Command-Line Arguments

Command-Line Arguments

- So far, all of your programs have begun pretty much the same way.

```
int main(void)
{
```

- Since we've been collecting user input through in-program prompts, we haven't needed to modify this declaration of `main()`.

- If we want the user to provide data to our program before the program starts running, we need a new form.

Command-Line Arguments

- To collect so called **command-line arguments** from the user, declare main as:

```
int main(int argc, string argv[])
{
```

- These two special arguments enable you to know what data the user provided at the command line and how much data they provided.

Command-Line Arguments

- `argc` (argument count)
  - This integer-type variable will store the **number** of command-line arguments the user typed when the program was executed.

command	argc
`./greedy`	1
`./greedy 1024 cs50`	3

Command-Line Arguments

- `argv` (argument vector)
  - This array of strings stores, one string per element, the actual text the user typed at the command-line when the program was executed.
  - The first element of `argv` is always found at `argv[0]`. The last element of `argv` is always found at `argv[argc-1]`.
    - Do you see why?

Command-Line Arguments

- `argv` (argument vector)
  - Let's assume the user executes the greedy program as follows

```
./greedy 1024 cs50
```

argv indices	argv contents
`argv[0]`	"`./greedy`"
`argv[1]`	
`argv[2]`	
`argv[3]`	

Command-Line Arguments

- `argv` (argument vector)
  - Let's assume the user executes the greedy program as follows

```
./greedy 1024 cs50
```

argv indices	argv contents
`argv[0]`	"`./greedy`"
`argv[1]`	"`1024`"
`argv[2]`	
`argv[3]`	

Command-Line Arguments

- `argv` (argument vector)
  - Let's assume the user executes the greedy program as follows

```
./greedy 1024 cs50
```

argv indices	argv contents
`argv[0]`	`"./greedy"`
`argv[1]`	`"1024"`
`argv[2]`	`"cs50"`
`argv[3]`	

Command-Line Arguments

- `argv` (argument vector)
  - Let's assume the user executes the greedy program as follows

```
./greedy 1024 cs50
```

argv indices	argv contents
`argv[0]`	"`./greedy`"
`argv[1]`	"`1024`"
`argv[2]`	"`cs50`"
`argv[3]`	`???`