CS11600-1
Intro to Programming II

Announcement:

- **Newsgroup uchi.cs.cs11600 has been created.**

Plan

- Last Time
  - Pointers & References. Function Arguments (Section 3.1, 3.3)
  - Namespaces (Section 3.6)
- Today
  - I/O Operations (Examples)
  - String Operations (Examples)
  - Dynamic Allocation of Memory (Section 3.7)

I/O operations

Streams

- Output
  - <<
  - flush
  - setf, unsetf (ios_base::showbase, ios_base::oct)
  - *cout << 1234 << "," << hex << 1234 << "," << oct << 1234 << endl;*
- Input
  - >>
  - get ()
  - get (p, n, term)
  - getline (p, n, term)
  - setf, unsetf

Streams

```
#include <iostream>
#include <fstream>
#include <string>
int main (){
int counter[LETTER_COUNT];
  ifstream infile( "input.txt" );
  // Check if the open operation was succesful
  if (!infile)
      { cerr << "Error: unable to open input file!\n"; return -1; }
  // Initialize the array of counters
  for (int i = 0; i < LETTER_COUNT; i++) counter[i] = 0;
  // Read words one by one and count their content
  string word;
  while (infile >> word)
    for (int i = 0; i < word.size(); ++i)
      if ((word[i] - 'a' >= 0) && (word[i] - 'a' < LETTER_COUNT))
        counter[word[i] - 'a']++;
  // Print the result
  for (int i = 0; i < LETTER_COUNT; i++)
    cout << static_cast<char> ('a' + i) << "   " << counter[i] << endl;
  return 0;
}
```

FILE

```
#include <stdio.h>

int main (int argn, char *argv[])
{
 FILE *pf;
 if (pf = fopen (argv[1]))
   {
     char buffer[1024];
     do {
          int len = fgets (buffer, 1024, pf);
          write (stdout, buffer, len);
     } while (len > 0);
     fclose (pf);
}
```

String operations

- C++ style
- Inherited zero-terminated C string style

C++-style

string type (string):

length == size
empty
[ ]
at
+=
=
append
insert
replace
find
compare
c_str

C++-style

```
#include <string>
int main ()
{
    char *p = "aaaaaaaaaaaaaaaaaaa";

    string s0, s0 = "";

    string s1 = 'a'; // error
    string s2 = 7;  // error
    string s3 (7);  // error

    string s4 (7, 'a'), s5 = "Frodo", s6 = s5;

    string s7 (s5, 3, 2), s8 (p + 7, 3), s9 (p, 7, 3);

    s7 += s8; s7 += p;

    cout << s7[10] << " from: " << s7.c_str () << endl;
}
```

C-style

string.h:

```
...
 char *strcpy (char* p, const char* q);
 char *strcat (char* p, const char* q);

 char *strncpy (char* p, const char* q, int n);
 char* strncat (char*p, const char* q, int n);

 size_t strlen (const char* p);

 int strcmp (const char* p, const char* q);
 int strncmp (const char* p, const char* q, int maxlen);
...
```

C-style

```
int main () {

    char *p1 = "aaaaaaaaa", *p2 = "bbbbbbbbbb";

    strcmp (p1, p2);
    strncpy (p1, p2, 3);

}
```

Dynamic Memory

- **Lifetime of a *static* declared object is defined during execution**
- **Sometimes is necessary to define objects with sizes and lifetime controlled by programmer (Ex: reading a string from stdin)**
  - ***not anymore a sustainable reason***
- **new / delete (C++)**
  - ***ptr = new Type <> delete ptr***
  - ***ptr = new Type (initial_value) <> delete ptr***
  - ***ptr = new Type[size1] ... [sizeN] <> delete []ptr***
- **malloc / free (C - lower level manipulation of memory)**
  - ***ptr = (type*) malloc (memory_size)***
  - ***free (ptr)***

Example (e11.cc)

```
#include <iostream>

int main ()
{
    while (1)
        new int[100];
    return 0;
}
```

Array Allocation

Example:

```
// A single object with value 1024
int *obj = new int (1024);
// An array
int *vec = new int [1024];
int (*mat)[1024] = new int [4] [1024];
delete[] mat;

const int *pci = new const int (1024);
```

Example (e12.cc)

```
#include <iostream>

int*[] alloc_mat (int dim1, int dim2) {
    int (*mat)[dim1] = new [dim1][dim2];
    return mat;
}

int main ()
{
    int (*mat1)[10] = alloc_mat (10, 10);
    int (*mat2)[20] = alloc_mat(20,20);
    delete []mat1;
    delete []mat2;
}
```

Type Conversion

- Implicit
  - int a;
  - a = 3.4;

- Required by Arithmetic Operations
  - 4.5 * 3;

- Explicit
  - (int) 3.4
  - static_cast<int> (3.4);
  - dynamic_cast (runtime casting)

Homework

- **Due date: January 17, 8:00pm**

- Problem 0. Problem 7 / Page 81 (Navigating C++) -- nothing to hand in

- Problem 1. Write two functions (ten2tes and tes2ten) for translating large numbers represented as strings from base 10 to base 2, 8 or 16 and, respectively from 2, 8 or 16 to 10.

  ```
  char* ten2tes (char* bten, int base);
  char* tes2ten (char* tes);
  ```

  Example:

  10 (10) = xA (16) = 012 (8) = b1010 (2)
  20 (10) = x14 (16) = 024 (8) = b10100 (2)

- Problem 3. Problem 1 / Page 164 (Navigating C++)