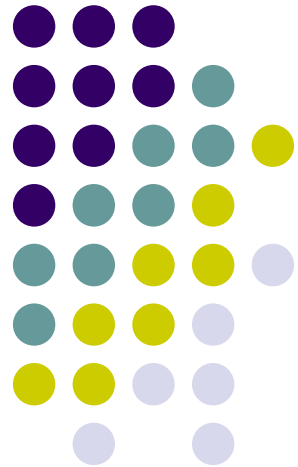
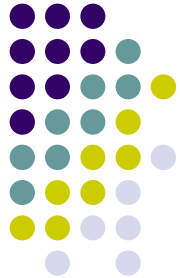
Teknik Kompiler 2

oleh: **antonius rachmat c, s.kom**

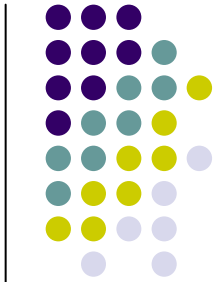


Definisi Bahasa Pemrograman



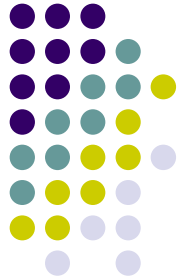
- *Function:* Bahasa pemrograman adalah bahasa yang digunakan untuk menulis program komputer, dimana dapat menginstruksikan komputer untuk melakukan seperti komputasi dan mengorganisasikan aliran kontrol antar perangkat yang ada.
- *Target:* Bahasa pemrograman berbeda dengan bahasa alami. Bahasa alami digunakan untuk berinteraksi antar manusia, sedangkan bahasa pemrograman digunakan untuk berkomunikasi antara manusia dengan komputer. Target bahasa pemrograman berbeda-beda. Misalnya Postscript adalah bahasa yang ditargetkan untuk mengkontrol printer dan tampilan.
- *Constructs:* Bahasa pemrograman berisi susunan definisi dan manipulasi struktur data dan pengkontrolan alir eksekusi program.
- *Expressive power:* Bahasa pemrograman adalah bahasa yang mengekspresikan kemampuan komputasinya berdasarkan suatu algoritma tertentu.

Pembagian Bahasa Pemrograman



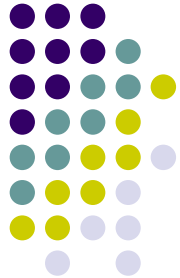
- Typed dan untyped Programming language
  - Typed: Harus dideklarasikan tipe datanya, tidak bisa menerima tipe data selain yang dideklarasikan. Contoh?
  - Untyped: Tidak harus dideklarasikan tipe datanya. Contoh: Assembly
- Static dan Dynamic Typing Programming Language
  - Lebih berdasarkan pada kompiler/interpreternya.
  - Static: pengecekan tipe dilakukan pada saat compile time
  - Dynamic: pengecekan tipe dilakukan pada saat run time
- Weak dan Strong Programming Language
  - Weak: suatu tipe data bisa diisi tipe data lain. Contoh C, tipe char bisa diisi tipe int.
  - Strong: tipe data harus sesuai.

Kriteria Bahasa Pemrograman yang baik



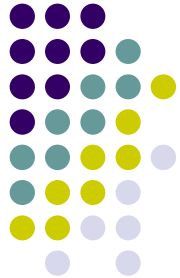
- Clarity, simplicity, dan unity
- Sesuai dengan tujuan pembuatannya
- Mendukung abstraksi
- Dukungan IDE
- Portabilitas
- Biaya minimum: biaya eksekusi, biaya translasi, biaya penciptaan, testing, biaya pemeliharaan

Sumber Perancangan Bahasa Pemrograman



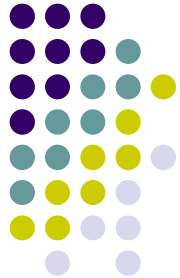
- Bahasa pemrograman dirancang agar manusia **bisa berkomunikasi dengan komputer**.
- Bahasa pemrograman bersumber dari bahasa alami (*natural language*), matematika maupun bahasa pemrograman yang sudah ada.

Dasar Pengelompokkan Bahasa Pemrograman



- Jika sumber berasal dari **bahasa alami** maka konstruksi bahasanya akan mirip dengan arti sebenarnya.
  - Hal itu tentu akan mempermudah pemrogram untuk memprogram, khususnya bagi yang belum berpengalaman.
  - Kadang dapat menjadi ambigu, karena kadang banyak bahasa alami yang memiliki arti banyak.
  - Contoh: **SQL**.

Dasar Pengelompokkan Bahasa Pemrograman (2)

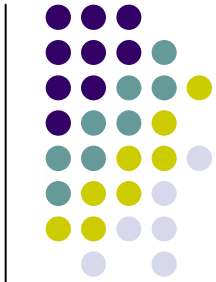


- Jika sumber berasal dari matematika maka bahasa akan menjadi sangat logis, matematis, dan penuh dengan konsep-konsep.

  Contoh: MATLAB, SciLab, MathCad

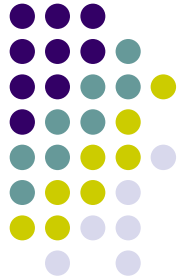
- Jika sumber berasal dari bahasa pemrograman yang sudah ada, maka pemrogram akan lebih mudah menggunakannya, terutama bila bahasa yang dibentuk berasal dari bahasa yang sudah sangat terkenal.
  - Perlu juga hati-hati karena bahasa yang sudah adapun kadang masih memiliki kekurangan/kesalahan yang serius.
    - Contoh: Java mengadopsi beberapa bahasa, seperti C, C++.
  - Biasanya bahasa yang lebih baru harus menambah fitur baru yang lebih bagus dan mengeliminasi kesalahan/kekurangan.

Tujuan Perancangan Bahasa Pemrograman



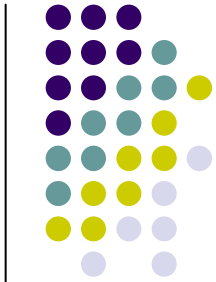
- **Untuk berkomunikasi dengan manusia**
  - Jika bahasa tidak dirancang baik, maka bahasa tersebut akan membingungkan pemrogramnya.
  - Suatu bahasa yang baik harus memiliki arti yang sesuai dengan semantiknya.
  - Perlu diingat: pemrogram masih jarang yang senang membuat dokumentasi.
  - Suatu ekspresi juga bisa bermakna ambigu:
    - `int x = y / k / c;`
    - `str := ‘1’ + ‘2’;`
    - `$str = 1 + 2;`
    - ```
      for (int i=0;i<10;i++)
         count++;
         System.out.println(“Hallo”);
      ```
      ```

Tujuan Perancangan Bahasa Pemrograman (2)



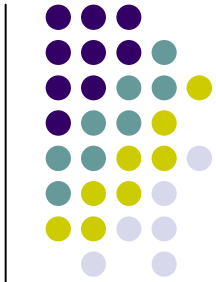
- **Untuk pendeteksian dan pencegahan kesalahan**.
- ***Usability****:* kenyamanan pemrogram untuk memprogram menggunakan suatu bahasa tertentu.
  - Suatu bahasa pemrograman yang baik harus mudah dipelajari dan diingat.
  - Bandingkan Pascal, C, C++, Java, VB. Mana yang lebih mudah dipelajari pemula? Apakah harus melihat help terus menerus?
- **Efektifitas**
  - Bahasa pemrograman yang baik harus dapat memfasilitasi suatu statemen yang jelas untuk suatu maksud tertentu.
  - Sebaiknya bahasa harus mampu membiarkan pemrogram menyatakan keinginannya dan mengahasilkan suatu hasil yang diinginkan.
  - **Kesulitan** : banyak pemrogram yang tidak bisa mengungkapkan keinginannya ke dalam bahasa program.

Tujuan Perancangan Bahasa Pemrograman (3)



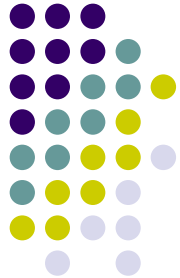
- ***Compilability***
  - Suatu bahasa yang baik harus mudah dikompilasi.
  - Tidak mempersulit proses kompilasi. Shortcut?
  - Kompleksitas dapat muncul karena adanya simbol yang memiliki arti yang berbeda-beda.
  - Misal tanda *parenthesis* (kurung buka/tutup):
    - Mengelompokkan sub ekspresi
    - Pembatasan argumen dalam fungsi / prosedur
    - Indeks pada array (VB)
    - Ekspresi matematika dan logika
    - Contoh skema yang sederhana tapi *powerfull* adalah *passing parameter by value* dan *by reference*.

Tujuan Perancangan Bahasa Pemrograman (4)



- ***Machine Independent**:* jika program telah sukses dikompilasi dan dieksekusi pada suatu mesin, saat dipindah ke mesin lainnya dapat berjalan dengan benar dan menghasilkan output yang tepat sama.
  - Tidak semua bahasa harus bersifat *machine independent*, ada yang harus spesifik sesuai dengan mesinnya.
  - Misalnya sistem operasi, bahasa aritmatika *floating point* (bdk. PASCAL?), karakter-karakter yang didukung (bdk. UNICODE?), dan karakter control (bdk. BELL?)

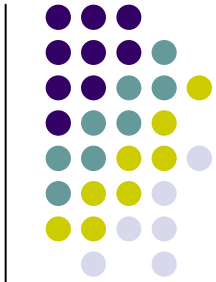
Tujuan Perancangan Bahasa Pemrograman (5)



- **Kesederhanaan (*Simplicity*)**
  - Memang bahasa yang sederhana baik dan mudah dipelajari, tapi kesederhanaan juga bisa berarti kelemahan disuatu sisi tertentu. Contoh: BASIC
- ***Uniformity***

  Suatu bahasa yang baik harus bersifat *uniform* (seragam) tidak ‘berbeda’ dari bahasa-bahasa yang sudah ada. Sehingga pemrogram akan lebih merasa familiar, cepat untuk diingat.

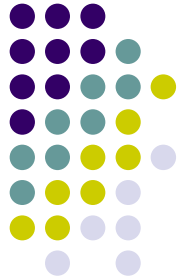
Detail Perancangan Bahasa Pemrograman



Microstructure

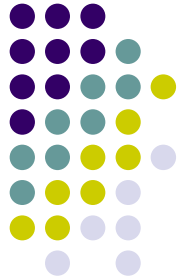
- Arti dari suatu konstruksi misalnya operator harus jelas wujudnya.
  - Dengan kata lain, token dari suatu bahasa harus mudah dikenali apa itu dan apa yang dilakukannya.
- Aspek terendah dari mikrostruktur adalah **set karakter** yang digunakan. Set karakter yang standar harus digunakan agar tidak terjadi hal yang tidak diinginkan pada saat ‘berpindah-pindah’.

Keywords (*microstructure*)



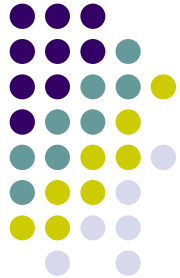
- *Keyword* harus jelas, mudah diucapkan, memiliki arti sesuai dengan bentuknya.
- *Keyword* harus dipilih sedemikian rupa agar tidak sama dengan variabel yang dipilih user. Hal yang perlu diperhatikan dalam memilih *keyword*:
  - *Keyword* adalah "*reserved*", tidak boleh dipakai user dalam membuat variabel, agar tidak terjadi tabrakan dan ambiguitas.
  - *Keyword* yang baik harus diawali dengan karakter khusus yang menadainya.

Detail Perancangan Bahasa Pemrograman (2)



- **Microstructure (2)**
- Aspek lain dari mikrostruktur adalah pengaturan komentar.
  - Suatu komentar adalah diawali dari suatu tanda simbol awal komentar sampai dengan ditemukannya tanda simbol tertentu sebagai akhir komentar, walaupun ada karakter apapun didalamnya termasuk spasi.
  - Simbol komentar idealnya
    - Terdiri dari 2 karakter, lebih baik barupa karakter yang sama
    - Simbolnya jarang digunakan.
    - Terdiri dari karakter yang berlokasi sama pada keyboard
- Dibedakan antara simbol komentar sebaris dan banyak baris (*multi line*). Simbol komentar *multi line* harus ditutup dengan simbol yang sama pula tanpa memperhatikan karakter apapun didalamnya.

Detail Perancangan Bahasa Pemrograman (3)



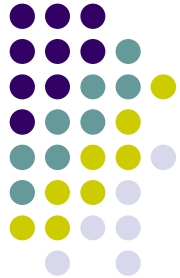
- **<u>Stuktur Ekspresi</u>**

  Struktur ekspresi berhubungan dengan urutan dan evaluasinya, bisa menggunakan :

  - *Explicit Bracketing*: menggunakan karakter `[,],{,},(,)`

  - Aturan dan urutan operator : kiri ke kanan, kanan ke kiri, dan prioritas

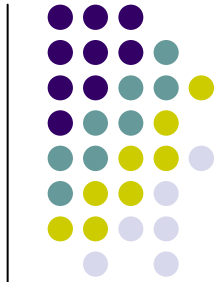
Detail Perancangan Bahasa Pemrograman (4)



Struktur Data

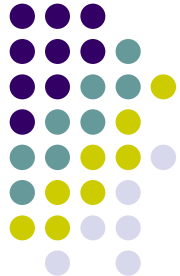
- Deklarasi data
  - Konstanta, tidak boleh berubah dalam program
  - *Type*, suatu pembentuk tipe data buatan baru
  - Variabel, bisa diubah dalam program
- Semuanya harus memperhatikan aspek *readibility*
- Perhatikan tipe data yang sudah disediakan oleh bahasa pemrograman

Tipe Data dalam Bahasa Pemrograman



- Ada beberapa kemungkinan:
  - Tidak ada sama sekali, contoh : *assembly*
  - *Soft typing*, ditentukan suatu tipe data yang bisa menerima nilai apapun. Contoh : bahasa scripting language, tipe data variant.
  - *Hard (Strong) Typing*, ditentukan suatu tipe data yang harus memuat satu jenis tipe data saja. Mudah dicari kesalahannya (Debug mudah). Contoh : C, C++, Java, Pascal.

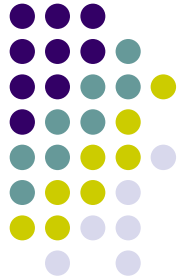
Jenis Tipe Data



- **Tipe data primitif**
  - Tipe data ini disediakan langsung oleh bahasa pemrograman yang ada. Contoh : integer, real, float, byte, char. Tipe data primitif ini dapat dikembangkan menjadi tipe data kompleks seperti enumerasi (set/himpunan).
    - Misal: Type gelar = (S1, S2, S3). Dapat juga berupa range dari suatu tipe data integer. Misal type tanggal = 1 to 31.
  - Tipe data karakter dan string juga termasuk tipe data primitif, namun tidak semua bahasa memiliki tipe data string seperti bahasa C dan C++.
  - Tipe data string adalah kumpulan (rangkaian) karakter-karaker yang dibentuk menjadi array. Tipe data karakter hanya bisa menerima 1 huruf sedangkan string banyak huruf. Pada C dan C++ string sama dengan char[]. Tipe data string harus memiliki panjang maksimum untuk keperluan penyimpanan di memori.
  - Tipe data terakhir adalah tipe data boolean. Tipe data ini tidak dimiliki semua bahasa, seperti C. Pada C, tipe data boolean bisa dibuat sendiri dengan menggunakan sintaks:

```
#define false 0
#define true 1
```

Jenis Tipe Data (2)



- **Complex data type**
  - Contohnya adalah tipe data seperti pointer. Pointer dapat membentuk tipe data record yang kompleks seperti Stack, List, dan Queue.
  - Ada beberapa bahasa yang juga menyediakan fungsi untuk mengubah tipe data dari tipe data tertetu ke tipe data lainnya. Contoh: Di Delphi ada fungsi `StrToInt()` atau di VB ada fungsi `val()`

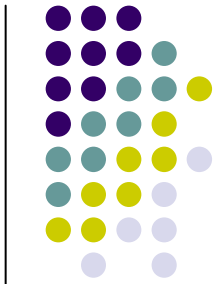
Jenis Tipe Data (3)

- **User Defined Data Type**
  - Tipe data ini dibuat sendiri oleh pemrogram. Contohnya adalah:

```
Type table = array [1..10] of integer;
Type mahasiswa = record
   Nama : String;
   NIM : String;
End;
typedef struct mahasiswa {
   char[30] nama;
   int umur;
};
Type warna = (merah, kuning, biru);
Type himpunan = set of warna;
```

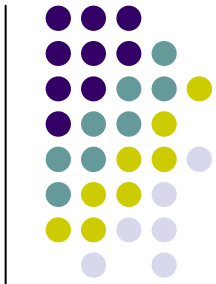
  - Tipe data seperti ini menambah kekompleksitasan kompilasi.

Jenis Tipe Data (4)



- Tipe data abstract (ADT)
  - Seperti UDT, tambah dengan kemampuan enkapsulasi, polymorfisme, dan ciri-ciri kemampuan tipe data berorientasi obyek lainnya.
  - Contoh: class

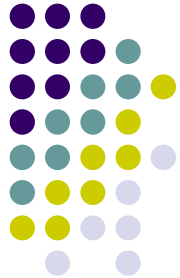
Strategi Penyimpanan



Alokasi variabel yaitu:

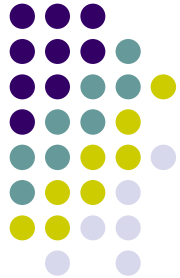
- Alokasi statik : pemakaian global
- Alokasi lokal, dinamik, dan otomatis : alokasi di dalam prosedur/fungsi
- *Retention* : dialokasikan di awal entry prosedur/blok tapi tidak dibebaskan pada saat keluar. Digunakan untuk *backtracking* dan rekursif

Detail Perancangan Bahasa Pemrograman (5)



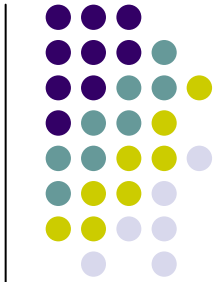
- **Struktur Kontrol dan Loop**
  - Jangan menggunakan GOTO/LABEL, jelek dan tidak teratur.
  - Struktur kontrol yang paling sederhana adalah blok (begin-end, { .. }).
  - Struktur kontrol yang terkenal adalah IF … THEN … ELSE …
  - Pada IF kita menggunakan ekspresi boolean, ingat boolean tidak ada di semua bahasa. Bisa menggunakan true/false, 0 dan 1, ganjil dan genap.
  - IF bisa dibentuk CASE/SWITCH. Hati-hati dengan CASE, ingat DEFAULT/ELSE untuk eksepsi kesalahannya.
  - Bentuk Loop : for, do..while, while..do, repeat..until
  - Bentuk prosedur / fungsi juga memiliki passing parameter by reference dan by value. Apa perbedaannya? Apa beda fungsi dan prosedur?

Detail Perancangan Bahasa Pemrograman (6)



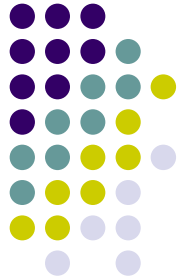
- Struktur Kompilasi
  - Menyangkut seluruh aspek pada saat kompilasi. Salah satu fasilitas kompiler adalah adanya fasilitas menyisipakan berkas. Ada juga yang memfasilitasi untuk memilih bagian tertentu dari teks untuk dikompile. Contoh : *Run to Cursor, go to definition, Run step-by-step*

Detail Perancangan Bahasa Pemrograman (7)



- Struktur I/O
  - I/O adalah struktur input dan output.
  - Ada 3 macam kemungkinan bentuk:
    - *Format-free form* : langsung ditampilkan sehingga mudah bagi user untuk memeriksa kebenaran program. Contoh pada VB.
    - *Formatted form* : output ditampilkan secara terformat, seperti di C : printf(), delphi/VB : format().
    - *File form* : output ditampilkan dalam file sekuensial, index sekuensial, dan direct.

Detail Perancangan Bahasa Pemrograman (8)



Skenenario Perancangan

1. Tentukan apa yang diinginkan.
2. Tentukan feature yang mungkin
3. Tentukan desain dan sesuaikan dengan featurenya
4. Tentukan rincian, interpret/compile, dan error checking.
5. Tuliskan user manual dan help.
6. Evaluasilah, jika salah mulai lagi dari langkah 3.
7. Jika sudah benar, optimisasilah dan uji segala kemungkinan.
8. Cobakan kepada pengguna, tunggu reaksinya.
9. Perbaiki bug dan mulai versi baru.

NEXT

- Regex 1