CS 161
Introduction to CS I
Lecture 27

• Command-line arguments
• File input and output
Week 10 tips

• Proficiency demo!
• Check Canvas for any missing grades
  • Notify cs161-020-ta@engr.orst.edu by Wednesday (3/11)
  • Your Canvas grade may not be your final course grade
• Final exam: Monday, 3/16, 6-7:50 p.m., LINC 128
  • All T/F and multiple choice (no short answer)
  • Review Midterm 1 and 2 solutions
  • See additional practice questions for structs and recursion (website)
  • No Thursday review session: review in class instead on Friday
Assignment 6: Train Journey

• Worth 80 points
  • Worth doing if any previous assignment earned < 80 points
  • Worth doing if you want practice with recursion 😊
  • Goal: extend the train_car struct (linked list) to allow passengers to board the train, then simulate a train journey
A note about the stack vs. heap

• I want 1,000,000 train_cars. Where can I get them?

1. /* Static allocation */
2. train_car my_train[1000000];

1. /* Dynamic allocation */
2. train_car* my_train = new train_car[1000000];
3. delete [] my_train;
4. my_train = NULL;
A note about the stack vs. heap

• I want 1,000,000 train_cars. Where can I get them?

```cpp
1. /* Static allocation */
2. train_car my_train[1000000];
```

```cpp
1. /* Dynamic allocation */
2. train_car* my_train = new train_car[1000000];
3. delete [] my_train;
4. my_train = NULL;
```

• => The heap is bigger than the stack
Give the user control over size of train

- Prompt user for n_cars
  1. /* Create my train */
  2. train_car* my_train = new train_car;
  3. my_train->kind = "Engine";
  4. my_train->next_car = NULL; /* be safe! */
  5. cout << "How many cars to add to the train? ";
  6. int n_cars;
  7. cin >> n_cars;
  8. add_cars(my_train, n_cars);

- Great for running test cases... unless you have to test many times
Give the user control over size of train

• Instead of waiting to type input each time, make it part of the command line
  • ./lec27-recur-train-args 1
    EngineCaboose
  • ./lec27-recur-train-args 3
    Engine_***___***_Caboose
  • ./lec27-recur-train-args 5
    Engine_***__***__***__***_Caboose
Give the user control over size of train

- Instead of waiting to type input each time, make it part of the command line

1. int main(int argc, char* argv[]) {
2.    train_car* my_train = new train_car;
3.    my_train->kind = "Engine";
4.    my_train->next_car = NULL;
5.    int n_cars = atoi(argv[1]);
6.    add_cars(my_train, n_cars);
7.    ...
8. }

See lec27-recur-train-args.cpp
Give the user control over size of train

- argc: number of arguments
- argv: array of C-style strings
  - argv[0] = name of executable
  - argv[1] = first user-specified argument
  - ...
- Convert C-style string to integer with atoi()
  - int n_cars = atoi(argv[1]);
- Likewise, atof() for floats
Good practice: check argc first

1. /* Expect and require argc == 2 (one user argument) */
2. if (argc != 2) {
3. cout << "Usage: " << argv[0] << " n_cars" << endl;
4. return 1; /* signal an error */
5. }

• To see the return value of the last command in linux:
  • echo $?
Your turn

• What is the value of `argc` if the user entered this command to run a program?
  
  `./my_prog the quick brown fox`

• What does the 2-D array (`argv`) look like?
Working with files

• File = linear sequence of characters
• Stream = channel on which data is sent or received
  • cin: channel connected to keyboard
  • cout: channel connected to screen
• To work with files, create a file stream
  • #include <fstream>
  • ifstream in_stream;
  • ofstream out_stream;
Write to an output file stream

• It works just like cout

1. ofstream out_stream;
2. out_stream.open("my_output.txt");
3. out_stream << "I am writing to a text file." << endl;
4. out_stream.close();
Read from an input file stream

• It works just like cin

1. string w;
2. int n_words = 0;
3. in_stream.open("my_output.txt");
4. while (in_stream >> w) {
5.     n_words++;
6. }
7. in_stream.close();
8. cout << "Read " << n_words << " words from file." << endl;
Using files with command-line arguments

• ./count_words input.txt
• ./write_opera output.txt
• ./translate input_english.txt output_piglatin.txt
Minute paper

• What can you do now that you could not have done at the start of the term?
  • Not what do you know or have heard of
  • What **skill** or **ability** do you have?
  • Programming? Design? Testing? Debugging?
Week 10 begins!

- Demonstrate your proficiency in lab! Flex your muscles!
- Read:
- Review and study for the **final exam**
- **Assignment 6** (due **Saturday, March 14**)

See you Wednesday!