COMP 1234

Assignment 1

Design

Pat Doe

A00000000

Feb 30th, 2020

[**Purpose 3**](#_imvoz5885gqn)

[**Data Types 3**](#_eply4qfzki5f)

[Arguments 3](#_uiw1cphccarh)

[Settings 3](#_j0nocaw9wm6u)

[Context 3](#_9iy9hutmp0xh)

[**Functions 4**](#_tcijuvfjxrde)

[**States 4**](#_ozlq9gt58y9w)

[**State Table 4**](#_krkms7lsurgr)

[**State Transition Diagram 6**](#_sixy75hmpkrl)

[**Pseudocode 7**](#_3aozcxyvj5r1)

[parse\_arguments 7](#_ysj48oonrabz)

[Parameters 7](#_3nt5syt5w301)

[Return 7](#_z5irfvv24ezx)

[Pseudo Code 7](#_8pizq1lhwc04)

[handle\_arguments 7](#_z00d7cfr90e5)

[Parameters 7](#_dsnrrhxd724s)

[Return 8](#_r79q6alcyq2r)

[Pseudo Code 8](#_np6c4p3bq9m7)

[usage 8](#_lt3tuwjfpltw)

[Parameters 8](#_vuvsgb5gkj1k)

[Return 8](#_nf9cqxmmhwpn)

[Pseudo Code 8](#_bd14opr7n4jl)

[display\_messages 9](#_y8ttauc9ywql)

[Parameters 9](#_iscxhtwlpdl3)

[Return 9](#_peaoyqmim5a9)

[Pseudo Code 9](#_8cyzb1p3sza2)

[display\_message 9](#_yvxzpn41moju)

[Parameters 9](#_ylm1h05u2uvu)

[Return 9](#_hy3buh82jbet)

[Pseudo Code 9](#_qmp3vztw0f9i)

[cleanup 9](#_xw1d2nbfg9iz)

[Parameters 9](#_mcduy46drst9)

[Return 10](#_qo84rowq3yt)

[Pseudo Code 10](#_tkx2jm47orxn)

# Purpose

* This program accepts 2 arguments from the command line:
  + -c <count>
  + <message>
* It prints the <message> to the console <count> times.

# Data Types

## Arguments

Purpose: To hold the unparsed command-line argument information

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| argc | integer | The number of arguments |
| argv | string[] | The arguments |
| program\_name | string | The name of the program |
| count | string | The number of times to display the message |
| message | string | The message to display |

## Settings

Purpose: To hold the settings the program needs to run.

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| count | unsigned integer | The number of times to display the message |
| message | string | The message to display |

## Context

Purpose: To hold the arguments, settings, and exit information

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| arguments | Arguments | The command line arguments |
| settings | Settings | The parsed command line arguments |
| exit\_code | integer | The exit code of the program |
| exit\_message | string | The error message to print before exiting |

# Functions

| **Function** | **Description** |
| --- | --- |
| parse\_arguments | Parse the command line arguments |
| handle\_arguments | Verify the command line arguments |
| usage | Display a usage message when the command line argument has an issue |
| display\_messages | Display the message passed on the command line the specified number of times |
| display\_message | Display the message passed on the command line |
| cleanup | Free up memory before exiting |

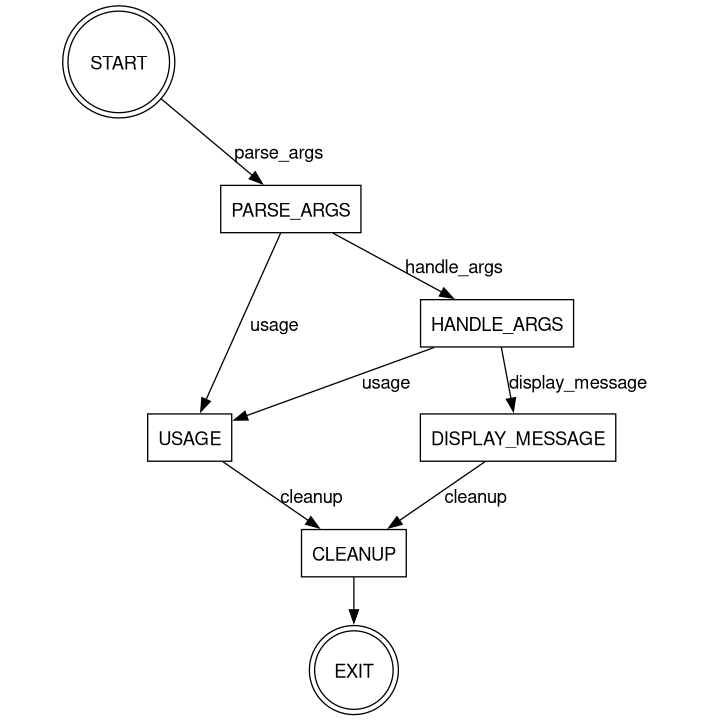
# States

| **State** | **Description** |
| --- | --- |
| PARSE\_ARGS | Parse command line arguments |
| HANDLE\_ARGS | Verify and convert the command line arguments for use |
| USAGE | Display an error message when the command line arguments have an issue |
| DISPLAY\_MESSAGES | Display the message passed on the command line the specified number of times |
| CLEANUP | Cleanup before exit |

# State Table

| **From State** | **To State** | **Function** |
| --- | --- | --- |
| START | PARSE\_ARGS | parse\_arguments |
| PARSE\_ARGS | HANDLE\_ARGS | handle\_arguments |
| PARSE\_ARGS | USAGE | usage |
| HANDLE\_ARGS | DISPLAY\_MESSAGES | display\_messages |
| HANDLE\_ARGS | USAGE | usage |
| USAGE | CLEANUP | cleanup |
| DISPLAY\_MESSAGES | CLEANUP | cleanup |
| CLEANUP | EXIT | - |

# State Transition Diagram



# Pseudocode

## parse\_arguments

### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| ctx | Context | The program context |

### Return

| **Value** | **Reason** |
| --- | --- |
| HANDLE\_ARGS | The args passed are all known |
| USAGE | There were invalid arguments passed |

### Pseudo Code

set ctx.arguments.program\_name to arguments.argv[0]

call getopt in a loop

if opt is ‘h’

call usage

if opt is ‘c’

set ctx.arguments.count to ctx.argument.argv[optind]

If opt is ‘?’

call usage

If there are more than 1 unnamed argument

return USAGE

ctx.arguments.message = ctx.arguments.argv[optind]

return HANDLE\_ARGS

## handle\_arguments

### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| ctx | Context | The program context |

### Return

| **Value** | **Reason** |
| --- | --- |
| DISPLAY\_MESSAGE | The args are correct |
| USAGE | There were issues with the args |

### Pseudo Code

if ctx.arguments.count is not set

set ctx.settings.count to 1

else

set ctx.setting.count to convert count to an unsigned integer

if the conversion fails

return USAGE

if ctx.arguments.message is not set

return USAGE

if ctx.arguments.message is empty

return USAGE

set ctx.settings.message to ctx.arguments.message

return DISPLAY\_MESSAGE

## usage

### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| ctx | Context | The program context |

### Return

* CLEANUP

### Pseudo Code

if ctx.exit\_message is present

print the message to stderr

print the program name and how to run the program to stderr

## display\_messages

### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| ctx | Context | The program context |

### Return

* CLEANUP

### Pseudo Code

loop ctx.settings.count times with iteration

call display\_message(iteration, ctx.settings.message)

return cleanup

## display\_message

### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| iteration | unsigned integer | The iteration |
| message | string | The message to display |

### Return

* Nothing

### Pseudo Code

print the iteration and message to stdout

## cleanup

### Parameters

| **Parameter** | **Type** | **Description** |
| --- | --- | --- |
| ctx | Context | The program context |

### Return

* EXIT

### Pseudo Code

Free any memory in the context

return EXIT