ABSTRACT DATA TYPE (ADT)

Abstract Data Type (ADT)

- ADT adalah koleksi data dan operasi yang dapat digunakan untuk memanipulasi data tersebut

- Dalam C++, ADT dapat dibuat dalam sebuah Class. Class dalam C++ merupakan pengembangan dari *struct* dalam bahasa pemrograman C.

- Class memiliki data dan fungsi. Data dan fungsi yang dideklarasi *private* tidak dapat diakses secara langsung oleh *client*, sementara data dan fungsi yang dideklarasi *public* akan bersifat publik dan dapat diakses oleh *client* secara langsung.

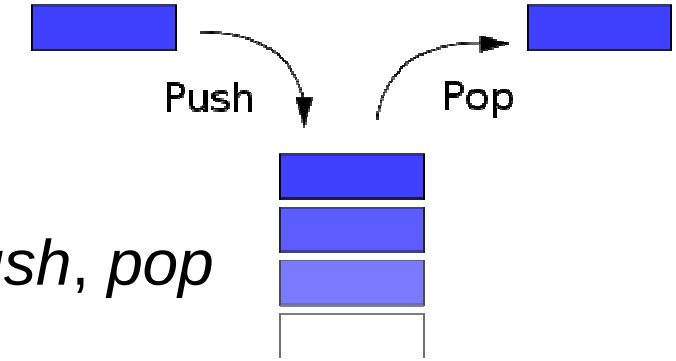
Contoh ADT

- Tipe jadi (*built-in*): boolean, integer, real, array, dll
- Tipe buatan (*user-defined*): stack, queue, tree, dll

- ADT Built-in:
  - Boolean
    - Nilai: true dan false
    - Operasi: and, or, not, xor, dll
  - Integer
    - Nilai: Semua bilangan
    - Operasi: tambah, kurang, kali, bagi, dll

Contoh ADT

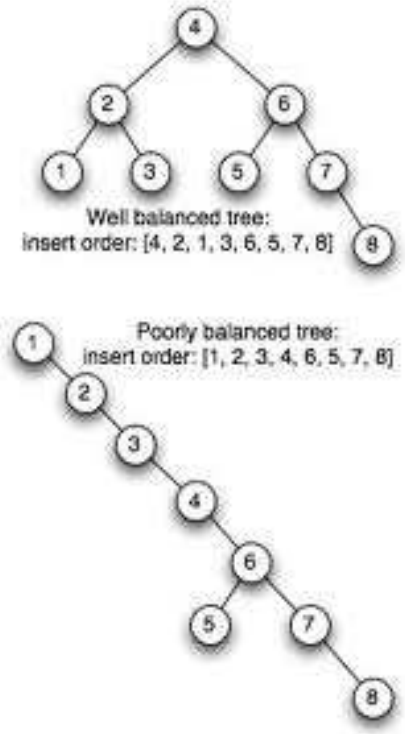
- ADT buatan (user-defined) :
  - Stack (tumpukan)
    - Nilai : elemen dalam stack
    - Operasi: *create*, *destroy*, *push*, *pop*



  - Queue (antrian)
    - Nilai: elemen dalam Queue
    - Operasi: *create*, *destroy*, *enqueue*, *dequeue*, dll

Contoh ADT

- Tree (pohon)
  - Nilai: elemen dalam pohon
  - Operasi: *insert*, *delete*, *find*, *traverse*



Class dan Struct

- Class atau Struct memiliki *member*. Setiap member memiliki nama dan tipe. Class atau Struct boleh memiliki member dengan tipe yang beragam. Oleh karena itu, Class atau struct dapat digunakan untuk membuat tipe data aggregate yang rumit.

```
struct point {
   double x, y;
};
```

- Dalam C++, nama struct (*tag*) adalah tipe. Oleh karena itu, `point` merupakan tipe. Pendeklarasian di atas dapat diibaratkan sebagai template dan tipe data `point` itu sendiri belum dialokasikan dalam memori.

Class dan Struct: Lanjutan

- Pendeklarasian:

```
point pt;
```

  mengalokasikan memori untuk variabel `pt` bertipe `point`

- Untuk mengakses member dari struct `pt`, operator *dot* harus digunakan.

- Contoh:

```
pt.x = -1;
pt.y = 0.5;
```

Class dan Struct: Lanjutan

- Nama member harus unik dalam skop struct. Karena saat mengakses member, nama struct harus ditulis, maka pemanggilan nama member yang sama dari struct yang berbeda tidak menjadi masalah.

```
struct fruit {
   char name[15];
   int calories;
};

struct vegetable {
   char name[15];
   int calories;
};
```

Class dan Struct: Lanjutan

```
fruit     a;
vegetable b;
```

Dalam bahasa C:

```
struct fruit         a;
Struct vegetable     b;
```

Pemanggilan `a.calories` dan `b.calories` tidak akan menimbulkan masalah.

Contoh:

```
struct point {
  double x, y;
};
point average(const point* d, int size){
  point sum = {0, 0};
  for (int i = 0; i < size; i++) {
    sum.x += d->x;
    sum.y += d->y;
    d++; // d adl iterator (pointer) ke point
  }
  sum.x = sum.x / size;
  sum.y = sum.y / size;
  return sum;
}
```

Contoh:

```
int main(){

  point data[5] = { {1.0, 2.0}, {1.0, 3.3},
                    {5.1, 0.5}, {2.0, 2.0},
                    {0, 0} };

  point avg_point;

  avg_point = average(data, 5);

  cout << "Average point = (" << avg_point.x
       << ", " << avg_point.y << ") " <<
       endl;
}
```

Fungsi Sebagai Member

- C++ membolehkan fungsi sebagai member dari struct, sementara dalam C tidak. Dalam C, hanya data yang bisa menjadi member dari sebuah struct.

- Untuk membuat fungsi sebagai member dari sebuah struct maka pendeklarasian fungsi dimasukkan dalam struct tersebut. Idenya adalah, bila sebuah fungsi dibutuhkan oleh sebuah struct untuk melakukan hal tertentu maka fungsi tersebut dibuat dan dideklarasikan dalam struct tersebut.

Contoh:

```
struct point {

  double x, y;

  void print() const {
    cout << "(" << x << ", " << y << ")";
  }

  void set(double u, double v) {
    x = u; y = v;
  }

};
```

Contoh:

```
int main(){

  point w1, w2;

  w1.set(0, 0.5);
  w2.set(-0.5, 1.5);
  cout << "\npoint w1 = ";
  w1.print();

  cout << "\npoint w2 = ";
  w2.print();

  cout << endl;
}
```

Struct: Private dan Public

- *Private* membuat data dan fungsi hanya dapat diakses di dalam struct itu saja, sedangkan *Public* membuat data dan fungsi dapat diakses dari luar struct.

```
struct point {
  pulic:
    void print() const {
      cout << "(" << x << "," << y << ")";
    }
    void print(const string& name) const;
    void set(double u, double v){
      x = u; y = v;}
    void plus(point c);
  private: double x, y;
};
```

Private dan Public

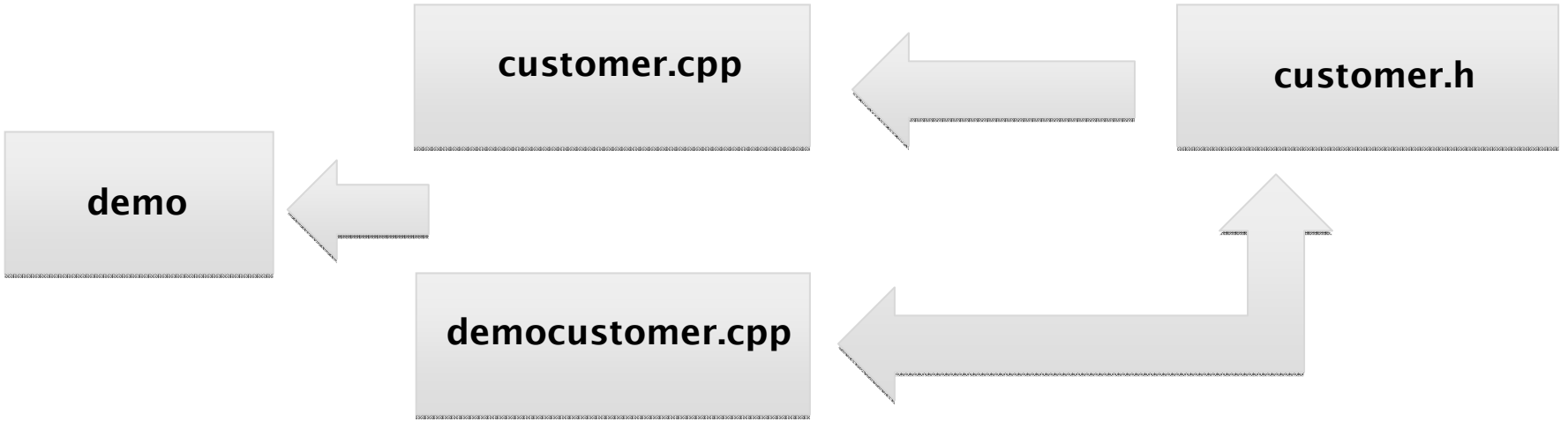
```
void foo(point w){
   ...
   cout << " x coord = " << w.x;     //error
   ...
}
```

Class: Private dan Public

- Class hampir sama dengan Struct. Bila struct secara default memiliki data dan fungsi yang public, Class memiliki data dan fungsi yang bersifat *private.*

```
class point {
  double x, y;   // secara implisit private
  pulic:
    void print() const {
      cout << "(" << x << "," << y << ")";
    }
    void print(const string& name) const;
    void set(double u, double v){
      x = u; y = v;}
    void plus(point c);
};
```

Contoh: Class



/> c++ -Wall -o demo democustomer.cpp customer.cpp

Download Code: customer.cpp, democustomer.cpp, customer.h

Tugas Bacaan:

Abstract Data Type

C++ Programming Tutorial Lesson 8: Struct
http://www.learn-programming.za.net/programming_cpp_learn08.html

C++ Programming Tutorial Lesson 10 : Class
http://www.learn-programming.za.net/programming_cpp_learn10.html