SSE2034:

System Software Experiment 3

Spring 2016

Jinkyu Jeong (jinkyu@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

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

Type Check

```
// typeid
#include <iostream>
#include <typeinfo>
using namespace std;

int main () {
  int * a,b;
  a=0; b=0;
  if (typeid(a) != typeid(b))
  {
    cout << "a and b are of different types:\n";
    cout << "a is: " << typeid(a).name() << '\n';
    cout << "b is: " << typeid(b).name() << '\n';
  }
  return 0;
}
```

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

Type Conversions

```
double x = 10.3;
short y = 1234;
int z;

z = y;          // implicit conversion

y = int (x);     // functional notation
z = (int) x;     // c-like cast notation
```

Type Casting

```
dobule a;
int b = static_cast(a);   // alike to c-like cast

const int c = 10
int d = const_cast(c); // const type cast

int *pa;
char *pb;
pa = reinterpret_cast<int *>(123); // pointer type cast
pb = reinterpret_cast<char *>(pa);
```

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 }
```

[Lab - Practice #1]

- **Split a string line into words and print the m 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]

- **Implement approximation of integral for q uadratic 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)

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