SUM & MEAN

0.0.1

Generated by Doxygen 1.8.13

Contents

1 SUM & MEAN **1**

2 File Index **5**

2.1 File List . . . 5

3 File Documentation **7**

3.1 README.md File Reference . . . 7

3.2 sumean.c File Reference . . . 7

3.2.1 Detailed Description . . . 8

3.2.2 LICENSE . . . 8

3.2.3 Function Documentation . . . 8

3.2.3.1 fpurge_stdin() . . . 8

3.2.3.2 isanumber() . . . 8

3.2.3.3 main() . . . 9

3.2.3.4 PAUSE() . . . 10

3.2.3.5 present() . . . 10

3.2.4 Variable Documentation . . . 10

3.2.4.1 count . . . 10

3.2.4.2 point . . . 10

3.2.4.3 skip . . . 11

3.2.4.4 temp . . . 11

3.2.4.5 w . . . 11

3.3 sumeanp.c File Reference . . . 11

3.3.1 Function Documentation . . . 12

3.3.1.1 fpurge_stdin() . . . 12

3.3.1.2 isanumber() . . . 12

3.3.1.3 main() . . . 12

3.3.1.4 PAUSE() . . . 13

3.3.1.5 present() . . . 13

3.3.2 Variable Documentation . . . 13

3.3.2.1 count . . . 13

3.3.2.2 point . . . 13

3.3.2.3 skip . . . 13

3.3.2.4 temp . . . 13

3.3.2.5 w . . . 13

Index 15

Chapter 1

SUM & MEAN

simple terminal app example in c language

Introduction

This programming example accepts in input a list of positive integers and computes their sum and average, showing them as output. At the end of the execution it shows the number **m** of input 1-D points, their sum

and their average value

The list can be read from standard input (keyboard or `file redirection`).

The `log` of what's going on with the computation is printed on the standard error output stream, while the actual numerical output is printed on the standard output at the end. This allows to use the app output as input of a following app, that read from the standard input. The output created on the standard input and/or the output created on the standard error can be also saved in different text files.

How To

The following instructions are given supposing you're yousing the GNU `gcc` c language compiler. You need to use a terminal emulator, like xterm on MacOS or gnome-terminal or konsole on GNU-Linux or `cmd` terminal on Microsoft Windows. Not all the instructions are given for all different operating systems but understanding the main examples, they may be derived easily

to compile on MSwin mingw32:

mingw32-make -f Makefile.win sumean

to compile on linux/MacOS/MS-Win+msys2:

make sumean

to run on MS-Win cmd shell

sumean

to remove all created files in MS-Win cmd shell

del sumean.o sumean ∗.log ∗.asc

to run on linux/MacOS/MS-Win+msys2

./sumean

to run on linux/MacOS/MS-Win+msys2 saving the results on an output file

./sumean >out.asc

to run on linux/MacOS/MS-Win+msys2 & logging

saving the instructions on an output file err.log saving the results on an output file out.asc this implies you giving input without hints (blind mode)

./sumean 2>err.log >out.asc

to remove all created files in GNU-linux/MacOS/MS-win+msys2

rm sumean.o sumean ∗.log ∗.asc

Using file explorer and mouse

the source file sumeanp.c is just the same app code with added a feature useful to run the app from file explorer graphical user interface. The letter p at the end of the file name means pause. After compiling you can just click on the sumeanp exe. It is named sumeanp.exe but take in mind that using Microsoft Windows file explorer the extension is written in a different column in the details view, or you can distinguish the file type from the different icon. Just compile the source.

to compile on GNU-Linux/MacOS:

make sumeanp

to compile on MS-Win:

mingw32-make -f Makefile.win sumeanp

Installation

The sumean app do not really need to be installed permanently on the system to be executed and exercise with it, it can be downloaded and saved in any folder of your computer, then compiled and run. Of course you need the permission to write on that folder, having it in a subfolder of the Desktop is ok. Some antiviruses and security systems on some OSs may wrongly detect the generated exe as harmful, take care of instruct or temporary stop/disable that systems. Many modern GNU-Linux distribution do not allow as default to run a binary exe app clicking on it with the mouse by the file browser. The Gnome file browser `nautilus` (now called only Files) can be instructed to do it, but the safer way to run it from Graphical User Interface (GUI) on GNU-Linux is to use a .desktop file as launcher and clicking on it instead. The file install.sh and uninstall.sh are there for that purpose, for people that wonna try this optional method.

Troubleshooting

In the simpler case, we're supposing that the gcc compiler executable can be called with the following instruction: `gcc sumean.c -o sumean` , which implies that somewhere in the list of the PATH environment variable is included a path to a folder containing the gcc executable file or a link to it. In the simpler case in ∗nix like OSs it is called just gcc and in Microsoft Windows it should be called gcc.exe .

Modern mingw32-gcc releases have that filename called

`x86_64-w64-mingw32-gcc.exe` for the 64 bit version and `i686-w64-mingw32-gcc.exe` for the 32 bit version, but may also include a copy called `gcc.exe` as well.

Documentation

Beyond this project page the following documentation is available

1. This doc in pdf version `ReadMe in pdf format`
2. Reference Manual `Ref Manual`
3. `Web Reference`

Computing Steps

The following are the processing steps executed by the program.

- Present to the user what the app is doing
- Tell user to enter data points; maximum of allowed pts = 39
- Start an infinite loop, flow interrupted only by break;
  - Read a value; end of file (EOF val = -1) will be checked
  - Value input before is in string format, if the string correctly express a legal numerical value, convert it into integer
  - Insert the read item in the 1-D points array and increment the partial sum with its value
  - Reject the item value otherwise
  - check when the end of the dataset is reached (reading -1)
  - if -1 read –> break; continue the loop otherwise
- EOF detected : compute average
- print final report #points, sum, average.

TODOs

Execution examples with remarkable datasets can be added. Example of how to run it and catch the app output from another app reading from stdin will be useful to show how to create a simple software pipeline. Another useful exeample to add will be using an external app that read the sumean output and graph the points succession and the mean value, and the sum.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

sumean.c
SUM & MEAN : simple terminal app example in c language 7
sumeanp.c 11

Chapter 3

File Documentation

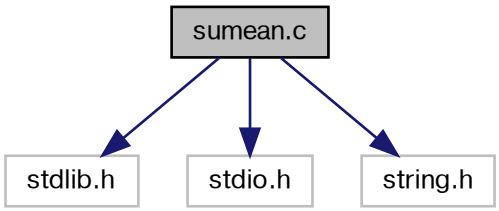
3.1 README.md File Reference

3.2 sumean.c File Reference

SUM & MEAN : simple terminal app example in c language.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

Include dependency graph for sumean.c:



Functions

- void PAUSE ()

  pause the execution until a key and enter are pressed
- void fpurge_stdin ()
- int isanumber (char s[ ])
- void present ()
- int main (int argc, char ∗∗argv)

Variables

- char temp [1024]
- long int point [40]
- unsigned char w =0
- unsigned long count =0
- unsigned char skip =0

3.2.1 Detailed Description

SUM & MEAN : simple terminal app example in c language.

- Author: Francesco Lazzarotto francesco.lazzarotto@inaf.it
- Date: 12/12/2021
- Version: 0.0.1
- Compiler: gcc (iso c99)
- Target: provide a programming example
- Notes: TBD

3.2.2 LICENSE

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

3.2.3 Function Documentation

3.2.3.1 fpurge_stdin()

```
void fpurge_stdin ( )
```

fpurge_stdin() purge the stdin for terminal I/O

3.2.3.2 isanumber()

```
int isanumber (
            char s[] )
```

isanumber() check if a given string is a positive integer

Parameters

char	s[] characters array containing the string to check

Returns

returns 1 if the string is a positive integer, 0 if it is not

3.2.3.3 main()

```
int main (
              int argc,
              char ** argv )
```

main() main function of the app

This example accepts a list of positive integers And computes their sum and average. The list can be read from stdin (keyboard or file redirection)

- Present to the user what the app is doing
- Tell user to enter data points; maximum of allowed pts = 39
- Start an infinite loop, flow interrupted only by break;
- Read a value; check for end of file (EOF val = -1)
- Value input before is in string format
- If the string correctly express a legal numerical value, convert it into integer and then insert it in the 1-D points array and increment the partial sum with its value
- Reject the item value otherwise
- check when the end of the dataset is reached (reading -1)
- -1 read –> break; continue the loop otherwise
- EOF detected –> compute average
- print final report #points, sum, average.

main() main function of the app

This example accepts a list of positive integers And computes their sum and average. The list can be read from stdin (keyboard or file redirection)

- Present to the user what the app is doing
- Tell user to enter data points; maximum of allowed pts = 39
- Start an infinite loop, flow interrupted only by break;
- Read a value; check for end of file (EOF val = -1)
- Value input before is in string format

- If the string correctly express a legal numerical value, convert it into integer and then insert it in the 1-D points array and increment the partial sum with its value
- Reject the item value otherwise
- check when the end of the dataset is reached (reading -1)
- -1 read –> break; continue the loop otherwise
- EOF detected –> compute average
- print final report #points, sum, average.
- **call PAUSE() and wait user input to close the terminal**

3.2.3.4 PAUSE()

```
void PAUSE ( )
```

pause the execution until a key and enter are pressed

PAUSE()

3.2.3.5 present()

```
void present ( )
```

present() print a description of what app is doing and usage

3.2.4 Variable Documentation

3.2.4.1 count

```
unsigned long count =0
```

3.2.4.2 point

```
long int point[40]
```

3.2.4.3 skip

```
unsigned char skip =0
```

3.2.4.4 temp

```
char temp[1024]
```

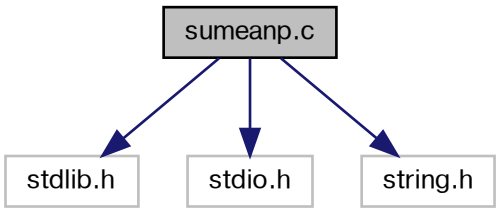
3.2.4.5 w

```
unsigned char w =0
```

3.3 sumeanp.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

Include dependency graph for sumeanp.c:



Functions

- void PAUSE ()
- void fpurge_stdin ()
- int isanumber (char s[ ])
- void present ()
- int main (int argc, char ∗∗argv)

Variables

- char temp [1024]
- long int point [40]
- unsigned char w =0
- unsigned long count =0
- unsigned char skip =0

3.3.1 Function Documentation

3.3.1.1 fpurge_stdin()

```
void fpurge_stdin ( )
```

3.3.1.2 isanumber()

```
int isanumber (
              char s[] )
```

3.3.1.3 main()

```
int main (
              int argc,
              char ** argv )
```

main() main function of the app

This example accepts a list of positive integers And computes their sum and average. The list can be read from stdin (keyboard or file redirection)

- Present to the user what the app is doing
- Tell user to enter data points; maximum of allowed pts = 39
- Start an infinite loop, flow interrupted only by break;
- Read a value; check for end of file (EOF val = -1)
- Value input before is in string format
- If the string correctly express a legal numerical value, convert it into integer and then insert it in the 1-D points array and increment the partial sum with its value
- Reject the item value otherwise
- check when the end of the dataset is reached (reading -1)
- -1 read –> break; continue the loop otherwise
- EOF detected –> compute average
- print final report #points, sum, average.
- **call PAUSE() and wait user input to close the terminal**

3.3.1.4 PAUSE()

```
void PAUSE ( )
```

3.3.1.5 present()

```
void present ( )
```

3.3.2 Variable Documentation

3.3.2.1 count

```
unsigned long count =0
```

3.3.2.2 point

```
long int point[40]
```

3.3.2.3 skip

```
unsigned char skip =0
```

3.3.2.4 temp

```
char temp[1024]
```

3.3.2.5 w

```
unsigned char w =0
```

Index

count
    sumean.c, 10
    sumeanp.c, 13

fpurge_stdin
    sumean.c, 8
    sumeanp.c, 12

isanumber
    sumean.c, 8
    sumeanp.c, 12

main
    sumean.c, 9
    sumeanp.c, 12

PAUSE
    sumean.c, 10
    sumeanp.c, 12
point
    sumean.c, 10
    sumeanp.c, 13
present
    sumean.c, 10
    sumeanp.c, 13

README.md, 7

skip
    sumean.c, 10
    sumeanp.c, 13
sumean.c, 7
    count, 10
    fpurge_stdin, 8
    isanumber, 8
    main, 9
    PAUSE, 10
    point, 10
    present, 10
    skip, 10
    temp, 11
    w, 11
sumeanp.c, 11
    count, 13
    fpurge_stdin, 12
    isanumber, 12
    main, 12
    PAUSE, 12
    point, 13
    present, 13
    skip, 13
    temp, 13
    w, 13

temp
    sumean.c, 11
    sumeanp.c, 13

w
    sumean.c, 11
    sumeanp.c, 13