Functions – Part 1

CS1A

- Why we need them?
  - Divide & Conquer
- What are they?
- Function Basics
  - Create, Define, & Use
  - Passing values into functions

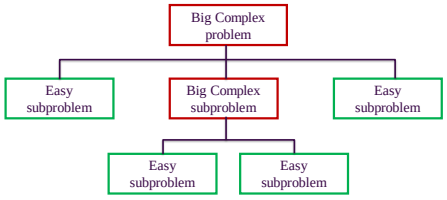
Divide and Conquer

- We break big complex problems into smaller comprehensible problems
  - This is how we build big programs
- WHY?
  - Smaller problems are easier to manage and code
  - Smaller code segments are easier to code & test (debug)
  - If we can break it into small independent tasks then we can have many programmers each working on their own task

Divide & Conquer

Break up big complex problems into smaller manageable problems

- Keep refining until all problems are easily understood tasks



Divide & Conquer Example

Calculate a grade point average.

Problem statement

- We need an interactive program (user will input data) that calculates a students' GPA given a series of letter grades.

Input/Output description

- Input → Letter grades
- Output → Grade point average

Algorithm development (set of steps, decomposition outline)

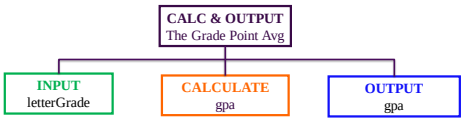
1. Read the letter grade
2. Convert it to a numerical grade
3. Calculate the grade point average
4. Output the grade point average

Each of these can be thought of as separate tasks!

Divide & Conquer Example (2)

A HIPO chart is one way to divide and conquer

- We want to break the problem down into logical segments
- Less complex tasks
- Smaller tasks are easier to code, test/debug, reuse, modify…



How do we do this in C++?

- Each separate task is called a component, module, subprogram
  - Each of these are logical groupings of code
- In C++ we call these functions
- Each function has its own task
  - It is a small program that we can use over and over again by calling it
- We have been discussing functions all along
  - e.g. toupper, ceil, fabs

These are pre-defined. We want to create our own functions.

- Functions should accomplish one task
  - This way we can reuse it
  - Make it general

Example

- Programs often have different tasks
  - Input, validate input, get the grade point value of a letter grade, calculate, gpa, etc..
- We can split these up into different functions
  - Each function processes its own task
  - We call upon it as necessary

Coding is similar – just have to separate them out into new functions and call them!

The Main function

int main ()

- Up until now we have used the function main
- All C++ programs must have this function
- The operating system runs the program until main returns 0

Function Basics

- All Functions must have a return datatype
  - This represents the datatype of the value it is returning
  - For main we use int
  - Some functions return some value
  - Others do not ← They still have to have a datatype

Example

- we can have a function that calculates the square root of a number. ← will return a float
- We can have a function that outputs a header (such as your name, date, class, etc.) ← doesn't return anything but needs a datatype none-the-less

- Once we create these functions we can call them over and over again and don't have to rewrite them every time we want to use them

Function Process

For every function we need to..

1 - Declare the function

2 - Define the function

3 - Use the function

Using a Function

```
theSquareRoot = sqrt(num1);
```

The parens are required

Variable

Function Name

Argument

Argument → sometimes we need to send data to the function, we use arguments to do this.

Argument_List → a list of arguments that follows the function call

Variable → if our function returns a value we need a place to store it.

NOTE: This function returns a value so we need to either assign that value to a variable or cout it.

e.g. cout << sqrt(num1);

How do we define a function?

Syntax

```
returnType FunctionName ( type parameterName...)
{
    statements;
    return value;
}
```

- Somewhere in statements you need to have a return statement.
- The function terminates when the return statement is executed
- The return statement returns the value of the code.

Example

```
int AddTwoInts(int num1, int num2)
{
    return num1 + num2;
}
```

NOTE:
We have to let the compiler know what the types are of our arguments (incoming data)

Declaring a function

We can declare them

Somewhere in our main source file

(main.cpp for example) → before int main () We have to tell the compiler about the function

We do this using a function prototype

Syntax

returnType functionName (type parameterName...);

Example

```
int AddTwoInts(int num1, int num2);
```

Note: this is exactly like the first line of our function definition except the ;

Why use a Prototype?

We need the prototype to be first because the compiler reads from top to bottom

1. We can put it in the same file as our int main()
   → Before the int main() ← For now we will do this
2. We can put it in a separate header file
   - We will cover this later
3. We can define the function before the int main() { } (rather then declaring the prototype) and then we won't need the prototype
   - This will require that our functions appear in a particular order making our code hard to maintain

NEVER use this option 1

Calling our function

Now we just need to call our function

```
#include <iostream>
using namespace std;
 int main ()
{
      int n1
      int n2;
      int sum;

      cout << "Enter the first value to be summed: ";
      cin   >> n1;

      cout << "Enter the second: ";
      cin   >> n2;

      sum = AddTwoInts(n1, n2);

      cout  << endl;
      cout  << "The sum is: " << sum << endl;

      return 0;
}
```

Prototype goes here

NOTE: we could have put AddTwoInts here

Putting it all together

```
#include <iostream>
using namespace std;

int AddTwoInts(int num1, int num2);

int main ()
{
      int n1, n2, sum;

      cout << "Enter the first value: ";
      cin  >> n1;

      cout << "Enter the second: ";
      cin  >> n2;

      sum = AddTwoInts(n1, n2);

      cout << "\nThe sum is: " << sum << endl;

      return 0;
}

int AddTwoInts(int num1, int num2)
{
    return  num1 + num2;
}
```

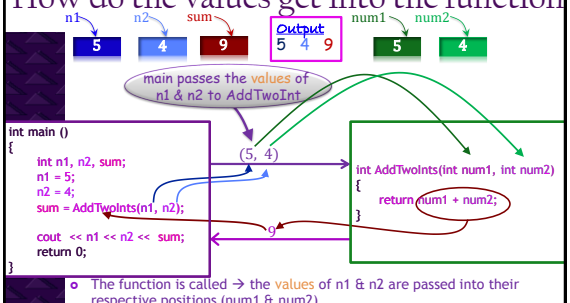
Prototype

Calling Function

function call

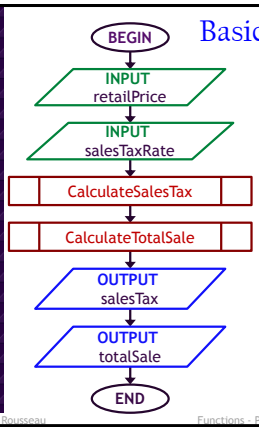
Function Definition

How do the values get into the function



- The function is called → the values of n1 & n2 are passed into their respective positions (num1 & num2)
- In this case the n1 is the first argument will be sent to the first parameter in the function → num1
- If we had put n2 first then the value in n2 would be sent to num1
- The value of num1 + num2 is returned → the function call gets the return value

Basic Functions Exercis



Let's make functions for the calculations

What does CalculateSalesTax need as input?
What will it output?

Write the function and write main up to and including the call to Calculate Sales Tax

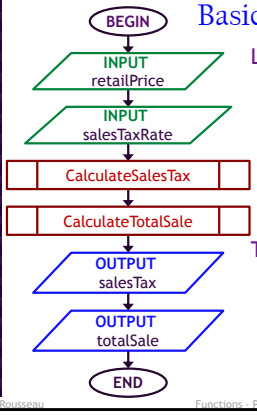
Calculate Sales Tax Function

Parameter or Local Variable

How do we know what needs to be passed in and what should be declared in the function?

- Function should be accomplish one task.
- If the value is defined outside of the function and used within the function it needs to be passed into the function.
- If the value just needs to be initialized or if it is defined within the function before it is used it should be declared in the function.

Basic Functions Exercis



Let's make functions for CalculateTotalSale

What does CalculateTotalSale need as input?
What will it output?

Then complete int main.

Calculate Total Sale Function