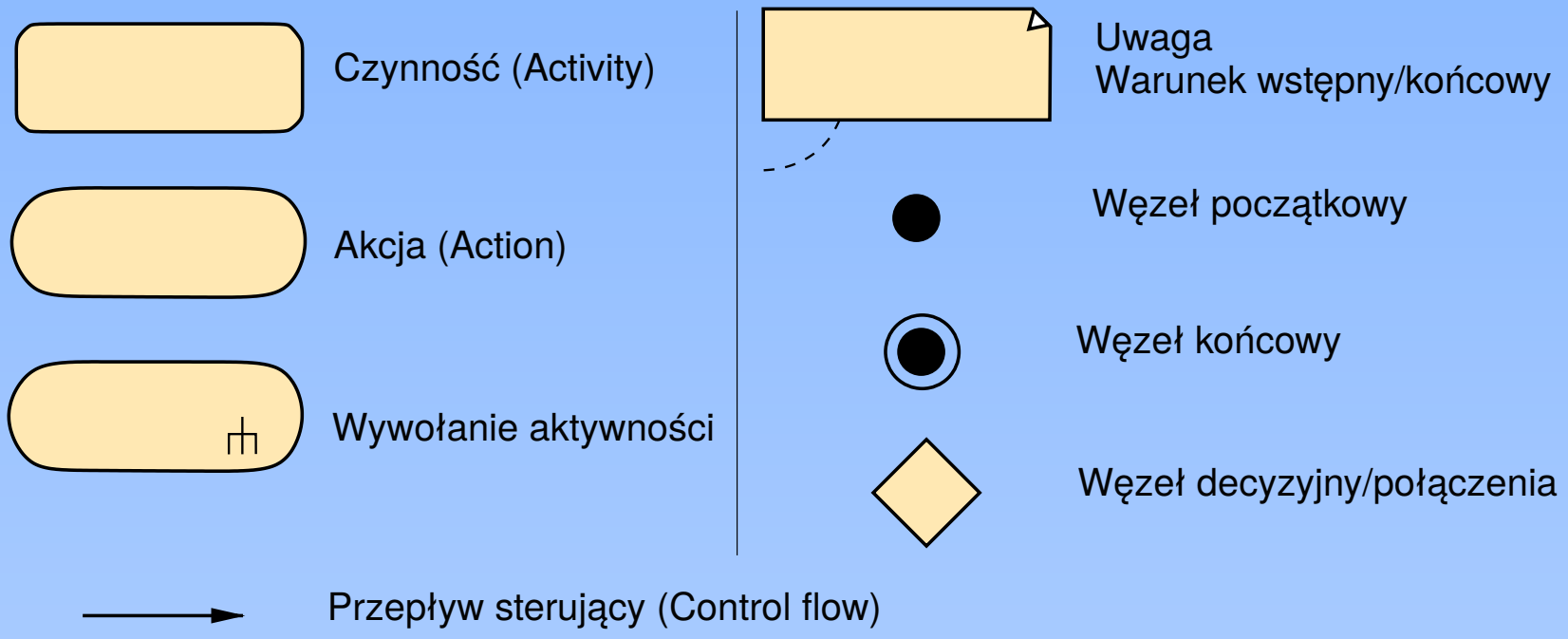
Wybrane elementy składowe diagramów czynności UML.

Diagramy czynności – język UML



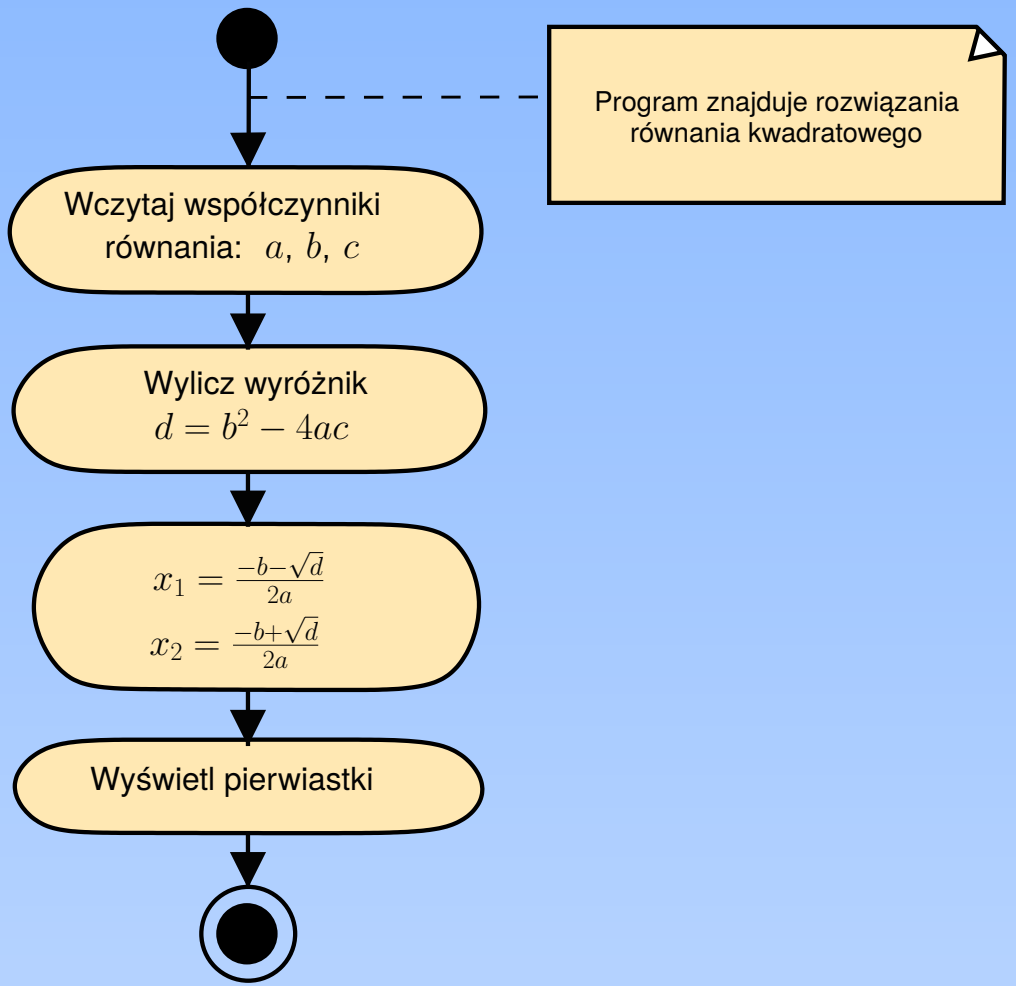
Wybrane elementy składowe diagramów czynności UML.

Diagramy czynności – język UML

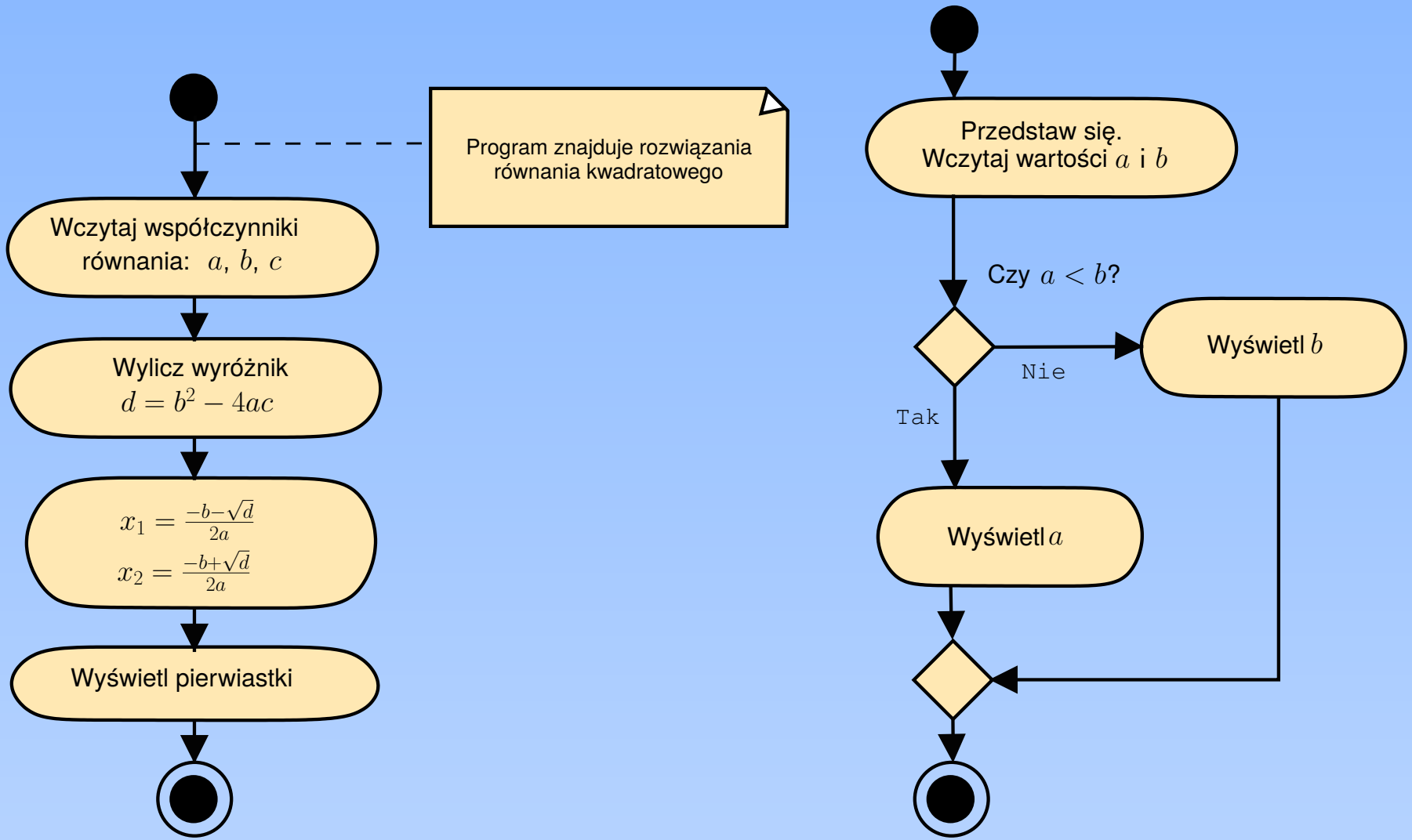


Wybrane elementy składowe diagramów czynności UML.

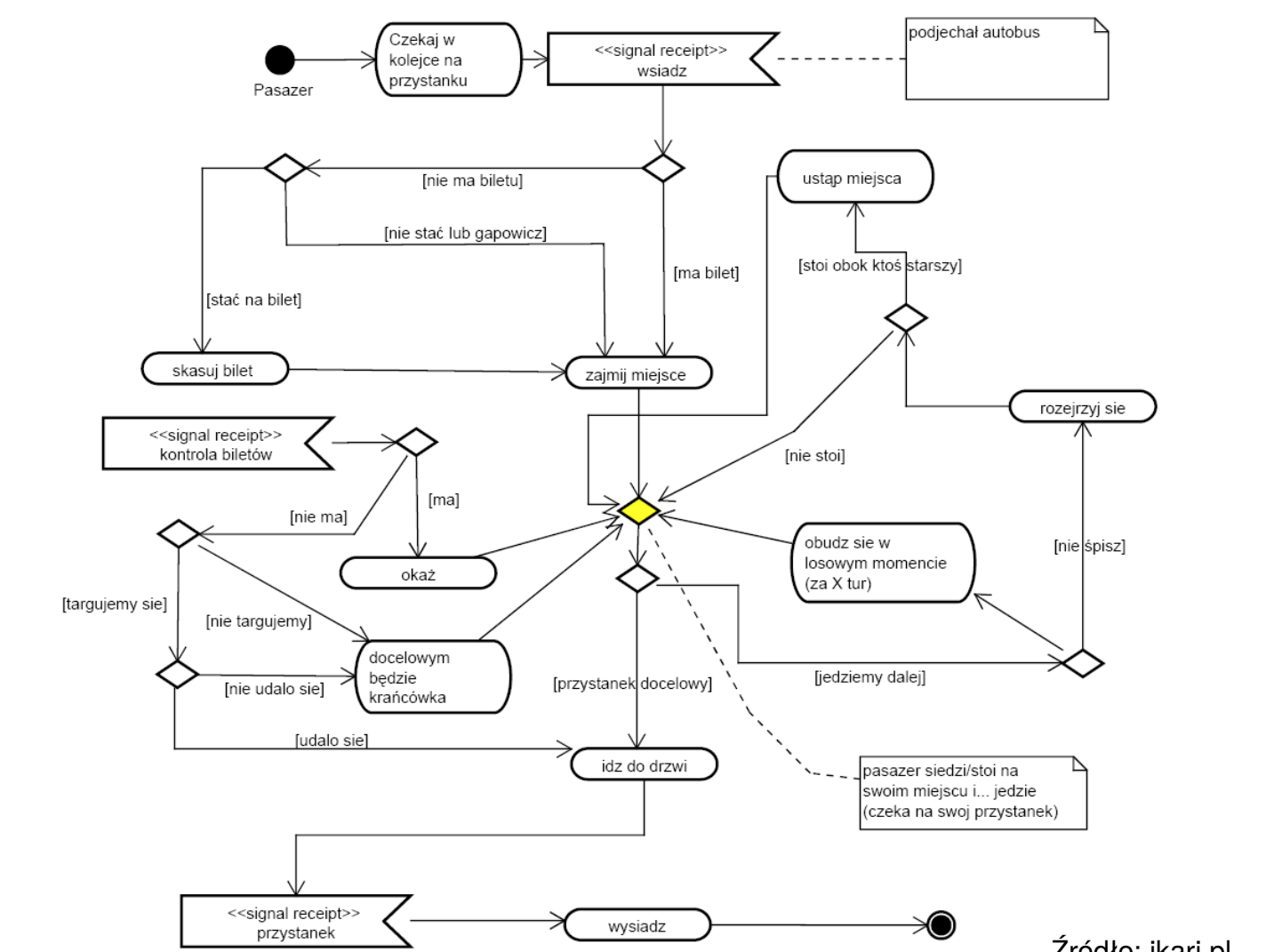
Diagramy czynności – przykłady



Diagramy czynności – przykłady

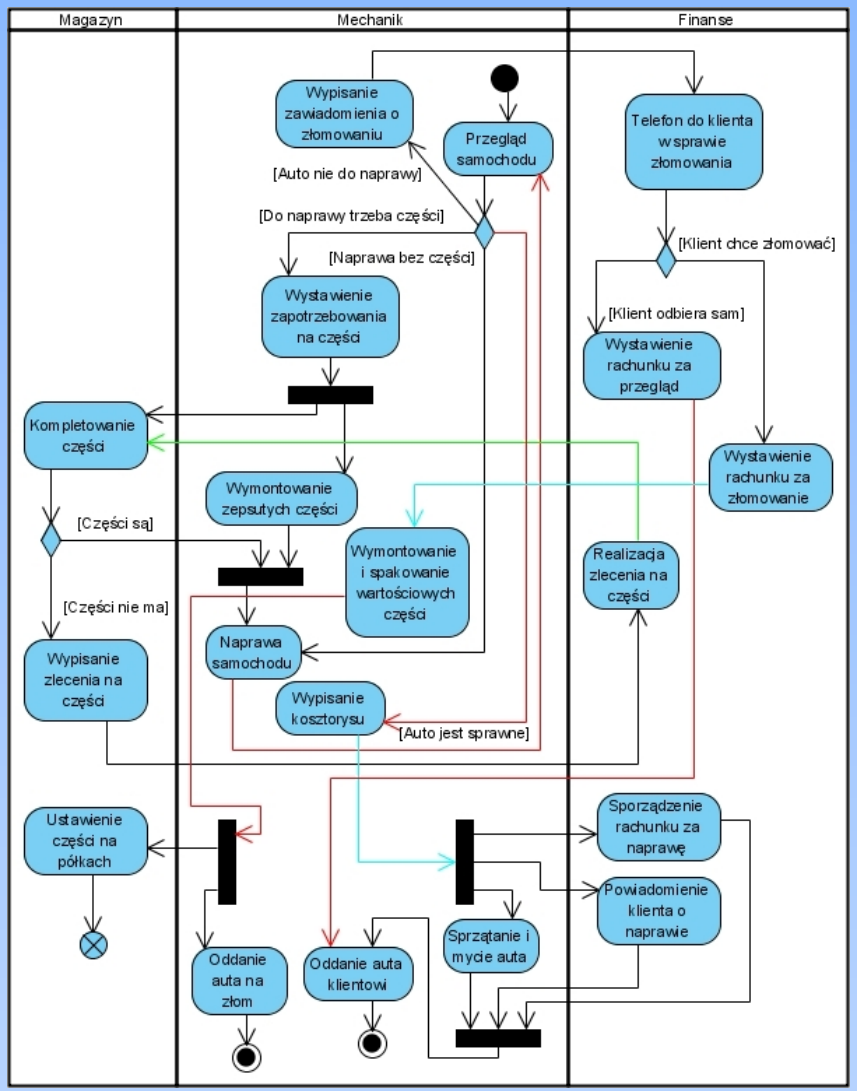


Diagramy czynności – przykłady



Źródło: ikari.pl

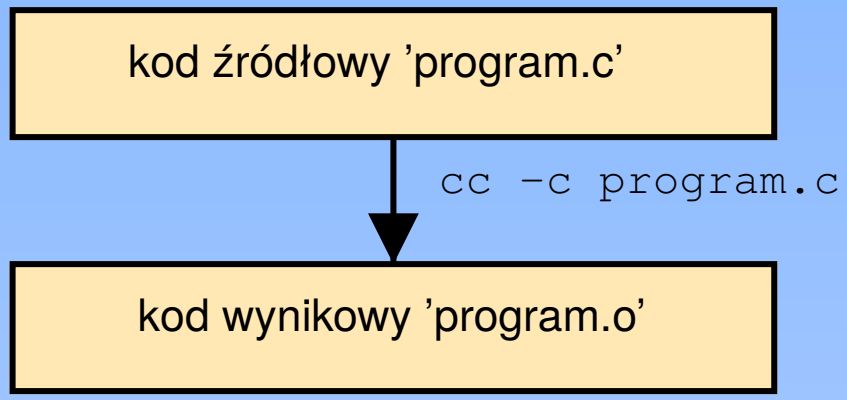
Diagramy czynności – przykłady



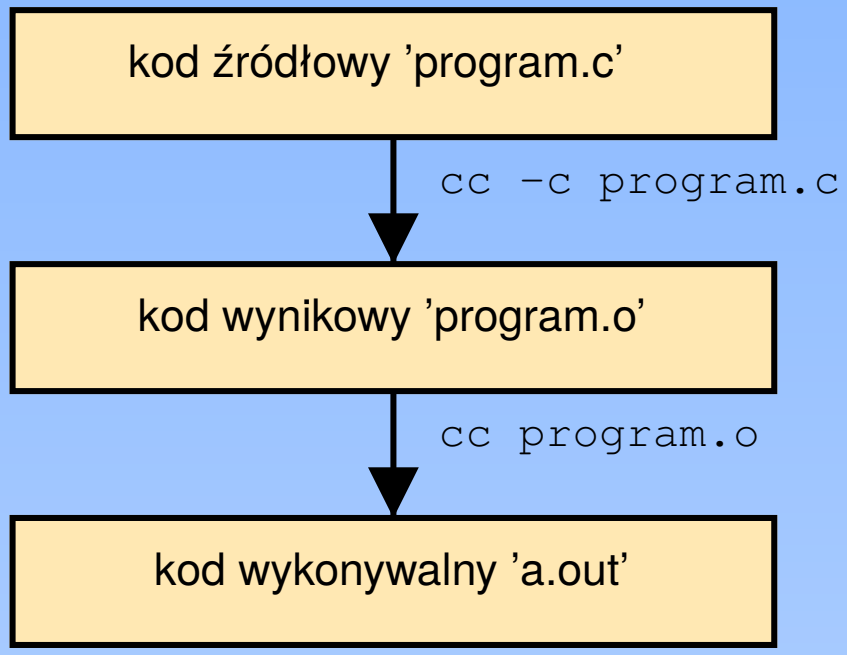
Źródło: A. Szczepański, AGH

Kompilacja i konsolidacja programów

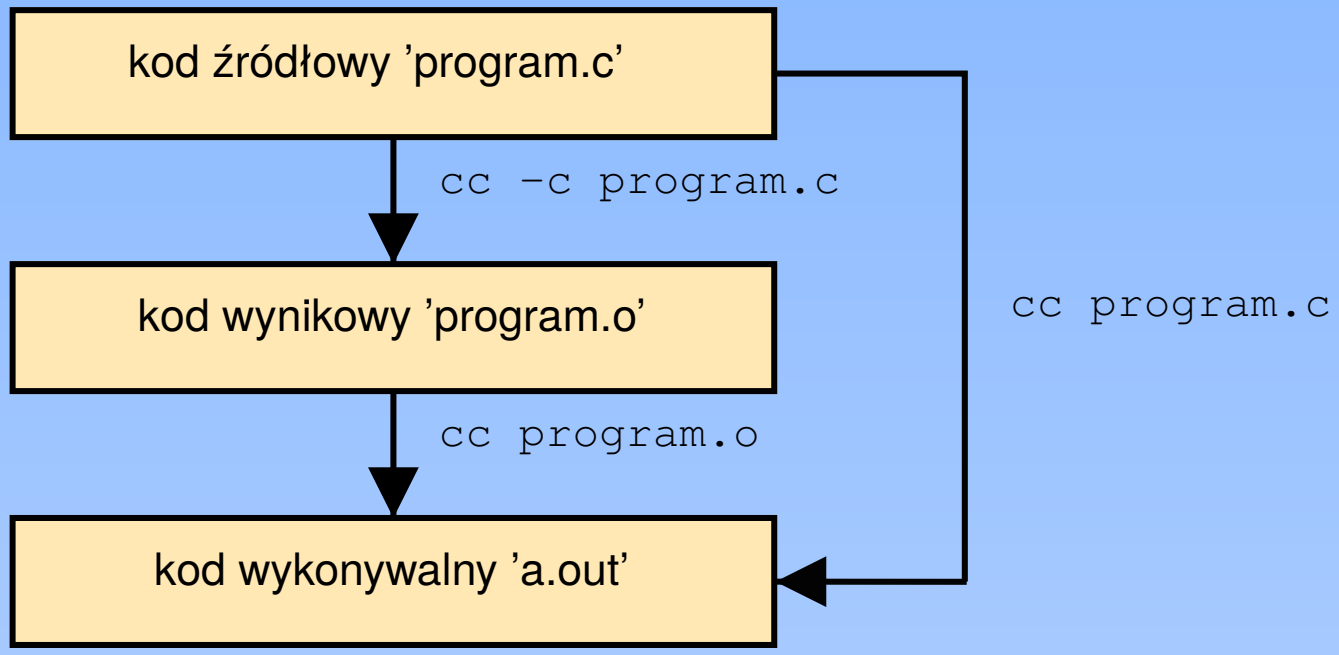
Kompilacja i konsolidacja programów



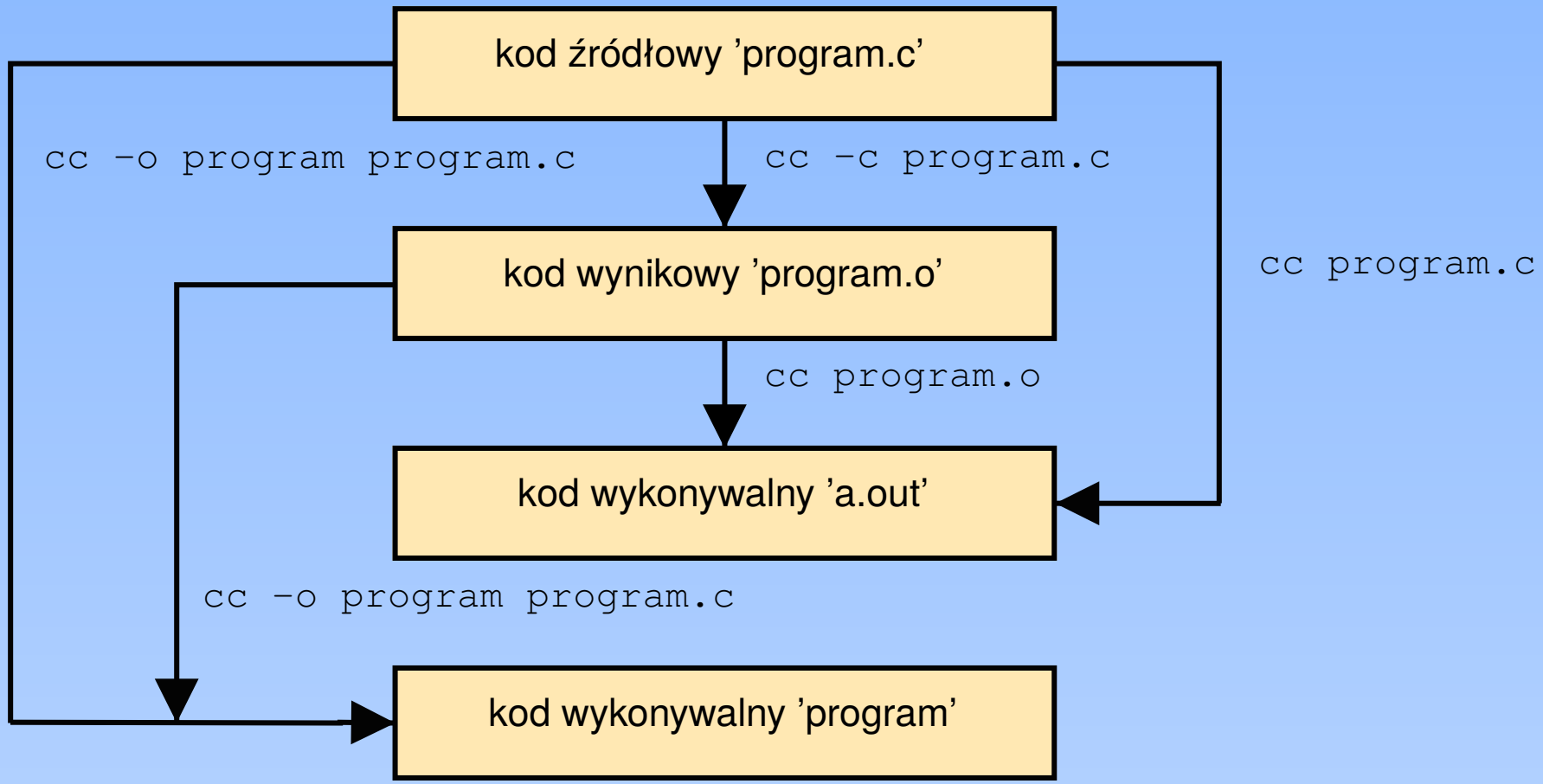
Kompilacja i konsolidacja programów



Kompilacja i konsolidacja programów



Kompilacja i konsolidacja programów



Opcje wywołania kompilatorów (wspólne)

Tradycyjnie, kompilatory C (`cc`, `gcc`) jednakowo rozpoznają opcje:

-onazwa — umieść postać wykonywalną kompilacji w pliku nazwa (domyślnie `a.out`)

Opcje wywołania kompilatorów (wspólne)

Tradycyjnie, kompilatory C (`cc`, `gcc`) jednakowo rozpoznają opcje:

-onazwa — umieść postać wykonywalną kompilacji w pliku nazwa (domyślnie `a.out`)

-c — pomiń ostatnią fazę przetwarzania (linkowanie), nie twórz programu wykonywalnego, pozostaw postać wynikową `.o`

Opcje wywołania kompilatorów (wspólne)

Tradycyjnie, kompilatory C (`cc`, `gcc`) jednakowo rozpoznają opcje:

-onazwa — umieść postać wykonywalną kompilacji w pliku nazwa (domyślnie `a.out`)

-c — pomiń ostatnią fazę przetwarzania (linkowanie), nie twórz programu wykonywalnego, pozostaw postać wynikową `.o`

-g — wpisz w program wykonywalny dodatkowe informacje dla debuggera

Opcje wywołania kompilatorów (wspólne)

Tradycyjnie, kompilatory C (`cc`, `gcc`) jednakowo rozpoznają opcje:

-onazwa — umieść postać wykonywalną kompilacji w pliku nazwa (domyślnie `a.out`)

-c — pomiń ostatnią fazę przetwarzania (linkowanie), nie twórz programu wykonywalnego, pozostaw postać wynikową `.o`

-g — wpisz w program wykonywalny dodatkowe informacje dla debuggera

-lbib — przeglądaj przez linker biblioteki `bib` (w kartotece `/usr/lib` lub innych, zdefiniowanych ścieżką linkera)

Opcje wywołania kompilatorów (wspólne)

Tradycyjnie, kompilatory C (`cc`, `gcc`) jednakowo rozpoznają opcje:

-onazwa — umieść postać wykonywalną kompilacji w pliku nazwa (domyślnie `a.out`)

-c — pomiń ostatnią fazę przetwarzania (linkowanie), nie twórz programu wykonywalnego, pozostaw postać wynikową `.o`

-g — wpisz w program wykonywalny dodatkowe informacje dla debuggera

-lbib — przeglądaj przez linker biblioteki `bib` (w kartotece `/usr/lib` lub innych, zdefiniowanych ścieżką linkera)

-On — wykonaj optymalizację kodu poziomu `n` (domyślnie poziom 2, na ogół bezpieczny)

Opcje wywołania kompilatorów (wspólne)

Tradycyjnie, kompilatory C (`cc`, `gcc`) jednakowo rozpoznają opcje:

-onazwa — umieść postać wykonywalną kompilacji w pliku nazwa (domyślnie `a.out`)

-c — pomiń ostatnią fazę przetwarzania (linkowanie), nie twórz programu wykonywalnego, pozostaw postać wynikową `.o`

-g — wpisz w program wykonywalny dodatkowe informacje dla debuggera

-lbib — przeglądaj przez linker biblioteki `bib` (w kartotece `/usr/lib` lub innych, zdefiniowanych ścieżką linkera)

-On — wykonaj optymalizację kodu poziomu `n` (domyślnie poziom 2, na ogół bezpieczny)

-w — pomiń ostrzeżenia (opcja zwykle szkodliwa!)

Opcje wywołania kompilatorów (różne)

Niektóre ważne i pożyteczne opcje występują tylko dla niektórych kompilatorów, lub mają inną postać:

-V — wyświetlaj wywołania kolejnych faz kompilacji (Sun cc)

-v — wyświetlaj wywołania kolejnych faz kompilacji (GNU cc, HP gcc)

Opcje wywołania kompilatorów (różne)

Niektóre ważne i pożyteczne opcje występują tylko dla niektórych kompilatorów, lub mają inną postać:

-V — wyświetlaj wywołania kolejnych faz kompilacji (Sun cc)

-v — wyświetlaj wywołania kolejnych faz kompilacji (GNU cc, HP gcc)

-xc99 — przestrzegaj standardu ANSI C99 (Sun cc)

-std=c99 — przestrzegaj standardu ANSI C99 (bez rozszerzeń GNU) (GNU gcc)

-xc99=none — przestrzegaj standardu ANSI C90 (Sun cc)

-ansi — przestrzegaj standardu ANSI C90 (GNU gcc)

Opcje wywołania kompilatorów (różne)

Niektóre ważne i pożyteczne opcje występują tylko dla niektórych kompilatorów, lub mają inną postać:

-V — wyświetlaj wywołania kolejnych faz kompilacji (Sun cc)

-v — wyświetlaj wywołania kolejnych faz kompilacji (GNU cc, HP gcc)

-xc99 — przestrzegaj standardu ANSI C99 (Sun cc)

-std=c99 — przestrzegaj standardu ANSI C99 (bez rozszerzeń GNU) (GNU gcc)

-xc99=none — przestrzegaj standardu ANSI C90 (Sun cc)

-ansi — przestrzegaj standardu ANSI C90 (GNU gcc)

-Xc — ściśle przestrzegaj standardu ANSI C (bez rozszerzeń K&R) (Sun cc)

-Aa — ściśle przestrzegaj standardu ANSI C (HP cc)

Opcje wywołania kompilatorów (różne)

Niektóre ważne i pożyteczne opcje występują tylko dla niektórych kompilatorów, lub mają inną postać:

-V — wyświetlaj wywołania kolejnych faz kompilacji (Sun cc)

-v — wyświetlaj wywołania kolejnych faz kompilacji (GNU cc, HP gcc)

-xc99 — przestrzegaj standardu ANSI C99 (Sun cc)

-std=c99 — przestrzegaj standardu ANSI C99 (bez rozszerzeń GNU) (GNU gcc)

-xc99=none — przestrzegaj standardu ANSI C90 (Sun cc)

-ansi — przestrzegaj standardu ANSI C90 (GNU gcc)

-Xc — ściśle przestrzegaj standardu ANSI C (bez rozszerzeń K&R) (Sun cc)

-Aa — ściśle przestrzegaj standardu ANSI C (HP cc)

-pedantic — wyświetlaj ostrzeżenia wymagane przez ANSI C (GNU gcc)

-Wall — wyświetlaj wszystkie ostrzeżenia o „dziwnych” konstrukcjach programowych (GNU cc)

Opcje wywołania kompilatorów (zalecane)

- kompilator Sun C (komputer diablo)

  -xc99 — przestrzegaj standardu ANSI C99 (domyślnie – zależnie od wersji kompilatora)

  -Xc — ściśle przestrzegaj standardu ANSI C bez rozszerzeń K&R (domyślnie – z rozszerzeniami K&R)

Opcje wywołania kompilatorów (zalecane)

- kompilator Sun C (komputer diablo)

  -xc99 — przestrzegaj standardu ANSI C99 (domyślnie – zależnie od wersji kompilatora)

  -Xc — ściśle przestrzegaj standardu ANSI C bez rozszerzeń K&R (domyślnie – z rozszerzeniami K&R)

- kompilator GNU C (komputer diablo, panamint)

  -std=c99 — przestrzegaj standardu ANSI C99 (bez rozszerzeń GNU) (domyślnie – standard zależnie od wersji kompilatora, zawsze z rozszerzeniami GNU)

  -pedantic — wyświetlaj ostrzeżenia wymagane przez standard ANSI C

  -Wall — wyświetlaj wszystkie ostrzeżenia o „dziwnych” konstrukcjach programowych

Opcje wywołania kompilatorów (zalecane)

- kompilator Sun C (komputer diablo)

  -xc99 — przestrzegaj standardu ANSI C99 (domyślnie – zależnie od wersji kompilatora)

  -Xc — ściśle przestrzegaj standardu ANSI C bez rozszerzeń K&R (domyślnie – z rozszerzeniami K&R)

- kompilator GNU C (komputer diablo, panamint)

  -std=c99 — przestrzegaj standardu ANSI C99 (bez rozszerzeń GNU) (domyślnie – standard zależnie od wersji kompilatora, zawsze z rozszerzeniami GNU)

  -pedantic — wyświetlaj ostrzeżenia wymagane przez standard ANSI C

  -Wall — wyświetlaj wszystkie ostrzeżenia o „dziwnych” konstrukcjach programowych

- lista dostępnych opcji
  - ⋆ kompilator Sun C: cc -flags
  - ⋆ kompilator GNU C: gcc --help=common

Kompilacja programów – wywołania

- kompilator Sun C (komputer diablo)

```
cc -xc99 -Xc prog.c          ⟸  wynik: program binarny a.out
cc -xc99 -Xc prog.c -lm      ⟸  wynik: program binarny a.out
                                dołączona biblioteka matematyczna
```

Kompilacja programów – wywołania

- kompilator Sun C (komputer diablo)

```
cc -xc99 -Xc prog.c          ⟸  wynik: program binarny a.out
cc -xc99 -Xc prog.c -lm      ⟸  wynik: program binarny a.out
                                dołączona biblioteka matematyczna
cc -xc99 -Xc -o prog prog.c  ⟸  wynik: program binarny prog
```

Kompilacja programów – wywołania

- kompilator Sun C (komputer diablo)

`cc -xc99 -Xc prog.c`	$\Longleftarrow$	wynik: program binarny `a.out`
`cc -xc99 -Xc prog.c -lm`	$\Longleftarrow$	wynik: program binarny `a.out` dołączona biblioteka matematyczna
`cc -xc99 -Xc -o prog prog.c`	$\Longleftarrow$	wynik: program binarny `prog`
`cc -xc99 -Xc -c prog.c`	$\Longleftarrow$	wynik: program wynikowy `prog.o`

Kompilacja programów – wywołania

- kompilator Sun C (komputer diablo)

`cc -xc99 -Xc prog.c`	$\Longleftarrow$	wynik: program binarny `a.out`
`cc -xc99 -Xc prog.c -lm`	$\Longleftarrow$	wynik: program binarny `a.out` dołączona biblioteka matematyczna
`cc -xc99 -Xc -o prog prog.c`	$\Longleftarrow$	wynik: program binarny `prog`
`cc -xc99 -Xc -c prog.c`	$\Longleftarrow$	wynik: program wynikowy `prog.o`
`cc prog.o`	$\Longleftarrow$	wynik: program binarny `a.out`

Kompilacja programów – wywołania

- kompilator Sun C (komputer diablo)

```
cc -xc99 -Xc prog.c          ⟸  wynik: program binarny a.out
cc -xc99 -Xc prog.c -lm      ⟸  wynik: program binarny a.out
                                dołączona biblioteka matematyczna
cc -xc99 -Xc -o prog prog.c  ⟸  wynik: program binarny prog
cc -xc99 -Xc -c prog.c       ⟸  wynik: program wynikowy prog.o
cc prog.o                    ⟸  wynik: program binarny a.out
```

- kompilator GNU C (komputer diablo, panamint)

```
gcc -std=c99 -pedantic -Wall prog.c          ⟸ program binarny a.out
gcc -std=c99 -pedantic -Wall prog.c -lm      ⟸ program binarny a.out
                                               dołączona biblioteka matematyczna
gcc -std=c99 -pedantic -Wall -o prog prog.c  ⟸ program binarny prog
gcc -std=c99 -pedantic -Wall -c prog.c       ⟸ program wynikowy prog.o
gcc prog.o                                   ⟸ wynik: program binarny a.out
```

Kompilacja programów – przykłady

- kompilator Sun C (komputer diablo)

```
diablo 24: cc -xc99 -Xc -o dwumian dwumian.c
"dwumian.c", line 4: warning: function must be of type int: main()
"dwumian.c", line 31: syntax error before or at: }
"dwumian.c", line 48: syntax error before or at: else
cc: acomp failed for dwumian.c
```

Kompilacja programów – przykłady

- kompilator Sun C (komputer diablo)

```
diablo 24: cc -xc99 -Xc -o dwumian dwumian.c
"dwumian.c", line 4: warning: function must be of type int: main()
"dwumian.c", line 31: syntax error before or at: }
"dwumian.c", line 48: syntax error before or at: else
cc: acomp failed for dwumian.c
```

- kompilator GNU C (komputer diablo, panamint)

```
[mucha@panamint:~]:gcc -std=c99 -Wall -pedantic -o dwumian dwumian.c
dwumian.c:4: warning: return type of 'main' is not 'int'
dwumian.c: In function 'main':
dwumian.c:21: warning: suggest parentheses around assignment used as truth value
dwumian.c:31: error: expected ';' before '}' token
dwumian.c:45: warning: suggest parentheses around assignment used as truth value
dwumian.c:48: error: expected ';' before 'else'
"dwumian.c", line 4: warning: function must be of type int: main()
"dwumian.c", line 31: syntax error before or at: }
"dwumian.c", line 48: syntax error before or at: else
cc: acomp failed for dwumian.c
```

Uruchamianie programów

```
diablo 25:
```

Uruchamianie programów

```
diablo 25: ./dwumian                    (* alternatywnie dwumian *)
```

Uruchamianie programów

```
diablo 25: ./dwumian                    (* alternatywnie dwumian *)
Program rozwiazuje rownanie kwadratowe.
Podaj wspolczynnik a, b, c:
```

Uruchamianie programów

```
diablo 25: ./dwumian                    (* alternatywnie dwumian *)
Program rozwiazuje rownanie kwadratowe.
Podaj wspolczynnik a, b, c:
3 4 -5
```

Uruchamianie programów

```
diablo 25: ./dwumian                (* alternatywnie dwumian *)
Program rozwiazuje rownanie kwadratowe.
Podaj wspolczynnik a, b, c:
3 4 -5
Rozwiazanie rownania sa pierwiastki:
  x1 = -2.119633
  x2 = 0.786300
diablo 26:
```

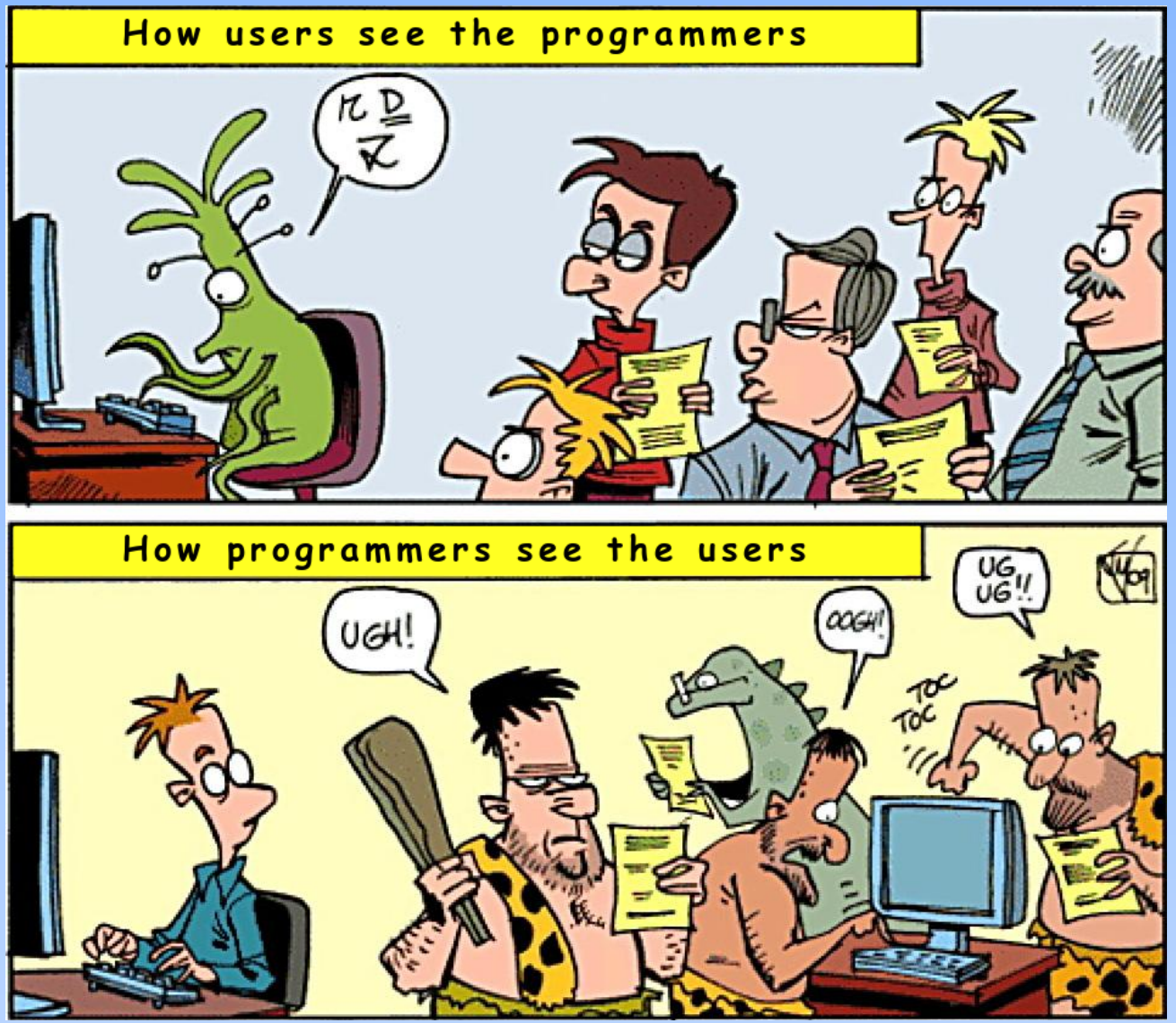

Uruchamianie programów

```
diablo 25: ./dwumian                    (* alternatywnie dwumian *)
Program rozwiazuje rownanie kwadratowe.
Podaj wspolczynnik a, b, c:
3 4 -5
Rozwiazanie rownania sa pierwiastki:
  x1 = -2.119633
  x2 = 0.786300
diablo 26: ./dwumian
Program rozwiazuje rownanie kwadratowe.
Podaj wspolczynnik a, b, c:
1
2
3
Brak rozwiazan rzeczywistych.
```

Uruchamianie programów cd.

```
diablo 25: ./dwumian
Program rozwiazuje rownanie kwadratowe.
Podaj wspolczynnik a, b, c:
-1 0 10000000000000000000
Rozwiazanie rownania sa pierwiastki:
  x1 = 99999997952.000000
  x2 = -99999997952.000000
```

Ku pokrzepieniu



Podsumowanie

- ## Zagadnienia podstawowe

1. Jaka jest relacja między specyfikacją zadania a algorytmem?
2. Jaka jest różnica między algorytmem a programem?
3. Czym się różni kompilacja od konsolidacji programu?
4. Czym są opcje kompilacji programu i jak ich użyć?
5. W których miejscach w kodzie programu należy zamieszczać komentarze?
6. Czy bezbłędne skompilowanie programu gwarantuje jego poprawne działanie?
7. Czy kompilacja programu na komputerze klasy PC (domowym) i stacji roboczej (np. Sun jak diablo) daje zawsze takie same rezultaty?
8. Co przedstawia diagram algorytmu?
9. Czy za pomocą diagramu algorytmu można przedstawić projekt każdego programu?

- ## Zagadnienia rozszerzające

1. Jakie elementy powinna zawierać dokumentacja programu?
2. Jakie inne poza podanymi na wykładzie opcje wywołania kompilatora są dostępne w kompilatorze `cc` (na diablo), a jakie w `gcc` (panamint, komputer domowy)?

3. Jakie informacje umieszcza w programie kompilator przy włączonej opcji `-g`? W jaki sposób ułatwia to debugowanie programu?
4. Wskaż elementarne metody ułatwiające proces odpluskwiania programu?
5. Linker może dołączać biblioteki do programu w sposób statyczny i dynamiczny. Czym różnią się te dwa sposoby? Jak stwierdzić, jakie biblioteki dynamiczne zostały dołączone do programu?

• Zadania

1. Zapisz za pomocą diagramu algorytm pozwalający na posortowanie trzech liczb.
2. Zapisz za pomocą diagramu algorytm kodujący tekst za pomocą szyfru Cezara.
3. Napisz najkrótszy z możliwych program w języku C.
4. Napisz program wyliczający liczby Fibonacciego.
5. Napisz program wypisujący przedziały, w których zadana funkcja kwadratowa przyjmuje wartości poniżej zera.

Indeks

- Literatura
- Materiały dostępne w sieci www
- Zawartość tematyczna wykładu
- Środowisko pracy
- Etapy pracy nad programem
- Edytor GNU Emacs
- Przykład programu w C
- Diagram programu
- Kompilacja programów
  - wywołania
  - przykłady
- Uruchamianie programów