CSCI251/CSCI851 Autumn-2020
Advanced Programming (**LT3**)

Lecture Tutorial 3

Outline

- From the lab:
- Assignment.
- Proceeding procedurally.

From the lab:

- Debugging:
  - File include structure.
  - Understand what the program does … it's not enough for it to compile.
    - Parts need to match…
  - Pass by reference or pass by value.
- Pointer arithmetic for array access …

```
int numbers[3];
int* ptr= numbers;
cin >> *(ptr+i);
cout << *(ptr+i);
```

```
char A[]="Elephant";
string B="Elephant";
int C[8]={5};
int *D=new int[8];

cout << sizeof(A) << endl;      9
cout << sizeof(B) << endl;      32
cout << sizeof(C) << endl;      32
cout << sizeof(D) << endl;      8
```

A: 8 characters + 1 terminating null character \0

B: Change the string, same size. Dynamic behind the scenes.

C: 8 int variables, 4 bytes each.

D: Change the number, same size. It's the size of the pointer, not the memory set aside.

Selecting clearly…

```
IF ((time < 1 day) OR ((time < 1 week) AND (workers <
 5)) OR ((time < 2 hours) AND (workers < 2))):
  IF ((time < 2 hours) AND (workers < 2)):
     IF (the work needs power tools):
        prohibit the work
     ELSE:
        allow the work
  ELSE:
     IF (the work needs power tools):
        prohibit the work
     ELSE:
        allow the work in 4-hour shifts
ELSE:
  IF (the work needs power tools):
        prohibit work
  ELSE:
        consult with steward
```

From:
Best Practices of Spell Design
Jeremy Kubica

```
is_small_job = (time < 2 hours) AND (workers < 2)
is_medium_job = ((time < 1 week) AND (workers < 5))
  OR (time < 1 day)
IF (the work does not need power tools):
   IF (is_small_job):
      allow the work
   ELSE IF (is_medium_job):
      allow the work in 4-hour shifts
   ELSE:
      consult with steward
ELSE:
   prohibit work
```

From:
Best Practices of Spell Design
Jeremy Kubica

- Easier to read, easier to test…

Function pointer calling example

- This is from the textbook: pages 247-249.

```
bool lengthCompare(const string &, const string &);
```

Function type: `bool(const string&, const string&)`

```
bool (*pf)(const string &, constr string &);
```

Pointer to a function of type: `bool(const string&, const string&)`

```
pf = lengthCompare;
pf = &lengthCompare;
```

- There are a couple of equivalent ways of setting parameters as a pointer to a function :

```
void useBigger(const string &s1, const string &s2, bool
pfLocal(const string &, const string &));

void useBigger(const string &s1, const string &s2, bool
(*pfLocal)(const string &, const string &));
```

- Calling is something like …

```
useBigger(s1, s2, lengthCompare);
```

- The book notes that this is likely a good place to use a typedef, such as:

```
typedef bool Func(const string&, const string&);
void useBigger(const string&, constr string&, Func);
```

Proceeding procedurally

- We had our scenario last week … reading in a collection of numbers, sorting them, and displaying them.
- Before we get too carried out we should consider the size of the overall code base.
- It's not large yet but it's good practice to break our program into sections across multiple files.

The task at hand …

- Read in a collection of numbers and sort them in increasing order before displaying them.
- The parts based on that:
  - Reading the data → a collection of numbers.

    vector<int> v , push_back.
  - Sorting the data. sort(v.begin(), v.end())
  - Displaying the data. for (auto e : v)
- We can write procedures/functions for each of these.

Assignment One

```
A:Head:1:2:15.
B:Torso:0:6:5.
C:Leg:0:4:6.
D:Arm:0:4:8.
E:Tail:0:6:2.
```

Parts.txt

Data files.
Specification.
Marking guide.

```
Carly:Cat:ABCCCCE.
Dodgy Dan:Dog:BCACECC.
Ernie:Ettin:AABCCDD.
Sally:Snake:AEEEEEE.
```

Customers.txt

```
Reliable Rover:70:1.
Sloppy Simon:20:4.
Technical Tom:90:3.
```

Builders.txt