Oregon State University

COLLEGE OF ENGINEERING | School of Electrical Engineering and Computer Science

CS 161

Introduction to CS I

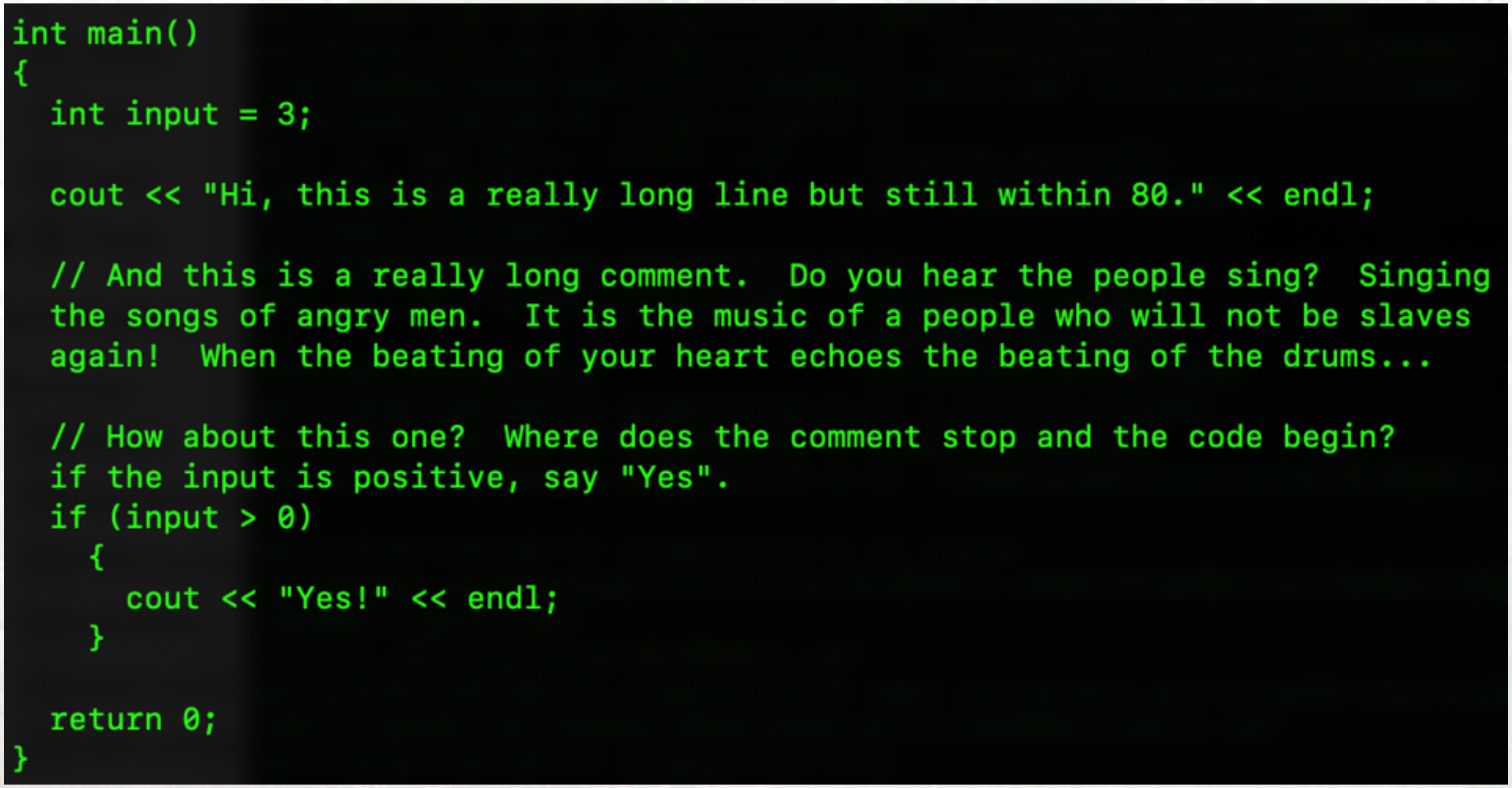
Lecture 9

- How do functions make programming easier?
- Introduce Assignment 3

Assignment submission note

- If you have technical trouble submitting,
  email me your files before the deadline so you don't miss it

Also, let's review the 80-char limit



```
int main()
{
  int input = 3;

  cout << "Hi, this is a really long line but still within 80." << endl;

  // And this is a really long comment.  Do you hear the people sing?  Singing
  the songs of angry men.  It is the music of a people who will not be slaves
  again!  When the beating of your heart echoes the beating of the drums...

  // How about this one?  Where does the comment stop and the code begin?
  if the input is positive, say "Yes".
  if (input > 0)
    {
      cout << "Yes!" << endl;
    }

  return 0;
}
```

Integers and characters (Lab 3)

- A character is stored as an 8-bit unsigned value
  - which can be interpreted as an integer (ASCII chart)
    - 'a' = 97 'A' = 65
    - 'b' = 98 'B' = 66
    - 'c' = 99 'C' = 67
    - …
    - 'z' = 122 'Z' = 90
  - `char c = 'c';`
  - `cout << (int)c << endl;`
  - `cout << (char)122 << endl;`

Bonus for loop variations

```
int end_loop, step; /* declare two variables */

cin >> end_loop;     /* let user control loop */
cin >> step;

for (int x = 0; x < end_loop; x += step)
{
  cout << x << endl;
}
```

Physical objects as functions?

- Key(lock) => open, close, rejected
- Lock(key) => success, failure
- Can_opener(can) => open can
- Tea_kettle(water) => hot water

Functions allow us to…

- Divide and conquer
  - Break problem/task into subtasks: **decomposition**
- Make programs easier to design
- Make code easier to understand
  - Abstract away from details
- Reduce replicated (repeated) code
  - Why does this matter?

Worksheet: Function abstraction

- One possible solution:

```
int get_sum(int n_steps, char c)
{
  int sum = 0;
  for (int i = 0; i < n_steps; i++)
  {
    cout << c;
    sum += i;
  }
  return sum;
}
```

Function calls

```
int sum = get_sum(10, '*');
cout << ": " << sum << endl;
```

```
int sum2 = get_sum(5, '=');
cout << ": " << sum2 << endl;
```

```
int sum2 = get_sum(3, '+');
cout << ": " << sum2 << endl;
```

Functions: multiple parameters

Function definition

```
float calc_BMI(float height, float weight)
{
  return weight / pow(height, 2);
}
```

- But only one return value
- Functions can call other functions

Functions: default values

Function definition

```
float calc_BMI(float height, float weight = 54)
{
  return weight / pow(height, 2);
}
```

Function call

```
int main()
{
  float bmi_1 = calc_bmi(1.9, 54);
  float bmi_2 = calc_bmi(1.9);
  return 0;
}
```

Functions: default values

Function definition

```
float calc_BMI(float height, float weight = 54)
{
  return weight / pow(height, 2);
}
```

- If a default value is provided for one parameter, default values must also appear for all following parameters
  - Why?

Functions: no parameters

Function definition

```
int welcome_user()
{
  cout << "Welcome, dear user!" << endl;
  cout << "Here are your instructions...” << endl;
  return 42;
}
```

Function call

```
int main()
{
  int ans;
  ans = welcome_user();
  return 0;
}
```

Functions: no return value

Function definition

```
void welcome_user()
{
  cout << "Welcome, dear user!" << endl;
  cout << "Here are your instructions...” << endl;
  return;
}
```

Function call

```
int main()
{
  welcome_user();
  return 0;
}
```

Stopping things

- `break` – end a switch or loop
- `return` or `return <value>` – end the current function; if `main()`, end program
- `exit(<value>)` – end the entire program from anywhere

Assignment 3: Flight Animator

- Goal: demonstrate good use of **loops** and **functions**
- Include at least 3 functions
- Max length 25 lines per function (except `main()`)
- Output flight duration in hours and minutes
- Output flight cost with a dollar sign and 2 decimal places

- **Read the "implementation tips"!**

What vocabulary did we learn today?

- Keyword `void` (no return value)
- Stop execution: `break, return, exit()`

What ideas and skills did we learn today?

- Characters are integers are characters
- Designing functions
- Functions with multiple parameters
- Functions with default parameter values
- Functions with no parameters
- Functions with no return values

Midterm preview (Midterm: Friday 1/31)

- Covers material through the **end of week 3** (no functions)
- You cannot use cell phones, calculators, tablets, laptops, or other devices, notes, books, Internet access, friends, etc.
- You will be required to sign a Statement of Academic Integrity on the exam for it to be graded
- **Wednesday lecture: Bring your questions**
- **Thursday: Evening review – 6-7 p.m. in KEC 1001**
- **Bring your ID to exam**

Oregon State University
College of Engineering

Week 4 begins!

- ❑ **Prepare for Midterm 1 (Friday, Jan. 31)**
- ❑ Attend lab (laptop required)
- ❑ Read **Rao Lesson 7** (pp. 151-158) - functions
- ❑ Start working on **Assignment 3 design** (due **Sunday, Feb. 2**)
- ❑ Play with **Edabit**: https://edabit.com/challenges
  - CS 161 Week 4 collection: https://tinyurl.com/cs161-week4
  - When you finish a challenge, look at other solutions
  - Ensure you select “C++” in the language drop-down (defaults to JavaScript)

See you Wednesday!

❑ Bring: your midterm questions