Intro to Programming – Part 1

CS1A

- Origins of C++
- Declaring Identifiers
- Basic Structure of a C++ Program
- Going from Flowcharts / Pseudocode to C++ code

Origins of C++

- Ken Thompson
  - Developed Unix
  - Used Assembly and "B" to program Unix
  - B → Derived from BCPL
- Denis Ritchie
  - 1970s, Developed C to program Unix
  - C was derived from B
- C was not as strictly typed as other languages
  - Allowed it to be more flexible
  - Easier to read and write than Assembly
  - But… more error prone → lacked automated checks

Origins of C++ (2)

- Bjarne Stroustrup
  - 1980s, Bell Labs developed C++
  - Based on C
    - If you can understand C you can understand C++
      - Vise versa is not necessarily true
    - C++ allows for object oriented programming (OOP)
      - More modern style of programming
    - Also, has stronger type checking and standards
      - Easier to code
      - Easier to reuse
      - Easier to modify
      - Easier to debug

Definitions

- Computer Programming
  - The process of implementing algorithms using a language the computer can understand
- Computer Program
  - An implementation of an algorithm
  - A step by step set of instructions performed by the computer
- High-level languages
  - "English" or "people friendly" languages
- Compiler
  - Interprets High level languages into machine language

What is a Programming Language

- Programming Language Consists of

  …a set of special words, symbols and rules used to construct a program

  - Syntax – rules that dictate how valid instructions are written.
  - Semantics – rules that dictate the meaning attached to the instructions

Storing data in Memory

- Once you give a place in memory a specific name you can refer to that location by using that name (the identifier)

  →symbolic referencing

Identifiers

- a descriptive name that maps to a location in the computers memory
  - Variables are one type of identifier
- We have identifiers so we can
  - Retrieve data
  - Reuse data
  - Modify data

Identifiers

2 types of Identifiers

- Variables - contains data values that may change during program execution
  - Can be retrieved (used)
  - Modified
- Constants - contains data values that can't be changed during program execution
  - The value must be declared
  - Can be retrieved
  - Can NOT be modified

Identifiers Naming Rules

- Required rules →
  - Identifiers can only have
    - Letters
    - Numbers
    - Underscore (_)
  - must begin with a letter
  - Can't have spaces
  - Can't have special characters
- Remember they are case-sensitive

payRate ≠ payrate ≠ Payrate ≠ PayRate

Can't use KEYWORDS

KEYWORD - a word that has some sort of predefined meaning in the context of a programming language

Identifiers – Naming Conventions

Stylistic rules → what we care about

ALWAYS Use meaningful names

Variables

- For 1 word → Keep it lowercase
- For 2 or more words
  - Begin with a lower case letter and only the first letter of each successive word will be capitalized

Constants

- All words in caps
- Use underscore to separate words
- eg TAX_RATE, PROGRAMMER

How to Declare Identifiers

- We need to tell the compiler what identifiers to use
- The compiler needs to know
  - The data type (or type)
    - How the contents of memory need to be viewed
    - In other words, what kind of information will be stored
    - C++ is strongly typed
      - You have to be specific about what you are storing
  - How much memory is needed to store your data

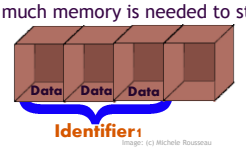


Image: (c) Michele Rousseau

Syntax	Description	Size	Data Values	Examples
int	integer	4 bytes	Pos. or neg. integers (whole numbers) (9 digits 2^{32})	3, 4, 235, 1215232, -23, -432
char	character	1 byte	Characters enclosed in quotes	'a', 'z','d','f'
float	floating point number	4 bytes	Pos. or neg. decimal numbers - including fractional part (up to 7 digits)	32.234, -23.32, 0.0, 1.25 123.2353
long	long integer	4 bytes (8 on some systems)	Same as int	Same as int,
double	double precision float	8 bytes	Floats up to 15 digits	Same as float, but larger #s
bool	Boolean	1 byte	One of 2 values: True or false	True False

Declaring an Identifier

Variables

Syntax

type variableName;

Examples:

```
int   sum;
float average;
char response;
```

The semi-colon tells the compiler that you are finished with the statement

Constants

Syntax

const type CONSTANT_NAME = value ;

Examples:

```
const int   DAYS_IN_WEEK   = 7;
const float SALES_TAX_RATE = 0.075;
```

Strings are special

If you have more than 1 character we need to store we use c-strings

C-Strings

- An array of characters
- The last character is called a null terminator (\0)
  - Tells the compiler where the end of the string is
- We need to specify how many characters we want

Syntax
char *variableName*[*size*];

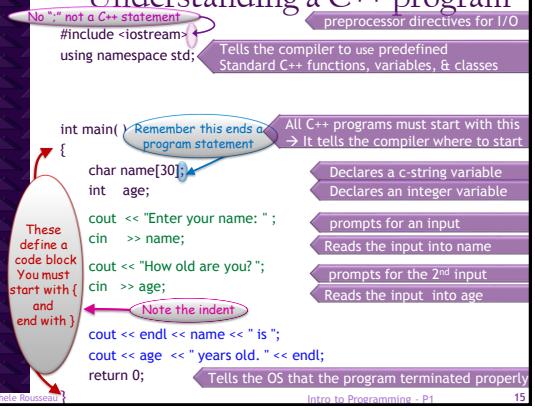
C-String Examples

char lastName[15]; → can hold 15 characters
(14 + the null terminator)

char lastName[3]; → can hold 3 characters
(2 + the null terminator)

Why is this a bad idea?

Understanding a C++ program



Breaking it down

#include <iostream>

- Pre-processor Directive
  - Tells the pre-processor that we want to use i/o functions so we #include them in our code
  - We need iostream to use cin / cout statements

using namespace std;

- Tells the compiler that we want to use all the standard C++ functions
- Functions are small code segments that we use to build our program

Don't worry too much about the details here yet
-- just be sure to include them

Main

```
int main ()
{
   body of function (i.e. program statements)
   return 0;
}
```

- Program execution begins with this function
- All C++ programs must have this function
- MUST BE an int
- Must start with { and end with }
- Must have return 0; as last statement
  - This returns the value 0 to the system so it knows the program completed properly

Declaring Identifiers

```
char name[30];
int  age;
```

End of C++ statement
Datatype (or Type)
Variable Name

- These are variable declarations
- Remember we must reserve memory locations to store data
- The compiler needs to know the
  - Type of data (or datatype)
  - and how much to store

Cin / Cout

```
cout << "Enter your name: ";
cin  >> name;
```

- cout and cin are how we communicate with the user
- cout
  - Inserts data to the output stream
  - Uses the insertion operator ("<<")
  - Can output a string literal using "" or variables

```
cout << endl << name << " is ";
```

Insertion operator; endl → inserts a newline; Variable Name; String Literal "\n" inserts a Carriage return Must be in ""

Note:
We can keep output much as we want in one line
→don't make it so long you can't read it when typing it in

CIN

```
cin >> name;
```

End of C++ statement; Extraction operator; MUST BE A VARIABLE

- cin extracts information from the input buffer using the extraction operator ">>"
- What you type on the keyboard gets put into the input buffer
- In this example the contents of the input buffer are put into the variable name
- YOU CAN ONLY HAVE VARIABLES on the RIGHT OF A CIN STATEMENT
  - → because the data must be stored somewhere

Note:
Generally speaking we want to pair cin with a cout so that the user knows the program is waiting for an input

Flowchart/Pseudocode to Code

FLOWCHART	⇔ PSEUDOCODE	⇔ CODE
BEGIN	⇔ BEGIN	⇔ `int main()` `{`
INPUT someVar	⇔ INPUT someVar	⇔ `cout << "Enter some value: ";` `cin >> someVar;`
OUTPUT someVar	⇔ OUTPUT someVar	⇔ `cout << "My value is: " << someVar;`
END	⇔ END	⇔ `return 0;` `}`
totVal = totVal + 3	⇔ CALC …	⇔ `totVal = totVal + 3;`

VARIABLE LIST
retailPrice
salesTaxRate
salesTax
totalSale

```
#include <iostream>
using namspace std;
int main()
{
    float retailPrice;
    float salesTaxRate;
    float salesTax;
    float totalSale;
    cout << "Enter the retail price: ";
    cin  >> retailPrice;
    cout << "Enter the salesTaxRate: ";
    cin  >> salesTaxRate;
    salesTax = retailPrice * salesTaxRate;
    totalSale = retailPrice + salesTax;
    cout << "Sale Tax: "  << salesTax  << endl;
    cout << "TotalSale: " << totalSale << endl;
    return 0;
}
```

BEGIN; INPUT retailPrice; INPUT salesTaxRate; salesTax = retailPrice * salesTaxRate; totalSale = retailPrice + salesTax; OUTPUT salesTax; OUTPUT totalSale; END

Using Basic CIN / COUT commands

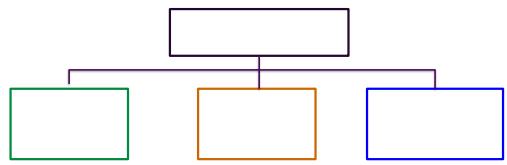
- CIN / COUT Example

Write a program that will take in two integers input from a user. It will sum those two integers. Output the integers and the sum as described below.
Draw a HIPO chart, pseudocode, then a flowchart.

SAMPLE INPUT / OUTPUT
Enter the first integer: 32
Enter the second integer: 41

32 + 41 = 73

HIPO Chart



INPUTS OUTPUTS PROCESSING

Psuedocode

BEGIN

END

Sum Flowchart

VARIABLE LIST:

INPUTS

OUTPUTS

PROCESSING

Comments

Comments are text in the source code that the compiler ignores

How to add comments

- // ← for a few lines or after a line of code
  - You can select a group of code and ctrl - / to comment out several lines at a time
  - If you ctrl - / on a comment it will uncomment the line
- Block comments

```
/*
    <anything between these will be commented>
*/
```

  - USE BLOCK COMMENTS FOR YOUR OUTPUT
    - Cut and paste output from the console window into the editing window so it will print out

Commenting your code

For all programs in this class

- Before int main()
  - Use comments to describe your program (more on this later)
- Data Table
  - The declaration section must contain a data table
  - The data table states
    - how its value is defined & used
    - describes what will be stored in that variable
- Other comments should be used throughout your code
  - Describe what each section is doing
    - (think in terms of input, processing, & output)
  - Complicated parts of the code → be descriptive!

Pair Programming

- One component of the XP (eXtreme Programming) software process model
- TWO programmers - ONE computer
  - Driver – types the code
    - Comes up with the algorithms and etc…
  - Navigator (or Observer)
    - Looks for ways to improve the code
  - Roles are switched frequently
- In this class…
  - Switch roles every 10 - 15 minutes – or after each function
  - Must be co-located – can't be done remotely
  - MUST BE COMPLETED TOGETHER IN LAB!
  - 3 scenarios
    - →BOTH students are responsible for understanding the code
  - Must pick a different partner for each lab for credit