PROGRAM STUDI
TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

Mata Kuliah
Dasar Pemrograman

Fungsi

TIM DASAR PEMROGRAMAN
TEKNIK INFORMATIKA S1
UNIVERSITAS DIAN NUSWANTORO

Capaian Pembelajaran

1. Menjelaskan konsep dekomposisi dan abstraksi
2. Menjelaskan realisasi dekomposisi dengan suatu fungsi dan realisasi abstraksi suatu fungsi dan prosedur
3. Menjelaskan fungsi tanpa parameter, dengan parameter
4. Menjelaskan fungsi dengan pengembalian nilai

Bagaimana kita menuliskan Kode?

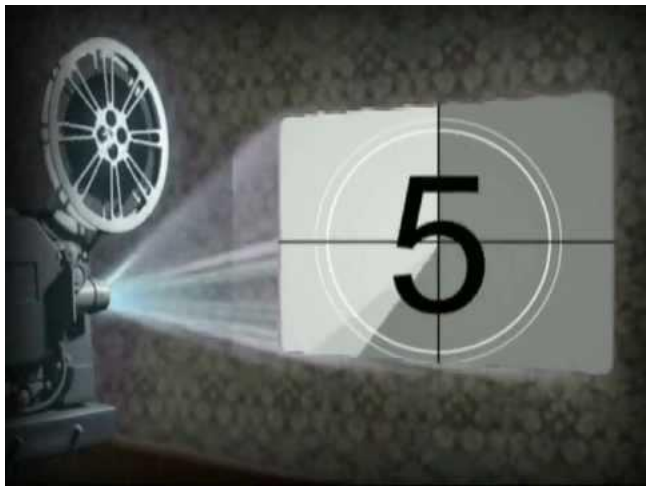
- Sejauh ini…
  - Menyinggung mekanik Bahasa pemrograman
  - Tahu bagaimana cara menulis program pada file yang berbeda pada setiap komputasi
  - Setiap file terdiri dari beberapa baris kode
  - Setiap kode berisi instruksi sekuensial
- Masalah yang timbul dari pendekatan ini:
  - Mudah bagi permasalahan dengan skala kecil
  - Kacau untuk permasalahan yang lebih besar
  - Sulit untuk tetap melacak secara detail (bagian kode program yang memiliki tugas khusus)
  - Bagaimana kita tahu informasi yang benar yang diberikan pada bagian kode yang benar

Memprogram yang Baik dan Benar

- Banyak koding yang ditulis (berbaris - baris) belum tentu sesuatu yang baik
- Untuk menghitung baik tidaknya programmer dengan menghitung banyaknya fungsionalitas yang berhasil diselesaikan
- Diperkenalkan suatu **fungsi**
- Fungsi merupakan mekanisme untuk mencapai **dekomposisi** dan **abstraksi**

Contoh - Proyektor

- Bayangkan suatu proyektor adalah kotak hitam (black box)
- Kita tidak tahu bagaimana alat tersebut bekerja
- Kita hanya tahu interface: input/output
- Kita bisa menghubungkan peralatan elektronik apapun padanya yang dapat berkomunikasi dengan input pada proyektor
- Black box: bagaimana caranya mengkonversi gambar dari sumber input ke dinding, dan memperbesarnya
- **IDE ABSTRAKSI**: kita tidak perlu tahu bagaimana proyektor bekerja untuk melakukan hal tersebut



Contoh - Proyektor

- Setiap proyektor mengambil input dan memproduksi output terpisah
- Semua proyektor bekerja bersama untuk memproduksi gambar yang lebih besar
- **IDE DEKOMPOSISI:** peralatan – peralatan yang berbeda bekerja bersama untuk mencapai satu tujuan
- LAKUKAN KONSEP ABSTRAKSI DAN DEKOMPOSISI INI PADA SAAT MEMPROGRAM

Membuat Struktur Pemrograman dengan Konsep Dekomposisi

- Pada contoh sebelumnya, ingat peralatan yang terpisah
- Pada pemrograman, pisahkan potongan kode pada suatu modul – modul
  - Yang berdiri sendiri (**self-contained**)
  - Yang digunakan memecah kode (**break up code**)
  - Yang dimaksudkan untuk digunakan kembali (r**eusable**)
  - Yang menjaga kode selalu terorganisir (**organized**)
  - Yang mempertahankan kode yang koheren (**coherent**), koheren menurut kbbi, keserasian atau kekompakan akibat dari adanya koordinasi
- Pada paradigma fungsional, kita akan menerapkan dekomposisi dengan fungsi
- Pada paradigma *object oriented*, kita akan menerapkan dekomposisi dengan class

Merinci Detail Program dengan ABSTRAKSI

- Pada contoh proyektor, instruksi – instruksi dari bagaimana kita menggunakan proyektor mencukupi, kita tidak perlu tahu untuk bagaimana membuat proyektor
- Pada pemrograman, pikirkan potongan kode sebagai suatu black box, dimana:
  - Kita tidak dapat melihat detail – detail
  - Tidak butuh untuk melihat detail – detail
  - Tidak ingin untuk melihat detail – detail
  - Menyembunyikan detail koding
- Terapkan abstraksi dengan spesifikasi fungsi atau docstring

Fungsi

- Menuliskan beberapa potongan kode yang dapat digunakan kembali, disebut dengan fungsi
- Fungsi hanya akan berjalan pada program yang kita buat dan mereka dipanggil (invoked) pada program kita
- Karakteristik fungsi:
  - Memiliki **nama**
  - Memiliki **parameter** (bisa jadi 0 atau lebih)
  - Memiliki **docstring** atau spesifikasi fungsi (opsional tetapi sangat direkomendasikan)
  - Memiliki **badan fungsi**
  - **Return value**, atau mengembalikan sesuatu (bisa jadi mengembalikan sesuatu yang kosong)

Notasi Algoritma untuk Suatu Fungsi

Fungsi NamaFungsi (<parameter>) □ <tipe hasil>

{Spesifikasi fungsi}

Kamus lokal

{Semua variabel-variabel yang dipakai dalam algoritma dari fungsi yang tidak bisa dipaggil diluar fungsi}

Algoritma

{Isi atau badan fungsi, dimana suatu algoritma akan dituliskan disini}

{perhatikan bahwa nilai akhir fungsi harus sesuai dengan tipe data dari hasil}

□ hasil

Contoh Notasi Algoritma fungsi is_genap()

Fungsi isGenap(i:integer) -> boolean

{diberikan suatu bilangan i dengan tipe integer untuk mengecek apakah bilangan tersebut bilangan genap atau bukan}

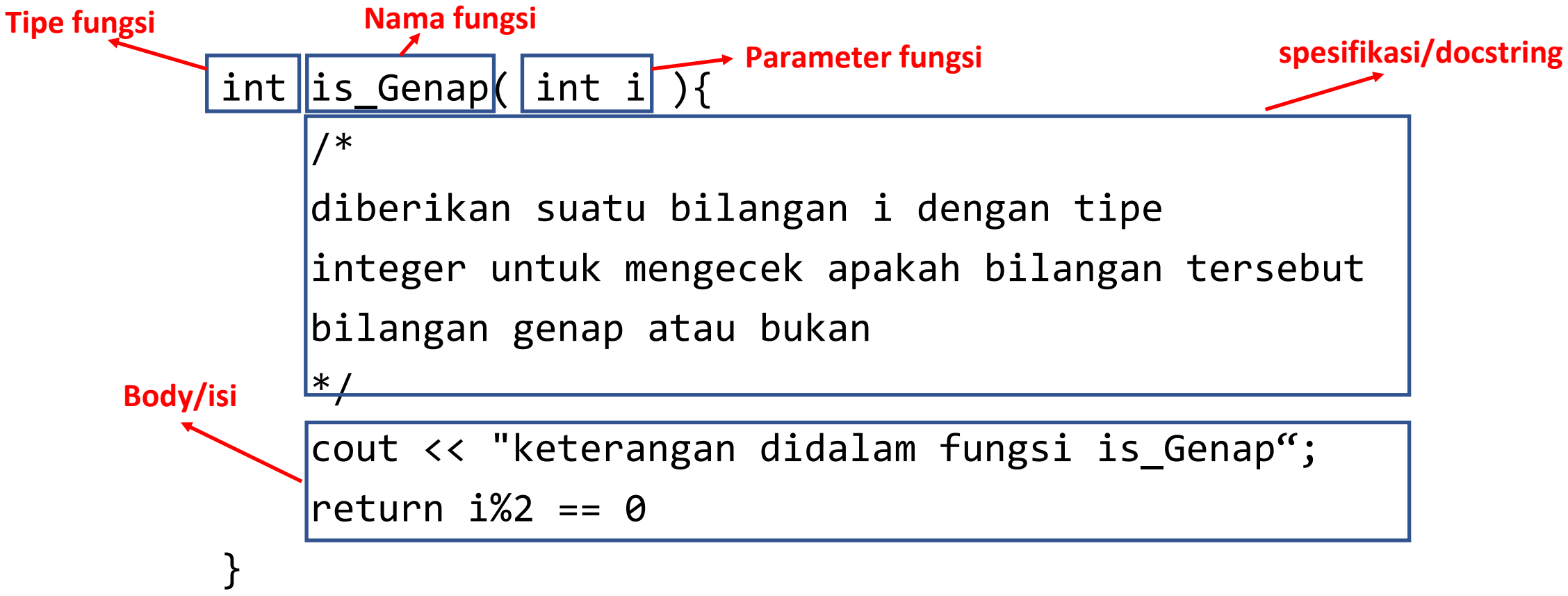
Kamus lokal

Algoritma

output("keterangan didalam fungsi is_Genap")

-> i % 2=0

Bagaimana cara Menulis fungsi pada C++



Perhatikan pada Body Fungsi

```
int is_Genap( int i ){
    /*
    diberikan suatu bilangan i dengan tipe
    integer untuk mengecek apakah bilangan tersebut
    bilangan genap atau bukan
    */
    cout << "keterangan didalam fungsi is_Genap“;
    return i%2 == 0
}
```

Beberapa Baris statement

Keyword untuk nilai pengembalian

Ekspresi untuk menghasilkan nilai yang akan dikembalikan. Kita bisa menampung ekspresi pada variabel, kemudian kembalikan nilai dari variabel tersebut, contoh:

```
a = i%2 == 0
return a
```

Cakupan Variabel (Variable Scope)

- **Parameter formal** untuk mengikat nilai dari **parameter actual** ketika fungsi di panggil
- **Cakupan/scope/frame/environment** yang baru akan diciptakan ketika sudah masuk dalam fungsi
- **Scope** memetakan nama ke objek

```
int f( x ){
          int x;
          x = x + 1;
          cout <<“in f(x): x = ” << x;
          return x;
}

int main(){
          int x, z;
          x = 3;
          z = f( x );
}
```

Definisi fungsi

Main program

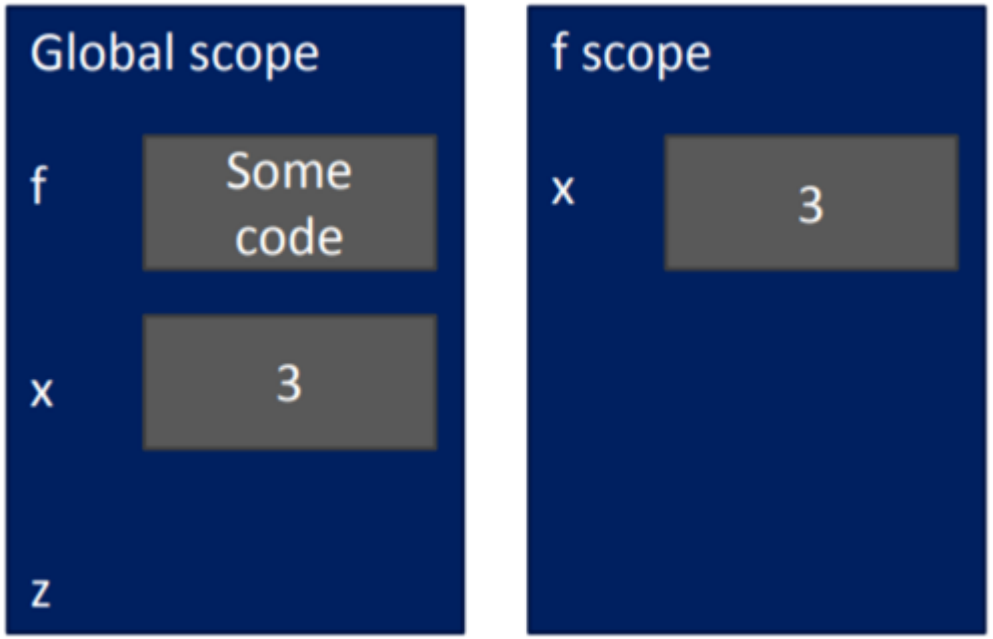
1. Inisialisasi variabel x

2. Panggil fungsi f dengan parameter dari variabel x

3. Assign nilai pengembalian dari fungsi pada variabel z

Variable Scope

```
int f( x ){
        int x;
        x = x + 1;
        cout <<“in f(x): x = ” <<
x;      return x;
}

int main(){
        int x, z;
        x = 3;
        z = f( x );
}
```

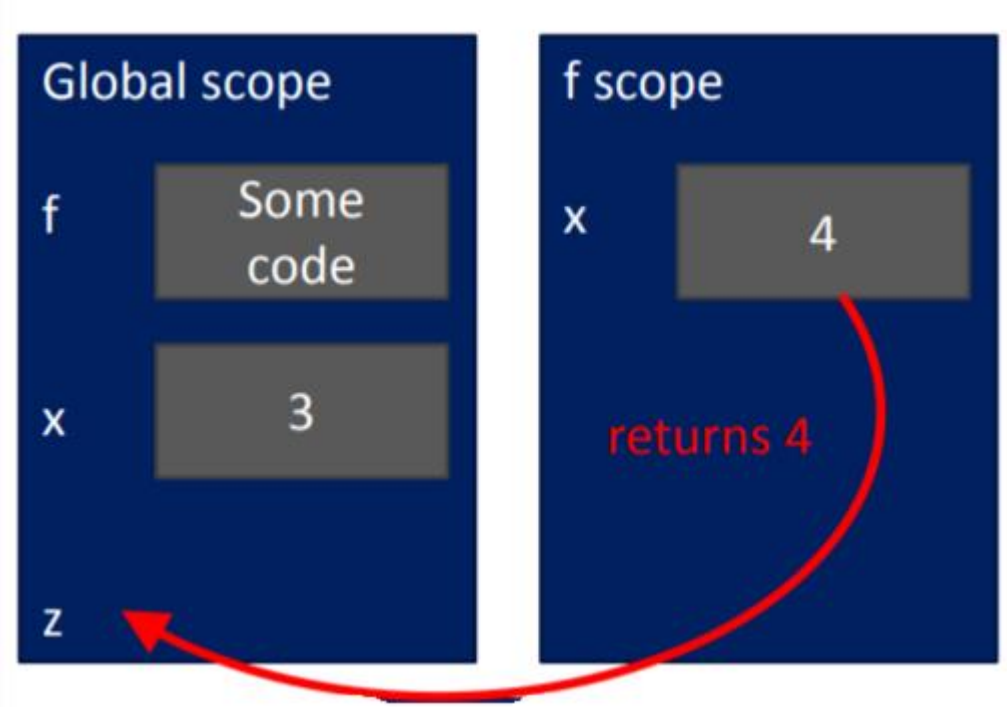


Variable Scope

```
int f( x ){
        int x;
        x = x + 1;
        cout <<“in f(x): x = ” <<
x;      return x;
}

int main(){
        int x, z;
        x = 3;
        z = f( x );
}
```

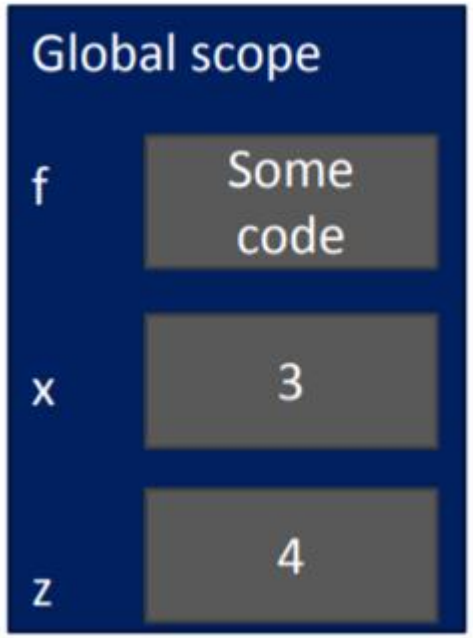
Variable Scope

```
int f( x ){
        int x;
        x = x + 1;
        cout <<“in f(x): x = ” <<
x;      return x;
}

int main(){
        int x, z;
        x = 3;
        z = f( x );
}
```



Variable Scope

```
int f( x ){
        int x;
        x = x + 1;
        cout <<“in f(x): x = ” <<
x;      return x;
}

int main(){
        int x, z;
        x = 3;
        z = f( x );
}
```



Referensi

Utama:

1. Bjarne Stroustrup, 2014, Programming: Principles and Practice Using C++ (Second Edition), Addison-Wesley Professional

Pendukung:

1. Introduction to Computer Science and Programming in Python, MIT https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016

2. Introduction to Computer Science and Programming, MIT https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00sc-introduction-to-computer-science-and-programming-spring-2011/index.htm

TERIMA KASIH

ANY QUESTIONS?