Tablica elementów, w której wyszukujemy połówkowo. Podzielono ją między dwa procesory pracujące z jednakową mocą obliczeniową. Określić przyśpieszenie przy wyszukiwaniu po podziale.
Jakie działania na liczbach zmiennoprzecinkowych mogą spowodować przepełnienie?
Masz 21 bitów na cechę i 30 na mantysę; napisać: liczbę największą, najmniejszą, najmniejszą dodatnia (nie-zero).
Podać różnice pomiędzy asemblerem, kodem maszynowym i językiem asemblera
Na przykładzie konstrukcji if w Pascalu podać co to jest: syntaktyka,semantyka
Sposoby alokowania danych w Pascalu.
Poblemy nierozwiązywalne i wysoce nierozwiązywalne.
Policzyć ilość informacji w 100-elementowym ciągu liczb z przedziału [1,255], z których każda jest równie prawdopodobna.
Podać sposób kodowania liczb z zakresu -10 do 1000; ile minimum bitów potrzeba, aby zapisać wszystkie te liczby?
Cechy quicksorta
Języki imperatywne, aplikatywne i deklaratywne
Podać definicję interpretacji
Instrukcja repeat: podać składnię i semantykę, narysować schemat
Przekazywanie przez zmienną i przez wartość: zaznaczyć poprawne wywołania
Podać rozmiary zadeklarowanych zmiennych (są podane rozmiary poszczególnych typów)
Ile bitów potrzeba do zapisania liczby zmiennoprzecinkowej z przedziału od 10-6 do 106 z dokładnością do trzech cyfr dziesiętnych?
Jak można zwiększyć szybkość dostępu do pamięci/CPU/urządzeń IO?
Czym się różni plik sekwencyjny od pliku o swobodnym dostępie?
Młody programista próbuje napisać rozwiązanie problemu nierozstrzygalnego, jakim działaniem zakończy się działanie jego programu?
Posortować swoje nazwisko metodą sortowania przez proste wstawianie, rozpisać krok po kroku
Napisać składnię ifa w notacji EBNF
Czym się różni CPU (procesor) od ALU (jednostki arytmetyczno-logicznej)?
Syntaktyka, semantyka i schemat blokowy pętli for :
Co to jest i do czego służy cache?
Robaczki!Zadeklaruj odpowiednie typy i zmienne by wyrażenie było poprawne:
Napisz fragment programu bez użycia instrukcji while
Za pomoca instrukcji strukturalnych:
Napisz co wypisze program:(komentarze dodane przez autorów opracowania)
Kilka EBNFów:
Algorytmy sortowania
Liczby zmiennoprzecinkowe w formacie iCon:
Pojęcia: translator, kompilator, debuger, linker, profiler (wady zalety)

Fazy kompilacji
Architektury
Podejścia do programowania top-down, bottom-up
System operacyjny
Maszyna wielopoziomowa

Opracowała grupa VI, Informatyka rok 1, 2010/11
Przy pomocy materiałów dostępnych na forum oraz EgzamWiki.

Tablica elementów, w której wyszukujemy połówkowo. Podzielono ją między dwa procesory pracujące z jednakową mocą obliczeniową. Określić przyśpieszenie przy wyszukiwaniu po podziale.

Brak przyspieszenia. Jak nie wiesz czemu, to nie wiem, czy jesteś na dobrym kierunku. / Lord iCon
(dlatego bo $\log_2(N/2)=\log_2(N)-1$)

Jakie działania na liczbach zmiennoprzecinkowych mogą spowodować przepełnienie?

- dodawanie, mnożenie, dzielenie, odejmowanie, jeśli ich wynik nie może być zapisany przy użyciu dostępnej liczby bitów.
(Jak przekraczamy zakres w dół to jest niedomiar, jak w górę to nadmiar).

Masz 21 bitów na cechę i 30 na mantysę; napisać: liczbę największą, najmniejszą, najmniejszą dodatnia (nie-zero).

Jeśli mamy 21 bitów na cechę i zapisujemy ją w U2 to cecha jest w przedzialiku $[-2^{20} \dots 2^{20}-1]$.
Zakładam, że w mantysie przeznaczamy jeden bit na znak liczby a 29 bitów na wartość mantysy. Wtedy minimalna znormalizowana (w notacji Icona) mantysa wynosi ½ . Maksymalna natomiast to suma szeregu ½ + ¼ + … + 1 / 2^n w tym wypadku aż do 1/ 2^29 = (1/2) * (1- (1/ 2)^29)/(1 - 1/2) = 1 - ½^29co można ale nie trzeba zaokrąglić do 1.
W takim razie maksymalna liczba dodatnia, którą możemy zapisać przy znormalizowanej mantysie to:
*maksymalna mantysa znormalizowana * 2^(maksymalna cecha)*
$(1 - 0.5^{29}) * 2^{(2^{20} -1)}$
Minimalna liczba dodatnia, którą możemy zapisać to:
*minimalna mantysa znormalizowana * 2^(minimalna cecha)*
½ * 2^(-2^20) = 2^(-1) * 2^(-2^20) = $2^{(- 2^{20} - 1)}$
Najmniejsza możliwa do zapisania liczba będzie taka sama, jak największa, tylko na minusie, gdyż nie uwzględniłem w mantysie kodu u2, bo chyba nikt nie ma pojęcia jak działa u2 dla

ułamków ;-)

Podać różnice pomiędzy asemblerem, kodem maszynowym i językiem asemblera

Kod maszynowy – rozkazy numeryczne wykonywane przez procesor
Asembler – program tłumaczący język asemblera do kodu maszynowego
Język asemblera – język programowania niskiego poziomu, kody operacji z kodu maszynowego zostały w nim zastąpione mnemonikami

Na przykładzie konstrukcji if w Pascalu podać co to jest: syntaktyka,semantyka

Syntaktyka (składnia) – opisuje rodzaje dostępnych symboli i zasady na podstawie których mogą być one łączone w większe struktury (notacja BNF, EBNF) [co potrafi zrozumieć kompilator]
Semantyka – definiuje precyzyjnie znaczenie symboli oraz ich funkcje w programie (najczęściej słownie) [do czego służy dana konstrukcja]
(Pragmatyka - opisuje, w jaki sposób należy używać tego języka, w jakim celu i przy użyciu jakich reguł praktycznych) [w jakiej sytuacji i w jaki sposób użyć danej konstrukcji]

instrukcja warunkowa ::= if <warunek> then <instrukcja> [else <instrukcja>]
Jeśli warunek jest spełniony to wykonaj instrukcję. W przeciwnym wypadku wykonaj instrukcję po słowie else, jeśli ono występuje.

Sposoby alokowania danych w Pascalu.

1. statyczne – zmienne globalne

2. dynamiczne-stos – (automatyczne)zmienne lokalne procedur i funkcji, zmienne przekazane przez wartość

3. dynamiczne-sterta –(kontrolowane) zmienne tworzone przez new

Poblemy nierozwiązywalne i wysoce nierozwiązywalne.

Łatwo rozwiązywalne: dobra złożoność (wielomianowa)
- Istnieją skuteczne algorytmy
Teoria OK, Praktyka OK

Trudno rozwiazywalne: złożoność wykładnicza lub gorzej
- Nie istnieją rozsądne algorytmy
Teoria OK, Praktyka :(

Problem nierozwiązywalny (nierozstrzygalny): nie istnieje dla niego algorytm który po skończonej liczbie kroków poda jednoznaczna odpowiedz dla dowolnych danych (albo poda błędną odpowiedź albo się zapętli). Przykład: problem stopu, problem domina(dla półpłaszczyzny)
- Nie istnieją dla nich algorytmy
Teoria :(, Praktyka :(

Problem wysoce nierozwiązywalny (nierozstrzygalny): zadanie sprowadzenia go do problemu nierozstrzygalnego jest samo w sobie nierozstrzygalne
(przyklad - okresowy problem domina)
- Nie można sprowadzić do tych, dla których nie istnieją algorytmy.
Teoria :/, Praktyka :/

Policzyć ilość informacji w 100-elementowym ciągu liczb z przedziału [1,255], z których każda jest równie prawdopodobna.
100* -(suma od 1 do 255) $1/255*\log_2(1/255) \approx 800$
Wzór Shannona na entropie jednego elementu to zbioru to ($-\Sigma^{n}_{1} p_i * log_2 p_i$) Czyli suma iloczynu prawdopodobieństwa danego elementu i logarytmu binarnego z tego prawdopodobieństwa dla wszystkich elementów zbioru po kolei, pomnożona przez (-1).

Podać sposób kodowania liczb z zakresu -10 do 1000; ile minimum bitów potrzeba, aby zapisać wszystkie te liczby?
W danym zakresie mamy 1011 liczb (10 ujemnych, 1 zero, 1000 dodatnich)
Przesuwamy zakres o –495. Otrzymamy wtedy zakres symetryczny:
-505..505
8=log[2]256<log[2]505<log[2]512=9
Do zapisania takiego zakresu potrzebujemy:
1 bit na zapamiętanie znaku
9 bitów na zapamiętanie wartości
Minimalna liczba bitów niezbędna do zapisania liczb z tego zakresu: 10 bitów
Potem przy odczytywaniu wartości stosujemy translacje o +495.
(a Pan Paweł sądzi, że równie dobrze można zapisywać cyfrę w zwykłym kodzie binarnym i wtedy też mamy log2[1011] zaokrąglony do góry, czyli 10 bitów. Oczywiście translacja o 10 :) // no i Pan Paweł dobrze myśli :P]

Cechy quicksorta

-niestabilny(nie zachowuje względnej kolejności)
-złożoność nlogn, pesymistyczna n^2
-wersje iteracyjna i rekurencyjna, wiele wariantów(horowitz, losowy, mediana)

Języki imperatywne, aplikatywne i deklaratywne

Imperatywne – pascal,c: program to ciąg wykonywanych po sobie instrukcji
Aplikatywne (funkcyjne) – lisp, (haskell, erlang): funkcja jako wartość podstawowa, definiowanie funkcji (często rekurencyjnych), często wzajemnie się wywołują
Deklaratywne (logiczne) – prolog, sql: opisujemy warunki jakie musi spełniać rozwiązanie, a nie konkretny algorytm jego znalezienia
Aplikatywne i deklaratywne są wyższego poziomu i mają słabszą efektywność

Podać definicję interpretacji

interpretacja – zamiana kodu zrodlowego na kod wykonywalny (linijka po linijce), dokonywana przez interpreter, przy kazdym uruchomieniu programu - > wykonywanie kodu programu podczas jego ‘czytania’.

Instrukcja repeat: podać składnię i semantykę, narysować schemat

<repeat> ::= repeat<ciąg instrukcji> until <warunek>
Instrukcja repeat jest pętlą, która wykonuje zawarty w jej wnętrzu (między słowami kluczowymi repeat, until) ciąg instrukcji co najmniej raz. Po spełnieniu warunku wyjścia pętla zostaje przerwana.

Przekazywanie przez zmienną i przez wartość: zaznaczyć poprawne wywołania

```
Procedure Dwa(var a : integer);
Procedure Jeden(a : integer);
```

a) Jeden(12); TAK
b) Jeden(k); TAK
c) Jeden(k+12); TAK

d) Dwa(12); NIE (var)
e) Dwa(k); TAK
f) Dwa(k+12); NIE (var)

Co to znaczy, ze algorytm sortowania tablicy ma złożoność O(n^2)?
Ilość operacji (czasu w wypadku złożoności czasowej) potrzebnych do posortowania tablicy przy wzroście jej rozmiaru rośnie tak, jak kwadrat ilości elementów tablicy.
(Istnieje takie c, że liczba operacji elementarnych potrzebnych do posortowania tablicy w zależności od ilości elementów (n) jest mniejsza od c*n^2 dla prawie wszystkich n.)

Podać rozmiary zadeklarowanych zmiennych (są podane rozmiary poszczególnych typów)

Przykład: Zmienna typu integer zajmuje 2 bajty, wskaźnik zajmuje 4 bajty. Ile pamięci zajmują następujące zmienne:

```
a : array[1..100] of ^integer;  - 100 * 4 B
b : ^array[1..100] of integer; -  1*4 B
c : ^array[1..100] of ^integer; - 1*4 B
```

Ile bitów potrzeba do zapisania liczby zmiennoprzecinkowej z przedziału od 10^{-6} do 10^6 z dokładnością do trzech cyfr dziesiętnych?

Do zapisu w systemie dziesiątkowym mantysy potrzebowalibyśmy 3 cyfr. Mamy więc zależność:
$2^{-x} = 10^{-10}$
2^x = 10^3
log2(2^x) = log2(10^3)
x = log2(10^3)
x = 3 log2(10)
x = 3/log10(2) ~ 3/0.301
$x \sim 10$
Mając mantysę Iconową dodajemy 1 bit na znak liczby, więc potrzeba 11 bitów.
Maksymalna mantysa wynosi (1 - ½ ^10) co można przybliżyć jako 1.
Teraz liczymy cechę, cecha to x, więc
$1 * 2^x = 10^6$
$x \sim 20$
W takim razie musimy przechować liczby [-20, 20] jako cechę (zakładamy, że cecha jest w U2 co wymaga symetrii zakresu).
Biorąc system U2 potrzeba nam 6 bitów na cechę, (to daje zakres [-32 .. 31])
Razem mamy 6+11 = 17 bitów.

(uwaga, znana także wersja zadania z dokładnością do 6 cyfr, wtedy wychodzi ~27 bitów)

Jak można zwiększyć szybkość dostępu do pamięci/CPU/urządzeń IO?

Pamięć: podzielić pamięć na moduły i zastosować przeplot. Zastosowanie pamięci Cache.
CPU: pipeline - przetwarzanie potokowe, superscalar
Pipelining jest to technika budowy procesorów polegająca na podziale logiki procesora odpowiedzialnej za proces wykonywania programu (instrukcji procesora) na specjalizowane grupy w taki sposób aby każda z grup wykonywała część pracy związanej z wykonaniem rozkazu. - gdy jedna część odbiera rozkaz, druga przetwarza poprzedni
Superskalarność (ang. Superscalar) – jest to cecha mikroprocesorów oznaczająca możliwość jednoczesnego ukończenia kilku instrukcji w pojedynczym cyklu zegara. Jest to możliwe dzięki zwielokrotnieniu jednostek wykonawczych, co umożliwia obliczenia równoległe.
IO: Zastosowanie DMA (kanał I/O), przerwania(?)
DMA (Direct Memory Access) - kopiowanie danych bez użycia CPU, wykorzystuje się praktycznie tylko w HDD i kartach dźwiękowych

Czym się różni plik sekwencyjny od pliku o swobodnym dostępie?

Plik sekwencyjny można czytać tylko od początku do końca (np taśma), dane można dopisywać tylko na koniec takiego pliku, natomiast plik o dostępie swobodnym można czytać i zapisywać w dowolnym miejscu (np. dysk twardy).

Młody programista próbuje napisać rozwiązanie problemu nierozstrzygalnego, jakim działaniem zakończy się działanie jego programu?

Program nie zakończy się, zakończy się błędem wykonywania lub będzie dawał błędne wyniki dla niektórych danych.

Posortować swoje nazwisko metodą sortowania przez proste wstawianie, rozpisać krok po kroku

WSTAWIANIE (wstawiamy w odpowiednie miejsce): - algorytm stabilny
C|ZAPLIŃSKI
C**Z**|APLIŃSKI
ACZ|PLIŃSKI
AC**P**Z|LIŃSKI
AC**L**PZ|IŃSKI
AC**I**LPZ|ŃSKI
ACIL**Ń**PZ|SKI
ACILŃP**S**Z|KI
ACIL**K**ŃPSZ|I

ACI**I**LKŃPSZ
ACIILKŃPSZ

WYBIERANIE (wybieramy najmniejszy) - algorytm niestabilny
CZAPLIŃSKI | A->1
AZCPLIŃSKI | C->2
ACZPLIŃSKI | I -> 3
ACIPLZŃSKI | I -> 4
ACIILZŃSKP | L->5
ACIILZŃSKP | K->6
ACIILKŃSZP | Ń->7
ACIILKŃSZP | P->8
ACIILKŃPZS | S->9
ACIILKŃPSZ

Napisać składnię ifa w notacji EBNF
if::= if <warunek> then <instrukcja> [else <instrukcja>]

Czym się różni CPU (procesor) od ALU (jednostki arytmetyczno-logicznej)?
CPU (procesor) - cyfrowe urządzenie sekwencyjne potrafiące pobierać dane z pamięci, interpretować je i wykonywać jako rozkazy

ALU (jednostka arytmetyczno-logiczna) - jedna z głownych części procesora prowadząca proste operacje na liczbach całkowitych

Syntaktyka, semantyka i schemat blokowy pętli for :

Syntaktyka EBNF:
FOR <zmienna> := <wyrazenie1> (TO | DOWNTO) <wyrazenie2> do <instrukcja>

Semantyka:
Działanie pętli for:
1.przypisz zmiennej ‘zmienna’ wyrażenie1
2.wykonaj instrukcję ‘instrukcja’
3. w przypadku słowa kluczowego TO zwiększ wartość zmiennej, w przypadku DOWNTO zmniejsz

4. Jeśli wartość zmiennej jest większa (w przypadku słowa TO)/mniejsza (w przypadku DOWNTO) niż wyrażenie2, to zakończ pętlę, w przeciwnym wypadku wróć do punktu2.

Co to jest i do czego służy cache?

Pamięć podręczna przyspiesza dostęp do relatywnie wolnej pamięci RAM. Charakteryzuje się bardzo krótkim czasem dostępu. Jest używana do przechowywania danych, które będą w niedługim czasie przetwarzane.
Cache to pamięć statyczna (więcej tranzystorów do 1 bitu informacji = droższa w wykonaniu)
Dla ścisłości DRAM - dynamiczna (1 tranzystor = 1 bit)

Robaczki!
Zadeklaruj odpowiednie typy i zmienne by wyrażenie było poprawne:

a^.x[3]^[7].b

```
type
ostatni = record
b:integer;
end;
przedostatni = array[1..10] of ostatni;
wskaznik2 = ^przedostatni;
pierwszy = record
x: array[1..10] of wskaznik2;    end;
wskaznik1 = ^pierwszy;
var
a:wskaznik1;
begin
new(a);
new(a^.x[3]);
a^.x[3]^[7].b:=51;
writeln(a^.x[3]^[7].b);
readln();
end.
```

a[3][4].b

```
type cos = record
b:integer;
end;
a:array [1..max,1..max] of cos;
```

a.b[6]

```
type cos = record
b:array [1..max] of integer;
end;
```

a^b.c
(Nierozwiązywalny?)

c(b.e)

```
type cos = record
e:integer;
end;
b:cos;
function c(a:integer):integer;
```

a[b[3]]

```
var
b:array[1..10] of integer;
a:array[1..10]of integer;
```

b.c(4) (raczej nie będzie)

```
type
f=procedure (i:integer);
cos = record
c:f;
end;
procedure wypisz(i:integer);
begin
writeln(i);
end;
var
b:cos;
begin;
b.c:=@wypisz;
b.c(4);
end.
```

a[5]^[6]

```
type tab = array[1..20] of integer;
ptab =^tab;
var a:array[1..10] of ptab;
```

a^^[‘b’]^^[c-d]

```
type tab = array[0..10] of integer;
p1 = ^tab;
p2 = ^p1;
type chartab = array['a'..'z']of p2;
p3 = ^chartab;
p4 = ^p3;
var
c,d:integer;
a:p4;
begin;
new(a);
new(a^);
new(a^^['b']);
new(a^^['b']^);
c:=10;
d:=5;
end.
```

a := b[‘w’,s[‘x’,’y’]]-ord[‘z’]

```
var
b:array['a'..'z',1..1000] of integer;
s:array['a'..'z','a'..'z'] of integer;
ord:array['a'..'z'] of integer;
a:integer;
begin
s['x','y'] := 1;
b['w'][1] := 150;
ord['z'] := 149;
a := b['w',s['x','y']] - ord['z'];
writeln(a);
readln();
end.
```

alternatywa:
```
var
s:array['x'..'x','y'..'y'] of char;
b:array['w'..'w','w'..'w'] of integer;
ord:array['z'..'z'] of integer;
a:integer;
begin
s['x','y']:='w';
b['w',s['x','y']]:=1;
ord['z']:=10;
a := b['w',s['x','y']]-ord['z'];
writeln(a);
end.
```

a^^[b(c^)^]

```
type tab = array[1..20] of integer;
ptab = ^tab;
ppointer = ^ptab;

var c:^integer;
a:ppointer;

function b(wartosc:integer):pint;
```

a[b=1]^^^.x

```
type r = record
   x:integer;
   end;
type e = ^r;
type w = ^e;
type q = ^w;
type tab = array[false..true] of q;

var
b:integer;
a:tab;
begin
b:=2;
new(a[false]);
```

```
new(a[false]^);
new(a[false]^^);
a[b=1]^^^.x:=99;
writeln(a[b=1]^^^.x);
end.
```

d^[a.x.x.x=x.x.x.a]^

```
type int_node = ^integer; {wskaźnik, bylejaki}
type tab = array[false..true] of int_node; {tablica wskaźników indeksowana boolami}
type tab_node = ^tab; {wskaźnik do tej tablicy}
{x.x.x.a}
type lev_2 = record
a:integer;
end;
type lev_1 = record
x:lev_2;
end;
type lev_0 = record
x:lev_1;
end;
{a.x.x.x}
type lev_2b = record
x:integer;
end;
type lev_1b = record
x:lev_2b;
end;
type lev_0b = record
x:lev_1b;
end;
var x:lev_0;
var a:lev_0b;
var d:tab_node;
begin
a.x.x.x:=10;
x.x.x.a:=11;
new(d);
new(d^[true]);
new(d^[false]);
```

```
d^[true]^:=1;
d^[false]^:=0;
writeln(d^[a.x.x.x=x.x.x.a]^);
end.
```

II rozwiązanie: **d^[a.x.x.x=x.x.x.a]^**

```
type
t1=array[boolean] of ^integer;
t2=^t1;
var
d:t2;
a:record
  x:record
   x:record
    x:integer;
   end;
  end;
end;
x:record
  x:record
   x:record
    a:integer;
   end;
  end;
end;
```

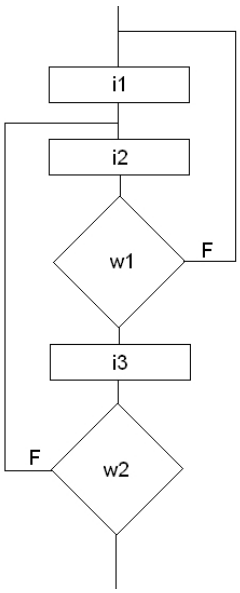
Napisz fragment programu bez użycia instrukcji while

będący odpowiednikiem poniższego programu:

```
read(i);
j := 0;
while i<100 do
begin
        i := i+1.0;
        j := j+2*i;
end;
writeln(i,j);
```

```
read(i);
j := 0;
if i<100 then
        repeat
                i := i+1.0;
                j := j+2*i;
        until i>=100;
writeln(i, j);
```

Za pomoca instrukcji strukturalnych:




```
i1;
repeat
   i2;
   while (not w1) do begin
         i1;
         i2;
    end
i3;
until w2;
```

Napisz co wypisze program:
(komentarze dodane przez autorów opracowania)

```
var a,b,c:integer;
```

```
function ola(a:integer; var b:integer):integer; {tak naprawdę b, c}
begin
  c:=b+a; {to jest c:=c+b czyli c:= 3+2 :=5}
  ola:=c-b; {to c-c=0, ola zwraca 0}
end;
procedure ala(var b:integer; c:integer);  {a, 0} {b - wskazuje na a}
begin
  a:=b+c; {a:=a+0}
  b:=6+a; {a:=a+6}
end;
begin
  a:=1;
  b:=2;
  c:=3;
  ala(a,ola(b,c));
  writeln('a= ',a,', b= ',b,', c= ',c);

end.
```

a=7,b=2,c=5

Kilka EBNFów:

[ ] - 0 lub 1 raz
{ } - 0 lub więcej razy
| - alternatywa
if <wyrażenie> **then** <instrukcja> [**else** <instrukcja>]
case <wyrażenie> **of** <case-item> { “;” <case-item> } [“;”] **end**
<case-item> ::= <stała> { “;” <stała>} “:” <instrukcja>
while <wyrażenie> **do** <instrukcja>
repeat <instrukcja> { “;” <instrukcja> } **until** <wyrażenie>
for <identyfikator_zmiennej> “:=” <wyrażenie> (**to** | **downto**) <wyrażenie> **do** <instrukcja>
więcej: http://www.lrz.de/~bernhard/Pascal-EBNF.html

Algorytmy sortowania

Stabilne: bąbelkowe O(n^2), wstawianie O(n^2), scalanie O(nlog n), zliczanie, kubełkowe, pozycyjne, biblioteczne
Niestabilne: wybieranie O(n^2), shella, grzebieniowe, quicksort i jego mody, heapsort O(n log n),

Algorytm	Stabilność	Czas średni	Uwagi
bąbelkowe	tak	O(n^2)	
wstawianie	tak	O(n^2)	optymistycznie n
scalanie	tak	O(n log n)	O(n) dodatkowej pamięci
zliczanie	tak	O(n + zakres)	Zakres elementów ogran
wybieranie	nie	O(n^2)	
shella	nie	O(n^1,15)	wstawianie lvl up
combsort	nie	prawd. O(n log n)	bublesort lvl up
quicksort	nie	O(n log n)	pesymistycznie O(n^2)
heapsort	nie	O(n log n)	brak dodatkowej pamięci

Liczby zmiennoprzecinkowe w formacie iCon:

- c - liczba bitów na cechę
- m - liczba bitów na mantysę
- [znak liczby][m-mantysa][c-cecha(ze znakiem]
- maksymalna cecha: $2^{(c-1)}-1$
- minimalna cecha: $-2^{(c-1)}$
- maksymalna mantysa: $1 - 2^{(-m)}$
- minimalna mantysa (znormalizowana): ½
- minimalna dodatnia: $(1/2)* 2^{(-2^{(c-1)})}$ (minimalna mantysa$^{\text{minimalna cecha}}$)
- maksymalna: $(1-2^{(-m)}) * 2^{((2^{(c-1)})-1)}$ (maksymalna mantysa$^{\text{maksymalna cecha}}$)

Pojęcia: translator, kompilator, debuger, linker, profiler (wady zalety)

Translator to specjalny program komputerowy (lub urządzenie), dokonujący tłumaczenia (translacji) programu napisanego w języku programowania, z postaci źródłowej do postaci wynikowej, zrozumiałej dla maszyny. Czasami zamiast określenia kod wynikowy używa się równoważnego kod obiektowy.
Translatory dzieli się na dwie grupy: kompilatory tłumaczące programy zapisane w językach wysokiego poziomu oraz asemblery tłumaczące programy zapisane w językach symbolicznych.

Uwaga: czasem nazywa się tak program tłumaczący instrukcje w jednym języku programowania na inny język programowania - tak było chyba mówione na wykładzie, tak jest w angielskiej wikipedii.

Kompilator (ang. compiler) to program służący do automatycznego tłumaczenia kodu napisanego w jednym języku (języku źródłowym) na równoważny kod w innym języku (języku wynikowym). Proces ten nazywany jest kompilacją. W informatyce pojęciem kompilatora określa się najczęściej program do tłumaczenia kodu źródłowego w języku programowania na język maszynowy. Niektóre z nich tłumaczą najpierw do języka asemblera, a ten na język maszynowy jest tłumaczony przez asembler.

Debug tool, Debugger – program komputerowy służący do dynamicznej analizy innych programów, w celu odnalezienia i identyfikacji zawartych w nich błędów.Podstawowym zadaniem debuggera jest sprawowanie kontroli nad wykonaniem kodu, co umożliwia zlokalizowanie instrukcji odpowiedzialnych za wadliwe działanie programu. Współczesne debuggery pozwalają na efektywne śledzenie wartości poszczególnych zmiennych, wykonywanie instrukcji krok po kroku czy wstrzymywanie działania programu w określonych miejscach.

Konsolidator (ang. **linker)** lub program konsolidujący to jeden z programów składowych kompilatora. Konsolidator w trakcie procesu konsolidacji łączy zadane pliki obiektowe i biblioteki statyczne tworząc w ten sposób plik wykonywalny.

Interpreter – program komputerowy, który analizuje kod źródłowy programu, a przeanalizowane fragmenty wykonuje.

Cross-kompilator (ang. Cross Compiler) jest to kompilator zdolny do generowania kodu wykonywalnego dla innej platformy (procesora) niż ta, na której wykonuje się cross-kompilator

Profiler - w inżynierii oprogramowania, program profilujący lub po prostu profilowanie, to forma dynamicznej analizy programu (przeciwność statycznej analizy kodu). Jest to badanie zachowania programu używając informacji zdobytych podczas jego wykonywania. Zwykle przeprowadza się je by dowiedzieć się, które części programu zoptymalizować, by zwiększyć jego ogólną prędkość lub zmniejszyć wymagania pamięci.

Fazy kompilacji

• preprocessing (preprocesor)
Preprocesor – program interpretujący, którego zadaniem jest przetworzenie tekstu wejściowego w sposób określony za pomocą poleceń preprocesora przez programistę na tekst wyjściowy. Dopiero tak przetworzony tekst poddawany jest analizie składniowej i kompilacji. Wynikiem działania preprocesora jest więc tekst wyjściowy po przetworzeniu podlegający następnie kompilacji.

• analiza leksykalna (skaner) - wyodrębnia podstawowe jednostki (liczby, słowa)
Lekser (ang. lexer lub scanner), nazywany też analizatorem leksykalnym, to program komputerowy który dokonuje analizy leksykalnej danych wejściowych, zwykle jako pierwsza część jakiegoś większego procesu, np. kompilacji.
• analiza składniowa (parser) - bada zgodność z gramatyką języka
Parser (inaczej analizator składniowy) w informatyce program dokonujący analizy danych wejściowych w celu określenia ich gramatycznej struktury w związku z formalną gramatyką. Nazwa analizator składniowy podkreśla analogię zastosowania programu do analizy stosowanej w gramatyce i lingwistyce. Dzięki temu procesowi komputery są w stanie przetworzyć czytelny dla człowieka tekst w strukturę danych przydatną do dalszej obróbki.
Najczęściej stosowane są własne standardy notacji oparte o BNF.
Istnieją generatory parserów
• optymalizacja niezależna od architektury
• optymalizacja zależna od architektury
• dołączanie bibliotek (linker)
• generacja kodu

Architektury

Architektura von Neumanna - rodzaj architektury komputera, przedstawionej po raz pierwszy w 1945 roku przez Johna von Neumanna.

Polega na ścisłym podziale komputera na trzy podstawowe części:
* procesor (w ramach którego wydzielona bywa część sterująca oraz część arytmetyczno-logiczna)
* pamięć komputera (zawierająca dane i sam program)
* urządzenia wejścia/wyjścia

Architektura wieloprocesorowa: magistrala do której przypięte są procesory z własną pamięcią
Architektura wielomaszynowa: sieć lokalna
System z wspólną pamięcią: jedna duża RAM do której przypięte są procesory.
Typowe rodzaje architektury - klasyfikacja Flynna:

- **SISD** (**S**ingle **I**nstruction, **S**ingle **D**ata)
- **SIMD** (**S**ingle **I**nstruction, **M**ultiple **D**ata)
- **MISD** (**M**ultiple **I**nstruction, **S**ingle **D**ata)
- **MIMD** (**M**ultiple **I**nstruction, **M**ultiple **D**ata)

Podejścia do programowania top-down, bottom-up

Top-down - analityczne, piszemy program używając elementów, które ‘napiszemy później’.
Bottom-up - syntetyczne, piszemy małe kawałki kodu, pomocnicze procedury itd, a potem

sklejamy je w większą całość

System operacyjny

1. System operacyjny jest to zbiór procedur (programów) przekształcający maszynę rzeczywistą w wirtualną.
2. System operacyjny jest to zorganizowany zespół programów, które pełnią rolę pośredniczącą między sprzętem, a użytkownikami, dostarczają użytkownikom zestawu środków ułatwiających projektowanie, kodowanie,uruchamianie i eksploatację programów oraz w tym samym czasie sterują przydziałem zasobów dla zapewnienia efektywnego działania.

Zasoby sprzętowe: czas procesora, pamięć operacyjna, urządzenia we/wy, inne komputery
Zasoby programowe: funkcje systemowe dostarczone programom użytkownika, pamięć zewnętrzna, katalogi i pliki, translatory i kompilatory
Podział ze względu na tryby pracy:
- do przetwarzania wsadowego (offline, batch)
- z podziałem czasu (on-line)
- czasu rzeczywistego (real-time)

Maszyna wielopoziomowa:

- generator aplikacji
- język wyższego poziomu
- język wysokiego poziomu
- język assemblera
- system operacyjny
- kod maszynowy
- mikroprogram (występował w starszych komputerach teraz to chyba są drivery)
- sprzęt (ostatnie 2 to maszyna rzeczywista)