

Detour back to shell – scripts

In preparation for this week's lab

Not covered in Reader (#1 just mentions)

Later: More OO design classes.

Bourne shell programs

- Are text files with `sh` commands – e.g., `myScript`
 - To execute, can do `sh myScript`
 - The program runs in a new shell – called a child shell
 - Or `chmod u+x myScript` – then just `./myScript`
 - Requires that `sh` is the default shell (usually `bash` okay too)
- `#` – normally identifies a comment
 - Special case if line 1 – `#!/bin/sh` – identifies shell
 - Means use `sh` as child shell for this script – works in all shells
- Can access command line arguments: `$1` to `$#`
 - e.g., `cp $1 $2 # copies first to second (if files)`
 - e.g., `echo $# # prints number of arguments`

sh variables and assignment

- `name="Jack Sprat" # note no spaces`
- `echo "The name is $name" # need '$'`
- `workdir=`pwd` # use `...` to assign result of ...`
 - Similarly, `echo "date and time is `date`"`
- Can read from standard input and calculate too
 - `echo "enter value"`
 - `read val`
 - `doubleval=`expr $val + $val``
 - Or just: `echo "doubled: `expr $val + $val`"`

sh control structures, and FYIs

- An `if-then-elif-else-fi` statement
 - Expression is a test: `test $# -gt 0`
 - Or simpler: `[$# -gt 0] # spaces mandatory`
 - Can test files too: `-d, -f, -e, -r, -w, -x, ...`
- A `while-do-done` statement: same expressions
- A `for-do-done` statement: `for variable in list`
 - List is command line arguments if not specified
- FYI: can program *any* shell, but different syntax
 - Also "scripting languages" (e.g., Perl, Python, ...)
- Examples at `~mikec/cs32/demos/scripts/`

First Exam
Wednesday, April 17

Starting Savitch Chapter 10

Classes

- A class is a data type whose variables are objects
 - Some pre-defined classes in C++ include `int`, `char`, `ifstream`
 - Of course, you can define your own classes too
- A class definition says two basic things
 - The kinds of values an object can hold
 - A description of the member functions

Example: class DayOfYear

- Decide on the values to represent
- This example's values are dates such as July 4 using an integer for the number of the month
 - Member variable month is int (Jan = 1, Feb = 2, etc.)
 - Member variable day is int
- Decide on the member functions needed
- Just one member function named output in the first version of this class

Simplest version of DayOfYear

```
class DayOfYear {  
public:  
    void output();  
    int month;  
    int day;  
};  
void DayOfYear::output() {  
    cout << "month = " << month  
        << ", day = " << day << endl;  
}
```

- Like a struct with an added method
 - All parts public
 - Clients access month, day directly

Notes about '::' and '.'

- '::' used with classes to identify a member
`void DayOfYear::output() { ... }`
 - Also used with namespaces – identifies scope
 - Called scope resolution operator
- '.' used with variables to identify object
`DayOfYear birthday;
birthday.output();`
 - Object reference is passed to the method as an implicit parameter