Foundation of Computer Science: Exam II

Dr Kafi Rahman, PhD
Assistant Professor @Computer Science
Truman State University

Agenda

- Maximum and Minimum
  - Brute force
  - Sequential processing

Study in Design: Max of Three

- Now that we have decision structures, we can solve more complicated programming problems.
- Suppose we need an algorithm to find the largest of three numbers.

Study in Design: Max of Three

```
int main()
{
    int x1, x2, x3, max;
    cout<<"Enter three integer values: ";
    cin>>x1>>x2>>x3;
    # missing code sets max to the largest value

    cout<<"The largest value is"<< max;
}
```

Strategy: Compare Each to All

- This looks like a three-way decision, where we need to execute one of the following:

```
max = x1;
max = x2;
max = x3;
```

- All we need to do now is preface each one of these with the right condition!

Strategy: Compare Each to All

- Let's look at the case where x1 is the largest.
- Only way x1 is the largest value, if
  - x1 is larger than x2 and at the same time it x1 must be larger than x3
  -
```
if (x1 >= x2 && x1 >= x3)
    max = x1;
```

Strategy: Compare Each to All

- We can separate these conditions with and

```
if (x1 >= x2 && x1 >= x3)
    max = x1;
else if (x2 >= x1 && x2 >= x3)
    max = x2;
else
    max = x3;
```

- We're comparing each possible value against all the others to determine which one is the largest.

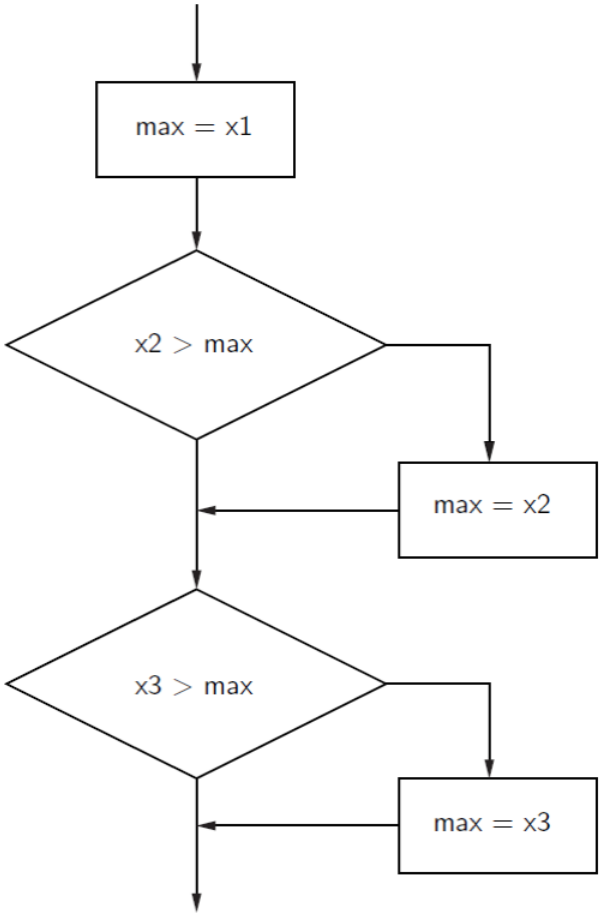
Strategy: Compare Each to All

- What would happen if we were trying to find the max of five values?
- We would need four Boolean expressions, each consisting of four conditions anded together.

Strategy: Sequential Processing

- You could probably look at three numbers and just know which is the largest. But what if you were given a list of a hundred numbers?
- One strategy is to scan through the list looking for a big number. When one is found, mark it, and continue looking. If you find a larger value, mark it, erase the previous mark, and continue looking.

Strategy: Sequential Processing



Strategy: Sequential Processing

- This idea can easily be translated into C++.

```
max = x1;
if (x2 > max)
    max = x2;
if (x3 > max)
    max = x3;
```

Strategy: Sequential Processing

- This process is repetitive and can be readily used in a loop.
- We prompt the user for a number, we compare it to our current max, if it is larger, we update the max value, repeat.

Strategy: Sequential Processing

```
/* Finds the maximum in a
series of numbers */
int main(){
  int n=0, number=0;
  cout<<"How many numbers: ";
  cin>>n;

  cout<<"Enter a number: ";
  cin>>number;
  /*Set max to be the first
  value */
  int max = number;
```

```
/* Now compare the rest of the
n-1 successive values */
  for (int i=1;i<n; i++)
  { cout<<"Enter a number: ";
    cin>>number;
    if (number > max)
       max = number;
  }
  cout<<"The maximum: "<<max;
}
```

Munch strategy

Function Prototype

- Write first the prototype, then the actual definition of a C++ function called fahrenheit_to_celsius that takes a Fahrenheit temperature (double) as input and returns the equivalent Celsius temperature (double). Be sure that your prototype is documented the way our style guide prefers. The formula for conversion (as defined by a math teacher) is:

- C = 5/9(F − 32)

- Which would be accurate prototype:
  - int fahrenheit_to_celsius(double c);
  - void fahrenheit_to_celsius(double c);
  - double fahrenheit_to_celsius(double);
  - double fahrenheit_to_celsius(double c);

Function Parameter as Action

```
/* This function will calculate the result depending on the value of the action variable */

void math_func(char action, double x, double y, double & res)
{
if(action == '+')
    res = a+b;
else if(action == '-')
    res = a-b;
else if(action == '*')
    res = a * b;
else if(action == '/')
    res = a/b;
else
    res = 0f;
}

// now calling the function
int main()
{
    double result=0;
    math_func('+', 10.5 , 11.5, result);
    cout<< result;
}
```

Group Discussion: Loop

- Given the following while loop:

```
const unsigned MAX_COUNT = 10;
unsigned count = 0;
unsigned total = 0;
while (count < MAX_COUNT)
{
    total += count;
    count++;
}
cout << total << endl;
```

- Show a for loop that does the equivalent calculation:
- Now show a do while loop that also does the equivalent calculation:

Group Discussion: Passing by Reference

- Some programming languages provide a division_result function that calculates both the integer quotient and the remainder when given two integers to divide.
- In C++ we can write a version of this function with four parameters.
  - The first two parameters are pass-by-value and can be called dividend and divisor.
  - The last two parameters are pass-by-reference and can be called quotient and remainder.
  - The division_result function does not return anything (void).

- Write the prototype and definition for a well-written C++ function that does this.
  - As an example, if dividend is 13 and divisor is 5, then the value assigned to quotient should be 2 and the value assigned to remainder should be 3.

Questions?