Standard Containers

- C++ has container classes
  - `vector` (dynamic array, random access)
  - `map` (associative array of key-value pairs)
  - `list` (linked list)
  - `string` (array of characters)
  - ...

Standard Containers

• Reduces memory management headaches
  – Avoid new/delete as much as possible
  – Compare:

```cpp
vector<int> a(10);  // releases memory when destroyed
int * b = new int[10];  // does not release memory
```

– Look up: Resource Acquisition Is Initialization
std::vector<T>

// vector of integers
vector<int> numbers;
numbers.push_back(1);
numbers.push_back(2);

for (size_t i = 0; i < numbers.size(); ++i)
    cout << numbers[i] << endl;

// vector of strings
vector<string> names;
names.resize(10);
for (size_t i = 0; i < names.size(); ++i)
    getline(cin, names[i]);
std::map<KeyT, ValueT>

// map names to ages
typedef map<string, int> NameToAgeMap;
NameToAgeMap ages;

ages["Sid"] = 304;
ages["Ohad"] = 3;

// look up a value
std::cout << ages["Ohad"] << std::endl;

// better: first check if value is in map
NameToAgeMap::const_iterator loc = ages.find("Ohad");
if (loc != ages.end())
    std::cout << "Ohad's age is " << loc->second << std::endl;
else
    std::cout << "Ohad skipped the census" << std::endl;
std::map<KeyT, ValueT>

// print out all names in map
for (NameToAgeMap::const_iterator iter = ages.begin();
     iter != ages.end();
     ++iter)
{
    cout << iter->first << "'s age is " << iter->second << endl;
}

// erase key-value pair from map
ages.erase("Sid");

// erase all key-value pairs from map
ages.clear();
```cpp
std::string

string a = "Hello";
string b = "World";
string hello_world = a + " " + b;

cout << a.substr(2) << endl;    // prints "llo"
cout << b.substr(2, 2) << endl; // prints "rl"

size_t first_vowel = a.find_first_of("aeiouAEIOU");

ostringstream buffer;
buffer << "My name is " << name
       << " and my age is " << age;
cout << buffer.str();
```
Warning! Warning! Warning!

static OSStatus
SSLVerifySignedServerKeyExchange(...)
{
    OSStatus err;

    ...

    if ((err = SSLHashSHA1.update(...) != 0)
        goto fail;
    goto fail;
    goto fail;
    if ((err = SSLHashSHA1.final(...) != 0)
        goto fail;
    goto fail;

    ...

    fail:
    ...
    return err;
}
Warning! Warning! Warning!

```c
static OSStatus
SSLVerifySignedServerKeyExchange(...)
{
    OSStatus err;

    ...

    if ((err = SSLHashSHAI1.update(...) != 0) != 0)
        goto fail;
    goto fail;
    if ((err = SSLHashSHAI1.final(...)) != 0)
        goto fail;
    
    ...

    fail:
    ...
    return err;
}
```

Always executes
Warning! Warning! Warning!

```c
static OSStatus
SSLVerifySignedServerKeyExchange(...) 
{
    OSStatus err;

    ...

    if ((err = SSLHashSHA1.update(...) != 0)
        goto fail;
    goto fail;
    goto fail;

    if ((err = SSLHashSHA1.final(...)) != 0)
        goto fail;

    ...

fail:
    ...
    ...
    return err;
}
```

Cost of not turning on/not paying attention to compiler warnings: $$$
Warning! Warning! Warning! Warning!

• Do NOT ignore:
  – signed/unsigned mismatch
  – truncation/loss of precision
  – “variable may be uninitialized”
  – function does not always return a value
  – unreachable code
  – variable declared but never used
  – ...

...
R2Pixel p(0, 0, 0), q(1, 1, 1), sum;

// first way to add
sum.SetRed (p.Red()   + q.Red()  );
sum.SetGreen(p.Green() + q.Green());
sum.SetBlue (p.Blue()  + q.Blue() );
Operator Overloading

R2Pixel p(0, 0, 0), q(1, 1, 1), sum;

// first way to add
sum.SetRed(p.Red() + q.Red());
sum.SetGreen(p.Green() + q.Green());
sum.SetBlue(p.Blue() + q.Blue());

// second way to add
sum = p + q;
Operator Overloading

R2Pixel p(0, 0, 0), q(1, 1, 1), sum;

// first way to add
sum.SetRed (p.Red() + q.Red());
sum.SetGreen(p.Green() + q.Green());
sum.SetBlue (p.Blue() + q.Blue());

// second way to add
sum = p + q;

Which is easier and more readable?
Modularity

Functions exist for a reason

double pqx = p.X() - q.X();
double pqy = p.Y() - q.Y();
double dist_pq = sqrt(pqx * pqx + pqy * pqy);
double uvx = u.X() - v.X();
double uvy = u.Y() - v.Y();
double dist_uv = sqrt(uvx * uvx + uvy * uvy);

vs

double dist_pq = distance(p, q);
double dist_uv = distance(u, v);
But please avoid this
... and also this

THE MOST INTERESTING MAN IN THE WORLD

HE SPEAKS FLUENT FRENCH, IN RUSSIAN
Every time you unnecessarily use one of these...

\[
\begin{align*}
\text{sqrt}(x) & \quad \text{sin}(x) \\
\text{cos}(x) & \quad \text{pow}(x) \\
\text{exp}(x) & \quad \text{log}(x) \\
\ldots
\end{align*}
\]
... somebody is mean to one of these
F'r instance...

double length = sqrt(x * x + y * y);
weight = f(length * length);

x_squared = pow(x, 2);

R2Vector normal_vector(cos(M_PI/2) * x - sin(M_PI/2) * y, sin(M_PI/2) * x + cos(M_PI/2) * y);

(vs normal_vector = R2Vector(-y, x))
Caveats

- Premature optimization is the source of all evil
- Don't sacrifice readability at the altar of optimization (obfuscation)
Pop Quiz

• What does this print?

```cpp
double a = 12;
cout << (2/3) * a << endl;
```
Pop Quiz

• What does this print?

double a = 12;
cout << (2/3) * a << endl;

Output: 0
Pop Quiz

• What does this print?

```cpp
double a = 12;
cout << (2/3) * a << endl;
```

Output: 0

Beware integer division
Pop Quiz

• What does this print?

```cpp
double a = 12;
cout << (2/3) * a << endl;
```

Output: 0

Beware integer division

Note: `rand() / RAND_MAX == 0`
Don't

- Recompute expensive invariants within loop or multiple times in same expression (e.g. isGray)
- Iterate through entire image to find pixels within a window.
- Round coordinates to integers before calling Resample(x, y, sampling_method).
- Create "dynamic arrays on stack" like "int x[n];" (this is valid C99, not C++: http://stackoverflow.com/questions/1204521/dynamic-array-in-stack)
- Multiply pixel by 0 instead of setting to zero (can be optimized away but doesn't help readability or performance).
Don't

- For B&W conversion: compute luminance, set red to luminance, recompute luminance, set green to luminance...

- Use ternary operator as "if", e.g.
  ```
  int a = ..., x;
  a == 0 ? x = b : x = c;
  cout << x << endl;
  ```
  (the ternary operator in general is perfectly ok, but not as used above, please)

- Use hardcoded integers instead of enums in case statements, e.g.
  ```
  enum { BLACK, WHITE } color;
  switch (color) { case 0: ...; case 1: ... }
  ```

- Submit executables and intermediate build products.

- Use Microsoft Word to edit a webpage :) (ok there's no real problem here but it makes it difficult for us to run the HTML through scripts for evaluation)
Useful Resources

- C++ FAQ
  http://www.parashift.com/c++-faq/

- C++ Reference
  http://www.cplusplus.com/reference/