

# Types

## ■ Basic types in C

- `char`, `short`, `int`, `long`
- `float`, `double`

## ■ Types added in C++ standard

- `wchar_t`: largest supported char (e.g., 8, 16, 32 bits)
- `bool`: `true` or `false`
- `long long`: 64-bit integer
- `long double`: longer precision than `double`

# Initialization

```
#include <iostream>
using namespace std;

int main() {
    int a = 5;           // c-like init
    int b(3);           // constructor init
    int c{2};           // uniform init

    cout << a << b << c << endl;
}
```

3 ways of variable initialization

# Type Deduction

```
int a = 0;
auto b = a;           // int b
auto c = 1.0;        // double c

decltype(a) d;       // the same type as a
```

`auto`: the compiler automatically figures out types of variables

# Typed Constant

```
const double pi = 3.14159265359;  
const char tab = '\t';  
  
double r = 10.0;    // radius  
  
double area = pi * r * r;  
double circle = 2 * pi * r;  
  
cout << area << tab << circle << endl;
```

Typed constants are natural to strongly-typed languages

# String

- **std::string**

- #include <string>

- **C-style string (not recommended)**

- null (/0) terminated character array
- strncpy(), strncat(), strlen()

```
#include <string>
...
std::string str1("C++ string");
std::string str2 = "another way of init";

char str3[] = "c-style string";
```

# String Manipulation

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s1("one");
    string s2("two");
    string s3;

    s3 = s1;           // string copy
    s3 = s1 + s2;    // string concatenation
    cout << "s1 + s2 = " << s3 << endl;
    cout << "s3 length = " << s3.length() << endl;
    cout << s3[0] << " " << s3[3] << endl;
}
```

string copy (=), string concatenation(+), string length(str.length())

# std::vector

## ■ std::vector - dynamic array

- #include <vector>
- a[idx], a.at(idx)
- a.size()

```
#include <vector>
...
std::vector<int> a1 = { 0, 1, 3 };
std::vector<double> f1 = { 0.1, 3.14 };

a1[2] = 2;           // element access as if array
f1.at(0) = 1.44;    // or use at()

cout << "size = " << a1.size() << endl;
```

# std::vector – stack

- **push\_back(), pop\_back()**
  - push/pop an element on/off the stack
- **back()**
  - return value at the top of the stack

```
vector<int> stk;

stk.push_back(0);    // { 0 }
stk.push_back(1);    // { 0, 1 }
stk.push_back(2);    // { 0, 1, 2 }

cout << stk.back() << endl;

stk.pop_back();     // { 0, 1 }
stk.pop_back();     // { 0 }
```



# std::vector – vector of vector

## ▪ vector<vector <int>>

- It can consider as 2-dimension array

```
#include <stdio.h>

int main(int argc, char *argv[]){
    int c_2d_array[2][2]={{1,2},{3,4}};
    int *c_array;

    c_array = c_2d_array[1];

    printf("%d\n", c_array[0]);

    return 0;
}
```

C-style

```
#include <iostream>
#include <vector>
using namespace std;

int main(int argc, char *argv[]){
    vector<vector <int>> cpp_2d_vector;
    vector<int> cpp_vector;

    cpp_2d_vector.push_back({1, 2});
    cpp_2d_vector.push_back({3, 4});
    cpp_vector = cpp_2d_vector.at(1);

    cout << cpp_vector.at(0) << endl;
    return 0;
}
```

C++-style

# std::vector – other functions

- <http://www.cplusplus.com/reference/vector/vector/?kw=vector>
  - Iterators – v.begin(), v.end(), ...
  - Capacity – v.size(), v.resize(), ...
  - Element Access – v.at(size\_type n), v.front(), ...
  - Modifiers – v.push\_back(), v.pop\_back();
  - Allocator – v.get\_allocator()

# [Lab – Practice #1(15%)]

- **Split a string line into words and print them in reverse order (use std::vector)**
  - Input: string of words separated by space
  - Output:
    - Original input string
    - One word per line, in reverse order

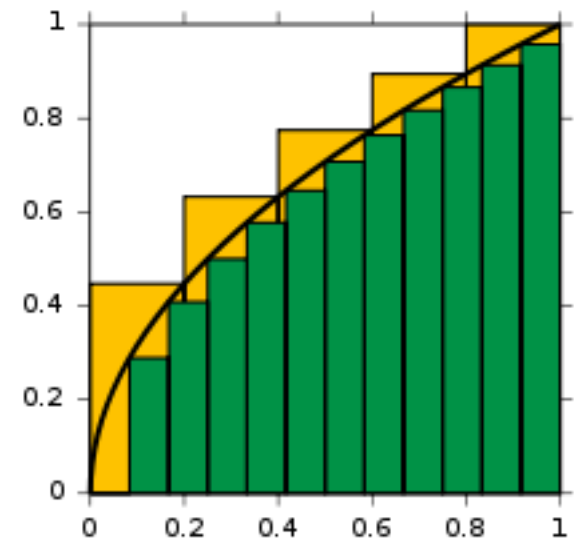
```
$ ./rssplit
input: Test my skill
skill
my
Test
```

```
// use string stream in Lec#1
// #include <sstream>
```

# [Lab – Practice #2(35%)]

- **Implement approximation of integral for quadratic equation**
  - use `std::vector<double>` for areas of partitions
  - use value of left-end for height of each partition
  - Output : integral (sum of green rectangles)

```
$ ./qeqintegral  
ax^2 + bx + c? [a b c]: 1 0 0  
range? [begin, end]: 0 10  
number of partitions? [positive]: 5  
integral = 240
```



# [Lab – Practice #3(50%)]

## ▪ Sentence to paragraph

- Output : Paragraph
- Rules
  - Input : # of words & word list & one string
  - 1. Find each word in sentence and make space each word.
  - 2. Careful of '.' you must space after period.
  - 3. First word after period should be capitalized first letter.
  - 4. Submit only .cpp file to mwahn402@gmail.com

```
$ ./para < input.txt > output.txt
```