宣州謝朓樓餞別校書叔雲

棄我去者，昨日之日不可留；
亂我心者，今日之日多煩憂。
長風萬里送秋雁，
對此可以酣高樓。
蓬萊文章建安骨，
中間小謝又清發。
俱懷逸興壯思飛，
欲上青天覽明月。
抽刀斷水水更流，
舉杯銷愁愁更愁。
人生在世不稱意，
明朝散髮弄扁舟。

Pointers as Arguments to a Function

- Ex5_03.cpp on P.262

- ```
  int incr10(int *num);
  // Function Prototype
  ```

- ```
  int* pnum = &num; // Pointer to num
  ```

- ```
  *num += 10;
  ```
- ```
  return *num;
  ```
- ```
  // de-reference the pointer to get the return value
  ```
  ```

Passing Arrays

- The pointer to the beginning of the array is passed by value to the function.
- Ex5_04.cpp on P.263
- `double average(double array[], int count);`
- `average(values, (sizeof values)/(sizeof values[0]));`
  - sizeof values = ?
  - sizeof values[0] =?
- card_shuffle.cpp

What Is a Reference?

- A reference is an alias for another variable (Chapter 4, P.207).
  - `long number = 0;`
  - `long& rnumber = number;`
  - `rnumber += 10;`
  - `cout << number;`

- Difference between a pointer and a reference:
  - A pointer needs to be de-referenced
  - A reference is an alias. There is no need for de-referencing.

Pass-by-reference

- Remember that a reference is merely an alias.
- Ex5_07.cpp on P.268
- The output shows that the function `incr10()` is directly modifying the variable passed.

Static Variables in a Function (P.283)

- With only “automatic” variables within a function, you can’t count how many times a function is called.
- Within a function, you can declare a variable as **static** so that its value persists from one call to the next.
  - Initialization of a static variable within a function only occurs the first time that the function is called.
- Ex5_14.cpp on P.284

Recursive Function Calls (P.285)

- Recursive function - A function calls itself
  - Either directly or indirectly
  - fun1 $\rightarrow$ fun2 $\rightarrow$ fun1
- Fibonacci sequence:
  - F(n) = F(n-1) + F(n-2)
  - F(0) = 0, F(1) = 1
- Factorial
  - N! = N*(N-1)*(N-2)*…*3*2*1
  - F(n) = n * F(n-1)
  - F(0) = 1
- Be sure to specify the boundary condition to stop the recursive call!

Fibonacci Sequence

- Implemented as a recursive function

```
// ch5_fibonacci.cpp
#include <iostream>
using std::cout;
using std::endl;

int f(int n)
{
        if (n == 0)
                return 0;
        if (n == 1)
                return 1;
        return f(n-1) + f(n-2); // recursive call
}

int main()
{
        for (int i=0; i<20; i++)
                cout << f(i) << ' ';

        cout << endl;
        return 0;
}
```

Compare with the one implemented as an array

Factorial

```
// factorial.cpp
#include <iostream>
using std::cout;
using std::endl;

int f(int n)
{
    if (n <= 0)
        return 1;
    else
        return f(n-1)*n;
}

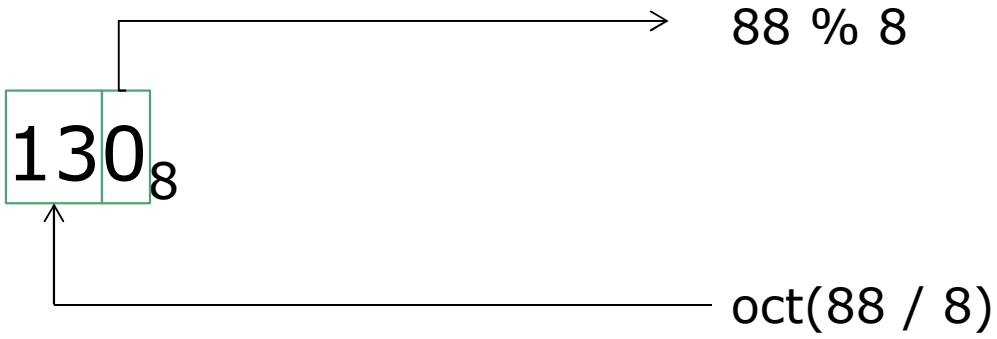
int main()
{
    for (int i=1; i<=10; i++)
        cout << i << "! = ";
        cout << f(i) << endl;
    return 0;
}
```

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

Decimal → Octal

- $23_{10} \rightarrow 27_8$

- $88_{10} \rightarrow 130_8$



- $128_{10} \rightarrow 200_8$

ch5_dec2oct.cpp

```
#include <iostream>
using std::cout;
using std::endl;

void oct(int n)
{
    if (n >= 8)
        oct( n / 8);   // recursive call
    cout << n % 8;
}

int main()
{
    oct(23); cout << endl;
    oct(88); cout << endl;
    oct(128); cout << endl;
    return 0;
}
```

```
27
130
200
```

Exercise & Homework

- P.292
  - Exercise 1, 2
- HW: Convert decimal to hexadecimal by a recursive function.

C Time Library <ctime>

- Types
  - clock_t - Clock type
  - size_t - Unsigned integral type
  - time_t - Time type
  - struct tm - Time structure (See Chapter 7)
- Time manipulation
  - clock - Ticks since the program was launched
  - time - Get current time
  - mktime - Convert tm structure to time_t
- Macro
  - CLOCKS_PER_SEC - Clock ticks per second
- Conversion
  - asctime - Convert tm structure to string
  - ctime - Convert time_t value to string
  - gmtime - Convert time_t to tm as UTC time
  - localtime - Convert time_t to tm as local time
  - strftime - Format time as string

time()

```
#include <ctime>
time_t time ( time_t * timer );
```

```
/* time example */
#include <iostream>
#include <ctime>

int main ()
{
  time_t seconds;

  seconds = time(NULL);
  std::cout << seconds/3600
    << " hours since"
    << " January 1, 1970\n";

  return 0;
}
```

```
/* time example */
#include <iostream>
#include <ctime>

int main ()
{
  time_t seconds;

  time(&seconds);
  std::cout << seconds/3600
    << " hours since"
    << " January 1, 1970\n";

  return 0;
}
```

ctime()

```
#include <ctime>
time_t time ( time_t * timer );
```

- Converts the time_t object (seconds since 1970.1.1) to a C string containing a human-readable version of the corresponding local time and date.

```
/* ctime example */
#include <iostream>
#include <ctime>

int main ()
{
  time_t rawtime;

  time ( &rawtime );
  std::cout << "The current local time is: "
   << ctime (&rawtime)  << std::endl;

  return 0;
}
```

output

The current local time is:
Sat Nov 10 11:57:33 2012