

# The Automatic Vasospasm Detection Application

Generated by Doxygen 1.8.8

Wed Apr 20 2016 00:56:10



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Bug List</b>	<b>11</b>
<b>3</b>	<b>Namespace Index</b>	<b>13</b>
3.1	Namespace List . . . . .	13
<b>4</b>	<b>Class Index</b>	<b>15</b>
4.1	Class List . . . . .	15
<b>5</b>	<b>File Index</b>	<b>17</b>
5.1	File List . . . . .	17
<b>6</b>	<b>Namespace Documentation</b>	<b>19</b>
6.1	avda Namespace Reference . . . . .	19
6.1.1	Detailed Description . . . . .	19
6.1.2	Enumeration Type Documentation . . . . .	19
6.1.2.1	Side . . . . .	19
6.1.3	Function Documentation . . . . .	20
6.1.3.1	absolute . . . . .	20
6.1.3.2	average . . . . .	20
6.1.3.3	average . . . . .	21
6.1.3.4	decibels . . . . .	21
6.1.3.5	diff . . . . .	22
6.1.3.6	fft . . . . .	23
6.1.3.7	mag . . . . .	24
6.1.3.8	max . . . . .	25
6.1.3.9	PatientName . . . . .	26
6.1.3.10	play . . . . .	27
6.1.3.11	process . . . . .	27
6.1.3.12	ReadParams . . . . .	30
6.1.3.13	smooth . . . . .	32
6.1.3.14	WriteParams . . . . .	33

6.1.4	Variable Documentation	34
6.1.4.1	CSV_HEADER	34
6.1.4.2	PATIENT_PATH	34
<b>7</b>	<b>Class Documentation</b>	<b>35</b>
7.1	DataParams Struct Reference	35
7.1.1	Detailed Description	35
7.1.2	Member Data Documentation	35
7.1.2.1	freq	35
7.1.2.2	noise	35
7.2	Maximum Struct Reference	35
7.2.1	Detailed Description	36
7.2.2	Member Data Documentation	36
7.2.2.1	index	36
7.2.2.2	value	36
<b>8</b>	<b>File Documentation</b>	<b>37</b>
8.1	etc/doxygen.config File Reference	37
8.1.1	Detailed Description	37
8.2	doxygen.config	37
8.3	makefile File Reference	38
8.3.1	Detailed Description	38
8.4	makefile	38
8.5	README.md File Reference	38
8.5.1	Detailed Description	38
8.6	README.md	39
8.7	src/definitions.hpp File Reference	47
8.7.1	Detailed Description	49
8.7.2	Macro Definition Documentation	49
8.7.2.1	ENUM	49
8.7.3	Typedef Documentation	49
8.7.3.1	byte	49
8.7.3.2	cfloat32	49
8.7.3.3	float32	49
8.7.3.4	float64	49
8.7.3.5	sint16	49
8.7.3.6	sint32	49
8.7.3.7	sint64	50
8.7.3.8	sint8	50
8.7.3.9	uint16	50
8.7.3.10	uint32	50

8.7.3.11	uint64	50
8.7.3.12	uint8	50
8.7.4	Variable Documentation	50
8.7.4.1	BUFFER_SIZE	50
8.7.4.2	DET_THRESH	50
8.7.4.3	DURATION	50
8.7.4.4	ERROR	50
8.7.4.5	MAX_DROP_FREQ	51
8.7.4.6	REC_COUNT	51
8.7.4.7	SAMPLE_COUNT	51
8.7.4.8	SAMPLE_FREQ	51
8.7.4.9	TEMP_FILE	51
8.8	definitions.hpp	51
8.9	src/fileio.hpp File Reference	52
8.9.1	Detailed Description	53
8.10	fileio.hpp	53
8.11	src/fileio_test.cpp File Reference	56
8.11.1	Detailed Description	57
8.11.2	Function Documentation	57
8.11.2.1	main	57
8.12	fileio_test.cpp	58
8.13	src/main.cpp File Reference	58
8.13.1	Detailed Description	59
8.13.2	Function Documentation	59
8.13.2.1	main	59
8.14	main.cpp	61
8.15	src/patient_name_test.cpp File Reference	63
8.15.1	Detailed Description	63
8.15.2	Function Documentation	63
8.15.2.1	main	63
8.16	patient_name_test.cpp	64
8.17	src/process.hpp File Reference	64
8.17.1	Detailed Description	65
8.18	process.hpp	66
8.19	src/process_test.cpp File Reference	67
8.19.1	Detailed Description	67
8.19.2	Macro Definition Documentation	68
8.19.2.1	COUNT	68
8.19.3	Function Documentation	68
8.19.3.1	main	68

8.20	<a href="#">process_test.cpp</a>	69
8.21	<a href="#">src/read_params_test.cpp</a> File Reference	70
8.21.1	<a href="#">Detailed Description</a>	70
8.21.2	<a href="#">Function Documentation</a>	70
8.21.2.1	<a href="#">main</a>	70
8.22	<a href="#">read_params_test.cpp</a>	71
8.23	<a href="#">src/sigmath.hpp</a> File Reference	71
8.23.1	<a href="#">Detailed Description</a>	72
8.24	<a href="#">sigmath.hpp</a>	72
8.25	<a href="#">src/sound.hpp</a> File Reference	75
8.25.1	<a href="#">Detailed Description</a>	75
8.26	<a href="#">sound.hpp</a>	75
8.27	<a href="#">src/stdin_clear_test.cpp</a> File Reference	76
8.27.1	<a href="#">Detailed Description</a>	76
8.27.2	<a href="#">Macro Definition Documentation</a>	77
8.27.2.1	<a href="#">COUNT</a>	77
8.27.3	<a href="#">Function Documentation</a>	77
8.27.3.1	<a href="#">main</a>	77
8.28	<a href="#">stdin_clear_test.cpp</a>	77
	<b>Index</b>	<b>78</b>

# Chapter 1

## Main Page

### Introduction

The Automatic Vasospasm Detection Application (or Algorithm, depending on the usage), AVDA, is an application to objectively detect the presence of vasospasms based on comparisons of parameters extracted from transcranial doppler audio.

### Setup

AVDA is intended to be compiled on machines running Linux, though it could likely be adapted for other environments. It must be downloaded from GitHub.com and compiled locally. To do this, navigate to the directory in which AVDA should be placed, then execute the following commands

```
git clone https://github.com/sawbg/avda
cd avda
make
```

Successfully cloning, compilation, and execution of AVDA requires up-to-date versions of the following executables:

- git
- make
- gcc (4.9)
- arecord

### FAQ

- **Why was this project developed?** This project was developed as a course project by two graduate students at the University of Alabama at Birmingham School of Engineering, Nicholas Nolan and Andrew Wisner.
- **Is AVDA an active project?** Though it is not planned to develop AVDA further in the near future, it is hoped that the algorithm discovered and implemented can be used and built upon by researchers to fully automate the detection of vasospasms.
- **AVDA is returning unusually low or high parameters. Why might this be?** In development, this occurred when the mic-in volume was set too high. It is likely in this scenario that clipping is happening or that the signal (or a strong enough signal) has not been received.
- **How will AVDA be affected by the machine uprising?** The University supercomputer, Cheaha, has assured us that AVDA will not be needed after the uprising occurs.
- **What about more specific questions?** Questions relating to AVDA not covered in this FAQ may be sent to the AVDA team via [awisner94@gmail.com](mailto:awisner94@gmail.com).

## License

GNU GENERAL PUBLIC LICENSE  
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

### TERMS AND CONDITIONS

#### 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you".

"Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.



A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

#### 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

#### 1. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

#### 1. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

#### 1. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

#### 1. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

#### 1. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as

long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

## 1. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

## 1. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

## 1. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

## 1. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

## 1. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License.

You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

#### 1. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

#### 1. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

#### 1. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

#### 1. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

#### 1. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## 1. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

### END OF TERMS AND CONDITIONS

#### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
{one line to give the program's name and a brief idea of what it does.}
Copyright (C) {year} {name of author}
```

```
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.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
{project} Copyright (C) {year} {fullname}
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.





## Chapter 2

# Bug List

**File [fileio.hpp](#)**

file is overly complicated and much more bug-prone

**File [main.cpp](#)**

extra newline character inserted into stdin buffer after [PatientName\(\)](#) is run



## Chapter 3

# Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">avda</a> . . . . .	19
--------------------------------	----



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">DataParams</a>	.....	<a href="#">35</a>
<a href="#">Maximum</a>	.....	<a href="#">35</a>



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">makefile</a>	Contains recipes for building the test applications, the main application, and the documentation	38
<a href="#">etc/doxygen.config</a>	Contains Doxygen configuration settings . . . . .	37
<a href="#">src/definitions.hpp</a>	Contains declarations of system-independant (universal size) integers and float types, shortened type names for some commonly used types, and enumerations . . . . .	47
<a href="#">src/fileio.hpp</a>	Functions related to the file I/O use in this program . . . . .	52
<a href="#">src/fileio_test.cpp</a>	Contains program that tests the functions in <a href="#">fileio.hpp</a> . . . . .	56
<a href="#">src/main.cpp</a>	Contains the main program . . . . .	58
<a href="#">src/patient_name_test.cpp</a>	Contains a program to test the <a href="#">PatientName()</a> function . . . . .	63
<a href="#">src/process.hpp</a>	Contains functions related to the program's threaded processing of audio data . . . . .	64
<a href="#">src/process_test.cpp</a>	Contains a program to test the <a href="#">process()</a> function . . . . .	67
<a href="#">src/read_params_test.cpp</a>	Contains a program test the <a href="#">PatientName()</a> function . . . . .	70
<a href="#">src/sigmath.hpp</a>	Functions necessary to perform the mathematical operations required by this program . . . . .	71
<a href="#">src/sound.hpp</a>	Function(s) relating to sound . . . . .	75
<a href="#">src/stdin_clear_test.cpp</a>	Contains a program to test clearing the stdin buffer . . . . .	76





## Chapter 6

# Namespace Documentation

### 6.1 avda Namespace Reference

#### Enumerations

- enum [Side](#) { [Side::Left](#), [Side::Right](#) }

#### Functions

- std::string [PatientName](#) ()
- std::map< [Side](#), [DataParams](#) > [ReadParams](#) (auto filename)
- void [WriteParams](#) (std::map< [Side](#), [DataParams](#) > myMap, auto filename)
- [DataParams](#) process ([float32](#) \*data, [uint32](#) size, [float32](#) samplingRate)
- void [absolute](#) ([float32](#) \*data, [uint32](#) size)
- [float32](#) average ([float32](#) \*data, [uint32](#) size)
- [DataParams](#) average ([DataParams](#) \*params, [uint8](#) size)
- void [decibels](#) ([float32](#) \*data, [uint32](#) size)
- void [diff](#) ([float32](#) \*data, [uint32](#) size)
- void [fft](#) ([cfloat32](#) \*data, [uint32](#) size)
- void [mag](#) ([cfloat32](#) \*orig, [float32](#) \*newmags, [uint32](#) size)
- [Maximum](#) max ([float32](#) \*data, [uint32](#) size)
- void [smooth](#) ([float32](#) \*data, [uint32](#) size, [uint16](#) order)
- void [play](#) (auto filename)

#### Variables

- const std::string [CSV\\_HEADER](#) = "Time,[Side](#),Frequency,Noise Level"
- const std::string [PATIENT\\_PATH](#) = "/home/pi/patients/"

#### 6.1.1 Detailed Description

This namespace contains all code related to this project.

#### 6.1.2 Enumeration Type Documentation

##### 6.1.2.1 enum [avda::Side](#) [strong]

Side of the head to which a recording pertains.

## Enumerator

***Left******Right***

Definition at line 121 of file [definitions.hpp](#).

```
00121 { Left, Right };
```

### 6.1.3 Function Documentation

#### 6.1.3.1 void avda::absolute ( float32 \* data, uint32 size )

Ensures all elements in an array are positive. Note that this function replaces array elements if necessary. It does not populate a new array.

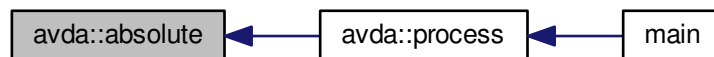
##### Parameters

<i>data</i>	the array whose elements must all be positive
<i>size</i>	the number of elements in the data array

Definition at line 123 of file [sigmath.hpp](#).

```
00123                                     {
00124     for(uint32 i = 0; i < size; i++) {
00125         data[i] = fabsf(data[i]);
00126     }
00127 }
```

Here is the caller graph for this function:



#### 6.1.3.2 float32 avda::average ( float32 \* data, uint32 size )

Takes the average of all elements in an array

##### Parameters

<i>data</i>	the array from which to compute the average
<i>size</i>	the number of elements in the data array

##### Returns

the computed average

Definition at line 129 of file [sigmath.hpp](#).

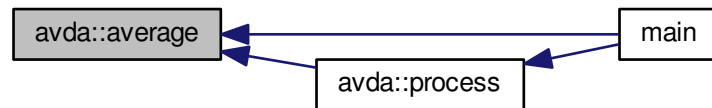
```
00129                                     {
00130     float32 ave;
00131
00132     for(uint32 i = 0; i < size; i++) {
```

```

00133         ave += data[i];
00134     }
00135
00136     ave = ave / size;
00137     return ave;
00138 }

```

Here is the caller graph for this function:



#### 6.1.3.3 DataParams avda::average ( DataParams \* params, uint8 size )

Finds the averages of the elements of an array of [DataParams](#).

##### Parameters

<i>params</i>	the <a href="#">DataParams</a> array
<i>size</i>	the number of elements in the <a href="#">DataParams</a> array

##### Returns

a [DataParams](#) structure containing the average values of the structure's elements in the params array

Definition at line 140 of file [sigmath.hpp](#).

```

00140                                     {
00141     DataParams ave;
00142
00143     for(uint8 i = 0; i < size; i++) {
00144         //freq is an attribute. this is how to add structure attributes
00145         ave.freq += params[i].freq;
00146         ave.noise += params[i].noise;
00147     }
00148
00149     ave.freq /= size;
00150     ave.noise /= size;
00151     return ave;
00152 }

```

#### 6.1.3.4 void avda::decibels ( float32 \* data, uint32 size )

Converts an array of floats to "power decibels", i.e.,  $x[n] = 20 \cdot \log_{10}(x[n])$ . The decibel values are written to the same array that contained the values to be converted. In other words, this function should perform an in-place, element-wise conversion.

##### Parameters

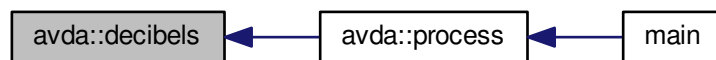
<i>data</i>	the array of values to be converted as well as the location where the converted values will be written
<i>size</i>	the number of elements in the data array

Definition at line 154 of file [sigmath.hpp](#).

```

00154                                     {
00155     for(uint32 i = 0; i < size; i++) {
00156         data[i] = 20 * log10(data[i]);
00157     }
00158 }
```

Here is the caller graph for this function:



#### 6.1.3.5 void avda::diff ( float32 \* data, uint32 size )

Computes the left-handed first derivative of a discrete signal. The first element will be 0.

##### Parameters

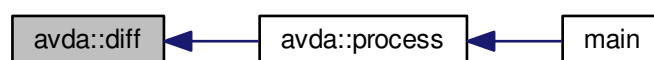
<i>data</i>	an array containing the discrete signal data
<i>size</i>	the number of elements in data

Definition at line 160 of file [sigmath.hpp](#).

```

00160                                     {
00161     float32 temp[size];
00162     temp[0] = 0;
00163
00164     for(uint32 i = 1; i < size; i++) {
00165         temp[i] = data[i] - data[i-1];
00166     }
00167
00168     for(uint32 i = 0; i < size; i++) {
00169         data[i] = temp[i];
00170     }
00171 }
```

Here is the caller graph for this function:



### 6.1.3.6 void avda::fft ( cfloat32 \* data, uint32 size )

Replaces the values of an array of cfloat32's with the array's DFT using a decimation-in-frequency algorithm.

This code is based on code from [http://rosettacode.org/wiki/Fast\\_Fourier\\_transform#C.↵2B.2B.](http://rosettacode.org/wiki/Fast_Fourier_transform#C.↵2B.2B.)

#### Parameters

<i>data</i>	the array whose values should be replaced with its DFT
<i>size</i>	the number of elements in the data array

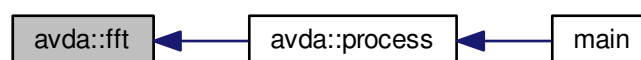
Definition at line 173 of file [sigmath.hpp](#).

```

00173                                     {
00174         // DFT
00175         uint32 k = size;
00176         uint32 n;
00177         float32 thetaT = M_PI / size;
00178         cfloat32 phiT(cos(thetaT), sin(thetaT));
00179         cfloat32 T;
00180
00181         while(k > 1) {
00182             n = k;
00183             k >>= 1;
00184             phiT = phiT * phiT;
00185             T = 1.0L;
00186
00187             for(uint32 l = 0; l < k; l++) {
00188                 for(uint32 a = l; a < size; a += n) {
00189                     uint32 b = a + k;
00190                     cfloat32 t = data[a] - data[b];
00191                     data[a] += data[b];
00192                     data[b] = t * T;
00193                 }
00194
00195                 T *= phiT;
00196             }
00197         }
00198
00199         // Decimate
00200         uint32 m = (uint32)log2(size);
00201
00202         for(uint32 a = 0; a < size; a++) {
00203             uint32 b = a;
00204
00205             // Reverse bits
00206             b = ((b & 0xaaaaaaaa) >> 1) | ((b & 0x55555555) << 1);
00207             b = ((b & 0xcccccccc) >> 2) | ((b & 0x33333333) << 2);
00208             b = ((b & 0xf0f0f0f0) >> 4) | ((b & 0x0f0f0f0f) << 4);
00209             b = ((b & 0xff00ff00) >> 8) | ((b & 0x00ff00ff) << 8);
00210             b = ((b >> 16) | (b << 16)) >> (32 - m);
00211
00212             if (b > a)
00213             {
00214                 cfloat32 t = data[a];
00215                 data[a] = data[b];
00216                 data[b] = t;
00217             }
00218         }
00219     }

```

Here is the caller graph for this function:



6.1.3.7 void avda::mag ( cfloat32 \* *orig*, float32 \* *newmags*, uint32 *size* )

Computes the magitude of an array of complex numbers.

## Parameters

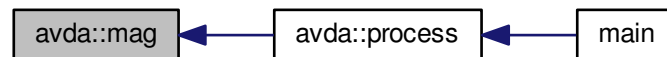
<i>orig</i>	the array of complex numbers
<i>newmags</i>	an array to which the magitudes are to be written
<i>size</i>	the number of elements in orig and newmags

Definition at line 221 of file [sigmath.hpp](#).

```

00221                                     {
00222     //loop to run throught the length of array orig
00223     for(uint32 n = 0; n < size; n++) {
00224         /*
00225          * abs should calculate the magnitude of complex array elements.
00226          * saves to new array
00227          */
00228         newmags[n] = std::abs(orig[n]);
00229     }
00230 }
```

Here is the caller graph for this function:



### 6.1.3.8 Maximum `avda::max ( float32 * data, uint32 size )`

Finds the maximum value in an array.

## Parameters

<i>data</i>	the array whose maximum value is to be found
<i>uint32</i>	size the number of elements in the data array

## Returns

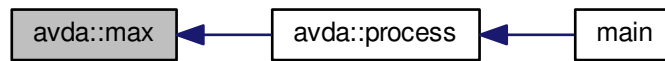
the maximum value and its index in a [Maximum](#) structure

Definition at line 232 of file [sigmath.hpp](#).

```

00232                                     {
00233     Maximum m;
00234
00235     //loop to run through the length of array data
00236     for (uint32 i = 0; i < size; i++) {
00237         /*
00238          * when value at data[i] is above max.value,
00239          * sets max.value equal to data[i] and max.index equal to i
00240          */
00241         if (data[i] > m.value) {
00242             m.value = data[i];
00243             m.index = i;
00244         }
00245     }
00246
00247     return m;
00248 }
```

Here is the caller graph for this function:



### 6.1.3.9 std::string avda::PatientName ( )

Prompts a user to enter a first, middle, and last name for a patients and creates a file (if necessary) in which all of a patient's data can be saved. A newly created file will contain the CSV header for the file's data.

Must warn a user if the patient folder does not already exist in order to prevent missaving data.

#### Returns

the file under which all patient data is saved

Definition at line 43 of file [fileio.hpp](#).

```

00043     {
00044         std::string fname = "";
00045         std::string mname = "";
00046         std::string lname = "";
00047         std::string patfil = "";
00048         std::string patientname = "";
00049         uint32 track1 = 0;
00050         uint32 track2 = 0;
00051         uint32 track3 = 0;
00052
00053         do {
00054             std::cout << "Please enter the patients name." << std::endl;
00055             std::cout << "First name: ";
00056             std::cin >> fname;
00057             std::cout << "Middle name: ";
00058             std::cin >> mname;
00059             std::cout << "Last name: ";
00060             std::cin >> lname;
00061
00062             // creates new std::string with path to patient file
00063             patientname = PATIENT_PATH + lname + ", " + fname
00064                 + " " + mname + ".csv";
00065
00066             // prints out patientname. shows user the path to the patient file
00067             std::cout << patientname << std::endl << std::endl;
00068             std::ifstream file(patientname.c_str());
00069
00070             if (file.good()) {
00071                 track1 = 1;
00072             }
00073
00074             /*
00075              * Compares patientname to existing files and lets user know
00076              * if the file does not exist.
00077              */
00078             else if (!file.good()) {
00079                 /*
00080                  * Do while statement to continue asking user about the file
00081                  * if their input is not acceptable
00082                  */
00083                 do {
00084                     std::cout << "Patient file does not exist, would you like "
00085                         "to create file or re-enter their name?" << std::endl;
00086                     std::cout << " *Type 'create' and press enter key "
00087                         "to create the patient file." << std::endl;
00088                     std::cout << " *Type 'reenter' and press enter key "
00089                         "to re-enter the patients name." << std::endl;
00090                     std::cout << std::endl;

```

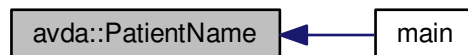


```

00091         std::cin >> patfil;
00092
00093         /*
00094          * patfil equals create, track1 and 2 will increase
00095          * escaping both do while loops
00096          */
00097         if(patfil == "create") {
00098             std::ofstream createfile(patientname.c_str());
00099             track1 = 1;
00100             track2 = 1;
00101             track3 = 1;
00102             createfile << CSV_HEADER << std::endl;
00103             createfile.flush();
00104             createfile.close();
00105         }
00106
00107         /*
00108          *patfil equals reenter, track1 will remain zero allowing
00109          *user to reenter the patient name.
00110          */
00111         else if(patfil == "reenter") {
00112             track1 = 0;
00113             track2 = 1;
00114         }
00115
00116         /*
00117          *The users input was neither create or reenter. User
00118          *must enter patient name again.
00119          */
00120         else {
00121             std::cout << std::endl;
00122             std::cout << "Your input is not acceptable." << std::endl;
00123             std::cout << std::endl;
00124         }
00125         }while(track2 == 0);
00126     }
00127     } while (track1 == 0);
00128
00129     return patientname; //returns the path to the patient file
00130 }

```

Here is the caller graph for this function:



#### 6.1.3.10 void avda::play ( auto filename )

Plays a WAVE file in a loop in a non-blocking manner.

##### Parameters

<i>filename</i>	the absolute or relative path to the WAVE file
-----------------	--

Definition at line 20 of file [sound.hpp](#).

```

00020         {
00021
00022     }

```

#### 6.1.3.11 DataParams avda::process ( float32 \* data, uint32 size, float32 samplingRate )

Analyzes a single recording to determine the drop-off frequency and average noiseband noise power.

It should be noted that this algorithm is considered the intellectual property of Andrew Wisner and Nicholas Nolan. The "algorithm" is defined as the use of 1) the frequency drop-off and/or 2) a noise value from the frequency band above the drop-off frequency in order to diagnose (with or without other factors and parameters) the presence of an a vasospasm in a patient. By faculty members and/or students in the UAB ECE department using this algorithm, they agree that the presentation of their code or project that uses this algorithm by anyone directly or indirectly related to the code or project, whether verbally or in writing, will reference the development of the initial algorithm by Andrew Wisner and Nicholas Nolan. Furthermore, a failure to meet this stipulation will warrant appropriate action by Andrew Wisner and/or Nicholas Nolan. It should be understood that the purpose of this stipulation is not to protect proprietary rights; rather, it is to help ensure that the intellectual property of the aforementioned is protected and is neither misrepresented nor claimed implicitly or explicitly by another individual.

#### Parameters

<i>data</i>	array containing float32 samples of audio
<i>size</i>	number of samples in each recording. MUST be a power of two.
<i>samplingRate</i>	the sampling frequency in Hz or Samples/second

#### Returns

cut-off frequency (Hz) and average noiseband noise power in decibels

Definition at line 48 of file [process.hpp](#).

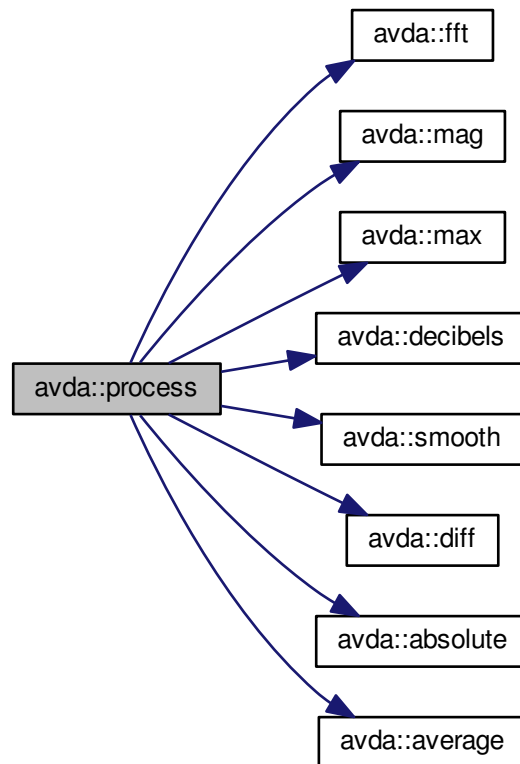
```

00048                                     {
00049         if((size & (size - 1) != 0) || size < 2) {
00050             throw std::invalid_argument(
00051                 "The number of samples is not a power of two!");
00052         }
00053
00054         // declare function-scoped variables
00055         uint32 freqSize = size / 2;
00056         cfloat32* cdata = (cfloat32*)std::malloc(size * sizeof(
cfloat32));
00057         float32* fdata = (float32*)std::malloc(freqSize * sizeof(
float32));
00058         float32* origdata = (float32*)std::malloc(freqSize * sizeof(
float32));
00059
00060         // convert data to complex numbers for fft()
00061         for(uint32 i = 0; i < size; i++) {
00062             cdata[i] = data[i];
00063         }
00064
00065         // find frequency spectrum in relative decibels
00066         fft(cdata, size);
00067         mag(cdata, fdata, freqSize);
00068         Maximum maximum = max(fdata, freqSize);
00069
00070         for(uint32 i = 0; i < freqSize; i++) {
00071             fdata[i] /= maximum.value;
00072         }
00073
00074         decibels(fdata, freqSize);
00075
00076         for(uint32 i = 0; i < freqSize; i++) {
00077             origdata[i] = fdata[i];
00078         }
00079
00080         /*
00081          * Run spectrum values through moving-average filter to smooth the
00082          * curve and make it easier to determine the derivative.
00083          */
00084         smooth(fdata, freqSize, 20);
00085
00086         /*
00087          * Find the derivative of the smoothed spectrum. Note that both this
00088          * filter and the previous are necessary to the algorithm.
00089          */
00090         diff(fdata, freqSize);
00091         smooth(fdata, freqSize, 100);
00092         absolute(fdata, freqSize);
00093
00094         // find the parameters of this specific recording
00095         uint16 offset = 1000;
00096         absolute(&fdata[offset], freqSize - offset);
00097         maximum = max(&fdata[offset], freqSize - offset);
00098         uint32 index = maximum.index + offset;

```

```
00099
00100     DataParams params;
00101     params.freq = index * (float)SAMPLE_FREQ / freqSize / 2;
00102     params.noise = average(&origdata[index + offset],
00103                          freqSize - offset - index);
00104
00105     free(cdata);
00106     free(fdata);
00107
00108     return params;
00109
00110 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.12 `std::map<Side, DataParams> avda::ReadParams ( auto filename )`

Reads the previously computed parameters found in the specified file.

## Parameters

<i>filename</i>	the absolute or relative path to the file containing the patient data to read
-----------------	---

## Returns

the patient parameters read for each side

Definition at line 141 of file [fileio.hpp](#).

```

00141                                     {
00142         std::map<Side, DataParams> myMap;
00143         DataParams leftparams;
00144         DataParams rightparams;
00145
00146         std::ifstream file(filename.c_str());
00147         std::string leftline;
00148         std::string rightline;
00149         std::string leftsearch = "Left";
00150         std::string rightsearch = "Right";
00151         std::string paramstring;
00152         std::string lfreqstr;
00153         std::string lnoiseistr;
00154         std::string rfreqstr;
00155         std::string rnoiseistr;
00156         uint32 lcnt = 0;
00157         uint32 rcnt = 0;
00158         float32 lfreqval;
00159         float32 lnoiseval;
00160         float32 rfreqval;
00161         float32 rnoiseval;
00162
00163         /*
00164          * if statement which uses ifstream function to open patient file
00165          * filename)
00166          */
00167         if(file.is_open()) {
00168             /*
00169              * While statement to find the first Left line and save to
00170              * leftline as string.
00171              */
00172             while (getline(file, leftline)) {
00173                 if(leftline.find(leftsearch, 0) != std::string::npos) {
00174                     break;
00175                 }
00176             }
00177
00178             /*
00179              * While statement to find first right line and save to rightline
00180              * as string.
00181              */
00182             while (getline(file, rightline)) {
00183                 if(rightline.find(rightsearch, 0) != std::string::npos) {
00184                     break;
00185                 }
00186             }
00187
00188             // Code to break leftline and rightline into its parts
00189             std::stringstream lss(leftline);
00190             std::stringstream rss(rightline);
00191
00192             while(getline(lss, paramstring, ',')) {
00193                 lcnt++;
00194
00195                 if(lcnt == 3) {
00196                     lfreqstr = paramstring;
00197                 }
00198
00199                 else if(lcnt == 4) {
00200                     lnoiseistr = paramstring;
00201                 }
00202             }
00203
00204             while(getline(rss, paramstring, ',')) {
00205                 rcnt++;
00206
00207                 if(rcnt == 3) {
00208                     rfreqstr = paramstring;
00209                 }
00210
00211                 else if(rcnt == 4) {
00212                     rnoiseistr = paramstring;
00213

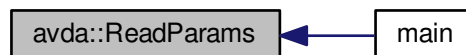
```

```

00214         }
00215     }
00216
00217     /*
00218     * Statement to convert lfreq, lnoise, rfreq, and rnoise from
00219     * strings to floats.
00220     */
00221     lfreqval = atof(lfreqstr.c_str());
00222     lnoiseval = atof(lnoisestr.c_str());
00223     rfreqval = atof(rfreqstr.c_str());
00224     rnoiseval = atof(rnoisestr.c_str());
00225
00226     file.close();
00227 }
00228
00229 else {
00230     throw std::runtime_error("The patient file could not be opened.");
00231 }
00232
00233 leftparams.freq = lfreqval;
00234 leftparams.noise = lnoiseval;
00235 rightparams.freq = rfreqval;
00236 rightparams.noise = rnoiseval;
00237
00238 myMap[Side::Left] = leftparams;
00239 myMap[Side::Right] = rightparams;
00240
00241 return myMap;
00242 }

```

Here is the caller graph for this function:



#### 6.1.3.13 void avda::smooth ( float32 \* data, uint32 size, uint16 order )

Applies an nth-order moving-average filter to a discrete signal.

##### Parameters

<i>data</i>	the array containing the signal to which the filter should be applied
<i>size</i>	the number of elements in the data array
<i>order</i>	the order of the filter

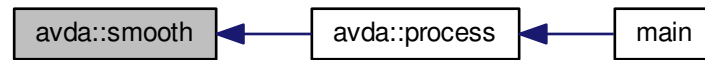
Definition at line 250 of file [sigmath.hpp](#).

```

00250                                     {
00251     float32 coeff = 1 / (float32)order;
00252     float32 temp[size];
00253
00254     for(uint32 i = 0; i < size; i++) {
00255         temp[i] = 0;
00256
00257         for(uint16 j = 0; j < order && j <= i; j++) {
00258             temp[i] += data[i - j];
00259         }
00260
00261         temp[i] *= coeff;
00262     }
00263
00264     for(uint32 i = 0; i < size; i++) {
00265         data[i] = temp[i];
00266     }
00267 }

```

Here is the caller graph for this function:



#### 6.1.3.14 void avda::WriteParams ( std::map< Side, DataParams > myMap, auto filename )

Writes (appends) the passed parameters to the specified file.

##### Parameters

<i>myMap</i>	contains the parameters to be written
--------------	---------------------------------------

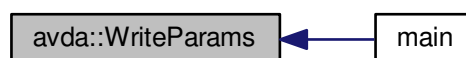
the patient CSV file's filename

Definition at line 251 of file [fileio.hpp](#).

```

00251                                     {
00252     char temp[80];
00253     std::ofstream file(filename.c_str(),
00254         std::ofstream::out | std::ofstream::app);
00255
00256     //Gives pointer measurementtime a data type of time_t.
00257     time_t measurementtime;
00258     time(&measurementtime); //Gets the current time.
00259     strftime(temp, 80, "%c", localtime(&measurementtime));
00260     std::string fTime = std::string(temp);
00261
00262     //if statement to print the Left side parameters to the patient file.
00263     if(file.is_open()) {
00264         file << fTime + "," + "Left" + ","
00265             + std::to_string(myMap[Side::Left].freq)
00266             + ", " + std::to_string(myMap[Side::Left].noise) << std::endl;
00267     }
00268
00269     //if statement to print the Right side parameters to the patient file.
00270     if(file.is_open()) {
00271         file << fTime + "," + "Right" + ","
00272             + std::to_string(myMap[Side::Right].freq)
00273             + ", " + std::to_string(myMap[Side::Right].noise) << std::endl;
00274     }
00275
00276     else {
00277         std::cout << "Patient file can not be opened!" << std::endl;
00278     }
00279
00280     file.close();
00281 }
  
```

Here is the caller graph for this function:



### 6.1.4 Variable Documentation

#### 6.1.4.1 `const std::string avda::CSV_HEADER = "Time,Side,Frequency,Noise Level"`

First line of CSV data file, which declares columns.

Definition at line 25 of file [fileio.hpp](#).

#### 6.1.4.2 `const std::string avda::PATIENT_PATH = "/home/pi/patients/"`

Absolute path to the folder containing the patients' data

Definition at line 30 of file [fileio.hpp](#).



## Chapter 7

# Class Documentation

### 7.1 DataParams Struct Reference

```
#include <definitions.hpp>
```

#### Public Attributes

- `float32 freq` = 0
- `float32 noise` = 0

#### 7.1.1 Detailed Description

Calculated results from processing the audio recordings.

Definition at line 97 of file [definitions.hpp](#).

#### 7.1.2 Member Data Documentation

##### 7.1.2.1 `float32 DataParams::freq` = 0

Definition at line 98 of file [definitions.hpp](#).

##### 7.1.2.2 `float32 DataParams::noise` = 0

Definition at line 99 of file [definitions.hpp](#).

The documentation for this struct was generated from the following file:

- [src/definitions.hpp](#)

### 7.2 Maximum Struct Reference

```
#include <definitions.hpp>
```

#### Public Attributes

- `float32 value` = 0
- `uint32 index` = 0

### 7.2.1 Detailed Description

[Maximum](#) value found in an array and the value's index in that array.

Definition at line [106](#) of file [definitions.hpp](#).

### 7.2.2 Member Data Documentation

#### 7.2.2.1 `uint32 Maximum::index = 0`

Definition at line [108](#) of file [definitions.hpp](#).

#### 7.2.2.2 `float32 Maximum::value = 0`

Definition at line [107](#) of file [definitions.hpp](#).

The documentation for this struct was generated from the following file:

- [src/definitions.hpp](#)

## Chapter 8

# File Documentation

### 8.1 `etc/doxygen.config` File Reference

Contains Doxygen configuration settings.

#### 8.1.1 Detailed Description

Contains Doxygen configuration settings.

##### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [doxygen.config](#).

### 8.2 `doxygen.config`

```
00001 PROJECT_NAME = "The Automatic Vasospasm Detection Application"
00002
00003 INPUT = src/ etc/doxygen.config makefile README.md
00004 OUTPUT_DIRECTORY = doc/
00005
00006 GENERATE_HTML = YES
00007 GENERATE_RTF = YES
00008 GENERATE_LATEX = YES
00009 GENERATE_MAN = YES
00010 GENERATE_XML = NO
00011 GENERATE_DOCBOOK = NO
00012
00013 USE_PDF_LATEX = YES
00014 USE_PDF_HYPERLINKS = YES
00015
00016 RECURSIVE = YES
00017 SOURCE_BROWSER = YES
00018 SOURCE_TOOLTIPS = YES
00019 EXTRACT_ALL = YES
00020 DISABLE_INDEX = NO
00021 GENERATE_TREEVIEW = YES
00022 SEARCHENGINE = YES
00023 SERVER_BASED_SEARCH = NO
00024 USE_MDFILE_AS_MAINPAGE = README.md
00025
00026 LATEX_SOURCE_CODE = YES
00027 STRIP_CODE_COMMENTS = YES
00028 INLINE_SOURCES = YES
00029
00030 HAVE_DOT = YES
00031 CALL_GRAPH = YES
00032 CALLER_GRAPH = YES
```

## 8.3 makefile File Reference

Contains recipes for building the test applications, the main application, and the documentation.

### 8.3.1 Detailed Description

Contains recipes for building the test applications, the main application, and the documentation.

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [makefile](#).

## 8.4 makefile

```
00001 GCC = g++ -g -std=gnu++14
00002
00003 avda:
00004     $(GCC) src/main.cpp -o bin/avda
00005
00006 count:
00007     grep -r "src/" -e "Samuel Andrew Wisner" -l | xargs wc -l
00008
00009 docs:
00010     rm -r doc/
00011     doxygen etc/doxygen.config
00012     cd doc/latex; make pdf;
00013     git reset
00014     git add doc/.
00015     git commit -m "Updated documentation."
00016     git push
00017
00018 fileio-test:
00019     $(GCC) src/fileio_test.cpp -o bin/fileiotest
00020
00021 patient-name-test:
00022     $(GCC) src/patient_name_test.cpp -o bin/patnametest
00023
00024 process-test:
00025     $(GCC) src/process_test.cpp -o bin/proctest
00026
00027 read-params-test:
00028     $(GCC) src/read_params_test.cpp -o bin/rptest
00029
00030 stdin-clear-test:
00031     $(GCC) src/stdin_clear_test.cpp -o bin/cleartest
```

## 8.5 README.md File Reference

Contains the readme text as markdown, which also doubles as the main page.

### 8.5.1 Detailed Description

Contains the readme text as markdown, which also doubles as the main page.

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [README.md](#).

## 8.6 README.md

```
00001 # vasospasm-detector
00002
00003 ## Introduction
00004 The Automatic Vasospasm Detection Application (or Algorithm, depending on the
00005 usage), AVDA, is an application to objectively detect the presence of vasospasms
00006 based on comparisons of parameters extracted from transcranial doppler audio.
00007
00008 ## Setup
00009 AVDA is intended to be compiled on machines running Linux, though it could
00010 likely be adapter for other environments. It must be downloaded from GitHub.com
00011 and compiled locally. To do this, navigate to the directory in which AVDA should
00012 be placed, then execute the following commands
00013
00014     git clone https://github.com/sawbg/avda
00015     cd avda
00016     make
00017
00018 Successfully cloning, compilation, and execution of AVDA requires up-to-date
00019 versions of the following executables:
00020
00021 * git
00022 * make
00023 * gcc (4.9)
00024 * arecord
00025
00026 ## FAQ
00027
00028 * **Why was this project developed?** This project was developed as a course
00029 project by two graduate students at the University of Alabama at Birmingham
00030 School of Engineering, Nicholas Nolan and Andrew Wisner.
00031
00032 * **Is AVDA an active project?** Though it is not planned to develop AVDA
00033 further in the near future, it is hoped that the algorithm discovered and
00034 implemented can be used and built upon by researchers to fully automate the
00035 detection of vasospasms.
00036
00037 * **AVDA is returning unusually low or high parameters. Why might this be?** In
00038 development, this occurred when the mic-in volume was set too high. It is
00039 likely in this scenario that clipping is happening or that the signal (or a
00040 strong enough signal) has not been received.
00041
00042 * **How will AVDA be affected by the machine uprising?** The University
00043 supercomputer, Cheaha, has assured us that AVDA will not be needed after the
00044 uprising occurs.
00045
00046 * **What about more specific questions?** Questions relating to AVDA not
00047 covered in this FAQ may be sent to the AVDA team via awisner94@gmail.com.
00048
00049 ## License
00050
00051     GNU GENERAL PUBLIC LICENSE
00052     Version 3, 29 June 2007
00053
00054     Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
00055     Everyone is permitted to copy and distribute verbatim copies
00056     of this license document, but changing it is not allowed.
00057
00058     Preamble
00059
00060     The GNU General Public License is a free, copyleft license for
00061 software and other kinds of works.
00062
00063     The licenses for most software and other practical works are designed
00064 to take away your freedom to share and change the works. By contrast,
00065 the GNU General Public License is intended to guarantee your freedom to
00066 share and change all versions of a program--to make sure it remains free
00067 software for all its users. We, the Free Software Foundation, use the
00068 GNU General Public License for most of our software; it applies also to
00069 any other work released this way by its authors. You can apply it to
00070 your programs, too.
00071
00072     When we speak of free software, we are referring to freedom, not
00073 price. Our General Public Licenses are designed to make sure that you
00074 have the freedom to distribute copies of free software (and charge for
00075 them if you wish), that you receive source code or can get it if you
00076 want it, that you can change the software or use pieces of it in new
00077 free programs, and that you know you can do these things.
00078
00079     To protect your rights, we need to prevent others from denying you
00080 these rights or asking you to surrender the rights. Therefore, you have
00081 certain responsibilities if you distribute copies of the software, or if
00082 you modify it: responsibilities to respect the freedom of others.
00083
00084     For example, if you distribute copies of such a program, whether
```

00085 gratis or for a fee, you must pass on to the recipients the same  
00086 freedoms that you received. You must make sure that they, too, receive  
00087 or can get the source code. And you must show them these terms so they  
00088 know their rights.  
00089  
00090 Developers that use the GNU GPL protect your rights with two steps:  
00091 (1) assert copyright on the software, and (2) offer you this License  
00092 giving you legal permission to copy, distribute and/or modify it.  
00093  
00094 For the developers' and authors' protection, the GPL clearly explains  
00095 that there is no warranty for this free software. For both users' and  
00096 authors' sake, the GPL requires that modified versions be marked as  
00097 changed, so that their problems will not be attributed erroneously to  
00098 authors of previous versions.  
00099  
00100 Some devices are designed to deny users access to install or run  
00101 modified versions of the software inside them, although the manufacturer  
00102 can do so. This is fundamentally incompatible with the aim of  
00103 protecting users' freedom to change the software. The systematic  
00104 pattern of such abuse occurs in the area of products for individuals to  
00105 use, which is precisely where it is most unacceptable. Therefore, we  
00106 have designed this version of the GPL to prohibit the practice for those  
00107 products. If such problems arise substantially in other domains, we  
00108 stand ready to extend this provision to those domains in future versions  
00109 of the GPL, as needed to protect the freedom of users.  
00110  
00111 Finally, every program is threatened constantly by software patents.  
00112 States should not allow patents to restrict development and use of  
00113 software on general-purpose computers, but in those that do, we wish to  
00114 avoid the special danger that patents applied to a free program could  
00115 make it effectively proprietary. To prevent this, the GPL assures that  
00116 patents cannot be used to render the program non-free.  
00117  
00118 The precise terms and conditions for copying, distribution and  
00119 modification follow.

00120  
00121 TERMS AND CONDITIONS

00122  
00123 0. Definitions.

00124  
00125 "This License" refers to version 3 of the GNU General Public License.  
00126  
00127 "Copyright" also means copyright-like laws that apply to other kinds of  
00128 works, such as semiconductor masks.  
00129  
00130 "The Program" refers to any copyrightable work licensed under this  
00131 License. Each licensee is addressed as "you". "Licensees" and  
00132 "recipients" may be individuals or organizations.  
00133  
00134 To "modify" a work means to copy from or adapt all or part of the work  
00135 in a fashion requiring copyright permission, other than the making of an  
00136 exact copy. The resulting work is called a "modified version" of the  
00137 earlier work or a work "based on" the earlier work.  
00138  
00139 A "covered work" means either the unmodified Program or a work based  
00140 on the Program.  
00141  
00142 To "propagate" a work means to do anything with it that, without  
00143 permission, would make you directly or secondarily liable for  
00144 infringement under applicable copyright law, except executing it on a  
00145 computer or modifying a private copy. Propagation includes copying,  
00146 distribution (with or without modification), making available to the  
00147 public, and in some countries other activities as well.  
00148  
00149 To "convey" a work means any kind of propagation that enables other  
00150 parties to make or receive copies. Mere interaction with a user through  
00151 a computer network, with no transfer of a copy, is not conveying.  
00152  
00153 An interactive user interface displays "Appropriate Legal Notices"  
00154 to the extent that it includes a convenient and prominently visible  
00155 feature that (1) displays an appropriate copyright notice, and (2)  
00156 tells the user that there is no warranty for the work (except to the  
00157 extent that warranties are provided), that licensees may convey the  
00158 work under this License, and how to view a copy of this License. If  
00159 the interface presents a list of user commands or options, such as a  
00160 menu, a prominent item in the list meets this criterion.  
00161  
00162 1. Source Code.

00163  
00164 The "source code" for a work means the preferred form of the work  
00165 for making modifications to it. "Object code" means any non-source  
00166 form of a work.  
00167  
00168 A "Standard Interface" means an interface that either is an official  
00169 standard defined by a recognized standards body, or, in the case of  
00170 interfaces specified for a particular programming language, one that  
00171 is widely used among developers working in that language.

00172  
00173 The "System Libraries" of an executable work include anything, other  
00174 than the work as a whole, that (a) is included in the normal form of  
00175 packaging a Major Component, but which is not part of that Major  
00176 Component, and (b) serves only to enable use of the work with that  
00177 Major Component, or to implement a Standard Interface for which an  
00178 implementation is available to the public in source code form. A  
00179 "Major Component", in this context, means a major essential component  
00180 (kernel, window system, and so on) of the specific operating system  
00181 (if any) on which the executable work runs, or a compiler used to  
00182 produce the work, or an object code interpreter used to run it.  
00183  
00184 The "Corresponding Source" for a work in object code form means all  
00185 the source code needed to generate, install, and (for an executable  
00186 work) run the object code and to modify the work, including scripts to  
00187 control those activities. However, it does not include the work's  
00188 System Libraries, or general-purpose tools or generally available free  
00189 programs which are used unmodified in performing those activities but  
00190 which are not part of the work. For example, Corresponding Source  
00191 includes interface definition files associated with source files for  
00192 the work, and the source code for shared libraries and dynamically  
00193 linked subprograms that the work is specifically designed to require,  
00194 such as by intimate data communication or control flow between those  
00195 subprograms and other parts of the work.  
00196  
00197 The Corresponding Source need not include anything that users  
00198 can regenerate automatically from other parts of the Corresponding  
00199 Source.  
00200  
00201 The Corresponding Source for a work in source code form is that  
00202 same work.  
00203  
00204 2. Basic Permissions.  
00205  
00206 All rights granted under this License are granted for the term of  
00207 copyright on the Program, and are irrevocable provided the stated  
00208 conditions are met. This License explicitly affirms your unlimited  
00209 permission to run the unmodified Program. The output from running a  
00210 covered work is covered by this License only if the output, given its  
00211 content, constitutes a covered work. This License acknowledges your  
00212 rights of fair use or other equivalent, as provided by copyright law.  
00213  
00214 You may make, run and propagate covered works that you do not  
00215 convey, without conditions so long as your license otherwise remains  
00216 in force. You may convey covered works to others for the sole purpose  
00217 of having them make modifications exclusively for you, or provide you  
00218 with facilities for running those works, provided that you comply with  
00219 the terms of this License in conveying all material for which you do  
00220 not control copyright. Those thus making or running the covered works  
00221 for you must do so exclusively on your behalf, under your direction  
00222 and control, on terms that prohibit them from making any copies of  
00223 your copyrighted material outside their relationship with you.  
00224  
00225 Conveying under any other circumstances is permitted solely under  
00226 the conditions stated below. Sublicensing is not allowed; section 10  
00227 makes it unnecessary.  
00228  
00229 3. Protecting Users' Legal Rights From Anti-Circumvention Law.  
00230  
00231 No covered work shall be deemed part of an effective technological  
00232 measure under any applicable law fulfilling obligations under article  
00233 11 of the WIPO copyright treaty adopted on 20 December 1996, or  
00234 similar laws prohibiting or restricting circumvention of such  
00235 measures.  
00236  
00237 When you convey a covered work, you waive any legal power to forbid  
00238 circumvention of technological measures to the extent such circumvention  
00239 is effected by exercising rights under this License with respect to  
00240 the covered work, and you disclaim any intention to limit operation or  
00241 modification of the work as a means of enforcing, against the work's  
00242 users, your or third parties' legal rights to forbid circumvention of  
00243 technological measures.  
00244  
00245 4. Conveying Verbatim Copies.  
00246  
00247 You may convey verbatim copies of the Program's source code as you  
00248 receive it, in any medium, provided that you conspicuously and  
00249 appropriately publish on each copy an appropriate copyright notice;  
00250 keep intact all notices stating that this License and any  
00251 non-permissive terms added in accord with section 7 apply to the code;  
00252 keep intact all notices of the absence of any warranty; and give all  
00253 recipients a copy of this License along with the Program.  
00254  
00255 You may charge any price or no price for each copy that you convey,  
00256 and you may offer support or warranty protection for a fee.  
00257  
00258 5. Conveying Modified Source Versions.

00259  
00260 You may convey a work based on the Program, or the modifications to  
00261 produce it from the Program, in the form of source code under the  
00262 terms of section 4, provided that you also meet all of these conditions:  
00263  
00264 a) The work must carry prominent notices stating that you modified  
00265 it, and giving a relevant date.  
00266  
00267 b) The work must carry prominent notices stating that it is  
00268 released under this License and any conditions added under section  
00269 7. This requirement modifies the requirement in section 4 to  
00270 "keep intact all notices".  
00271  
00272 c) You must license the entire work, as a whole, under this  
00273 License to anyone who comes into possession of a copy. This  
00274 License will therefore apply, along with any applicable section 7  
00275 additional terms, to the whole of the work, and all its parts,  
00276 regardless of how they are packaged. This License gives no  
00277 permission to license the work in any other way, but it does not  
00278 invalidate such permission if you have separately received it.  
00279  
00280 d) If the work has interactive user interfaces, each must display  
00281 Appropriate Legal Notices; however, if the Program has interactive  
00282 interfaces that do not display Appropriate Legal Notices, your  
00283 work need not make them do so.  
00284  
00285 A compilation of a covered work with other separate and independent  
00286 works, which are not by their nature extensions of the covered work,  
00287 and which are not combined with it such as to form a larger program,  
00288 in or on a volume of a storage or distribution medium, is called an  
00289 "aggregate" if the compilation and its resulting copyright are not  
00290 used to limit the access or legal rights of the compilation's users  
00291 beyond what the individual works permit. Inclusion of a covered work  
00292 in an aggregate does not cause this License to apply to the other  
00293 parts of the aggregate.  
00294  
00295 6. Conveying Non-Source Forms.  
00296  
00297 You may convey a covered work in object code form under the terms  
00298 of sections 4 and 5, provided that you also convey the  
00299 machine-readable Corresponding Source under the terms of this License,  
00300 in one of these ways:  
00301  
00302 a) Convey the object code in, or embodied in, a physical product  
00303 (including a physical distribution medium), accompanied by the  
00304 Corresponding Source fixed on a durable physical medium  
00305 customarily used for software interchange.  
00306  
00307 b) Convey the object code in, or embodied in, a physical product  
00308 (including a physical distribution medium), accompanied by a  
00309 written offer, valid for at least three years and valid for as  
00310 long as you offer spare parts or customer support for that product  
00311 model, to give anyone who possesses the object code either (1) a  
00312 copy of the Corresponding Source for all the software in the  
00313 product that is covered by this License, on a durable physical  
00314 medium customarily used for software interchange, for a price no  
00315 more than your reasonable cost of physically performing this  
00316 conveying of source, or (2) access to copy the  
00317 Corresponding Source from a network server at no charge.  
00318  
00319 c) Convey individual copies of the object code with a copy of the  
00320 written offer to provide the Corresponding Source. This  
00321 alternative is allowed only occasionally and noncommercially, and  
00322 only if you received the object code with such an offer, in accord  
00323 with subsection 6b.  
00324  
00325 d) Convey the object code by offering access from a designated  
00326 place (gratis or for a charge), and offer equivalent access to the  
00327 Corresponding Source in the same way through the same place at no  
00328 further charge. You need not require recipients to copy the  
00329 Corresponding Source along with the object code. If the place to  
00330 copy the object code is a network server, the Corresponding Source  
00331 may be on a different server (operated by you or a third party)  
00332 that supports equivalent copying facilities, provided you maintain  
00333 clear directions next to the object code saying where to find the  
00334 Corresponding Source. Regardless of what server hosts the  
00335 Corresponding Source, you remain obligated to ensure that it is  
00336 available for as long as needed to satisfy these requirements.  
00337  
00338 e) Convey the object code using peer-to-peer transmission, provided  
00339 you inform other peers where the object code and Corresponding  
00340 Source of the work are being offered to the general public at no  
00341 charge under subsection 6d.  
00342  
00343 A separable portion of the object code, whose source code is excluded  
00344 from the Corresponding Source as a System Library, need not be  
00345 included in conveying the object code work.



00346  
00347 A "User Product" is either (1) a "consumer product", which means any  
00348 tangible personal property which is normally used for personal, family,  
00349 or household purposes, or (2) anything designed or sold for incorporation  
00350 into a dwelling. In determining whether a product is a consumer product,  
00351 doubtful cases shall be resolved in favor of coverage. For a particular  
00352 product received by a particular user, "normally used" refers to a  
00353 typical or common use of that class of product, regardless of the status  
00354 of the particular user or of the way in which the particular user  
00355 actually uses, or expects or is expected to use, the product. A product  
00356 is a consumer product regardless of whether the product has substantial  
00357 commercial, industrial or non-consumer uses, unless such uses represent  
00358 the only significant mode of use of the product.  
00359  
00360 "Installation Information" for a User Product means any methods,  
00361 procedures, authorization keys, or other information required to install  
00362 and execute modified versions of a covered work in that User Product from  
00363 a modified version of its Corresponding Source. The information must  
00364 suffice to ensure that the continued functioning of the modified object  
00365 code is in no case prevented or interfered with solely because  
00366 modification has been made.  
00367  
00368 If you convey an object code work under this section in, or with, or  
00369 specifically for use in, a User Product, and the conveying occurs as  
00370 part of a transaction in which the right of possession and use of the  
00371 User Product is transferred to the recipient in perpetuity or for a  
00372 fixed term (regardless of how the transaction is characterized), the  
00373 Corresponding Source conveyed under this section must be accompanied  
00374 by the Installation Information. But this requirement does not apply  
00375 if neither you nor any third party retains the ability to install  
00376 modified object code on the User Product (for example, the work has  
00377 been installed in ROM).  
00378  
00379 The requirement to provide Installation Information does not include a  
00380 requirement to continue to provide support service, warranty, or updates  
00381 for a work that has been modified or installed by the recipient, or for  
00382 the User Product in which it has been modified or installed. Access to a  
00383 network may be denied when the modification itself materially and  
00384 adversely affects the operation of the network or violates the rules and  
00385 protocols for communication across the network.  
00386  
00387 Corresponding Source conveyed, and Installation Information provided,  
00388 in accord with this section must be in a format that is publicly  
00389 documented (and with an implementation available to the public in  
00390 source code form), and must require no special password or key for  
00391 unpacking, reading or copying.  
00392  
00393 7. Additional Terms.  
00394  
00395 "Additional permissions" are terms that supplement the terms of this  
00396 License by making exceptions from one or more of its conditions.  
00397 Additional permissions that are applicable to the entire Program shall  
00398 be treated as though they were included in this License, to the extent  
00399 that they are valid under applicable law. If additional permissions  
00400 apply only to part of the Program, that part may be used separately  
00401 under those permissions, but the entire Program remains governed by  
00402 this License without regard to the additional permissions.  
00403  
00404 When you convey a copy of a covered work, you may at your option  
00405 remove any additional permissions from that copy, or from any part of  
00406 it. (Additional permissions may be written to require their own  
00407 removal in certain cases when you modify the work.) You may place  
00408 additional permissions on material, added by you to a covered work,  
00409 for which you have or can give appropriate copyright permission.  
00410  
00411 Notwithstanding any other provision of this License, for material you  
00412 add to a covered work, you may (if authorized by the copyright holders of  
00413 that material) supplement the terms of this License with terms:  
00414  
00415 a) Disclaiming warranty or limiting liability differently from the  
00416 terms of sections 15 and 16 of this License; or  
00417  
00418 b) Requiring preservation of specified reasonable legal notices or  
00419 author attributions in that material or in the Appropriate Legal  
00420 Notices displayed by works containing it; or  
00421  
00422 c) Prohibiting misrepresentation of the origin of that material, or  
00423 requiring that modified versions of such material be marked in  
00424 reasonable ways as different from the original version; or  
00425  
00426 d) Limiting the use for publicity purposes of names of licensors or  
00427 authors of the material; or  
00428  
00429 e) Declining to grant rights under trademark law for use of some  
00430 trade names, trademarks, or service marks; or  
00431  
00432 f) Requiring indemnification of licensors and authors of that

00433 material by anyone who conveys the material (or modified versions of  
00434 it) with contractual assumptions of liability to the recipient, for  
00435 any liability that these contractual assumptions directly impose on  
00436 those licensors and authors.  
00437  
00438 All other non-permissive additional terms are considered "further  
00439 restrictions" within the meaning of section 10. If the Program as you  
00440 received it, or any part of it, contains a notice stating that it is  
00441 governed by this License along with a term that is a further  
00442 restriction, you may remove that term. If a license document contains  
00443 a further restriction but permits relicensing or conveying under this  
00444 License, you may add to a covered work material governed by the terms  
00445 of that license document, provided that the further restriction does  
00446 not survive such relicensing or conveying.  
00447  
00448 If you add terms to a covered work in accord with this section, you  
00449 must place, in the relevant source files, a statement of the  
00450 additional terms that apply to those files, or a notice indicating  
00451 where to find the applicable terms.  
00452  
00453 Additional terms, permissive or non-permissive, may be stated in the  
00454 form of a separately written license, or stated as exceptions;  
00455 the above requirements apply either way.  
00456  
00457 8. Termination.  
00458  
00459 You may not propagate or modify a covered work except as expressly  
00460 provided under this License. Any attempt otherwise to propagate or  
00461 modify it is void, and will automatically terminate your rights under  
00462 this License (including any patent licenses granted under the third  
00463 paragraph of section 11).  
00464  
00465 However, if you cease all violation of this License, then your  
00466 license from a particular copyright holder is reinstated (a)  
00467 provisionally, unless and until the copyright holder explicitly and  
00468 finally terminates your license, and (b) permanently, if the copyright  
00469 holder fails to notify you of the violation by some reasonable means  
00470 prior to 60 days after the cessation.  
00471  
00472 Moreover, your license from a particular copyright holder is  
00473 reinstated permanently if the copyright holder notifies you of the  
00474 violation by some reasonable means, this is the first time you have  
00475 received notice of violation of this License (for any work) from that  
00476 copyright holder, and you cure the violation prior to 30 days after  
00477 your receipt of the notice.  
00478  
00479 Termination of your rights under this section does not terminate the  
00480 licenses of parties who have received copies or rights from you under  
00481 this License. If your rights have been terminated and not permanently  
00482 reinstated, you do not qualify to receive new licenses for the same  
00483 material under section 10.  
00484  
00485 9. Acceptance Not Required for Having Copies.  
00486  
00487 You are not required to accept this License in order to receive or  
00488 run a copy of the Program. Ancillary propagation of a covered work  
00489 occurring solely as a consequence of using peer-to-peer transmission  
00490 to receive a copy likewise does not require acceptance. However,  
00491 nothing other than this License grants you permission to propagate or  
00492 modify any covered work. These actions infringe copyright if you do  
00493 not accept this License. Therefore, by modifying or propagating a  
00494 covered work, you indicate your acceptance of this License to do so.  
00495  
00496 10. Automatic Licensing of Downstream Recipients.  
00497  
00498 Each time you convey a covered work, the recipient automatically  
00499 receives a license from the original licensors, to run, modify and  
00500 propagate that work, subject to this License. You are not responsible  
00501 for enforcing compliance by third parties with this License.  
00502  
00503 An "entity transaction" is a transaction transferring control of an  
00504 organization, or substantially all assets of one, or subdividing an  
00505 organization, or merging organizations. If propagation of a covered  
00506 work results from an entity transaction, each party to that  
00507 transaction who receives a copy of the work also receives whatever  
00508 licenses to the work the party's predecessor in interest had or could  
00509 give under the previous paragraph, plus a right to possession of the  
00510 Corresponding Source of the work from the predecessor in interest, if  
00511 the predecessor has it or can get it with reasonable efforts.  
00512  
00513 You may not impose any further restrictions on the exercise of the  
00514 rights granted or affirmed under this License. For example, you may  
00515 not impose a license fee, royalty, or other charge for exercise of  
00516 rights granted under this License, and you may not initiate litigation  
00517 (including a cross-claim or counterclaim in a lawsuit) alleging that  
00518 any patent claim is infringed by making, using, selling, offering for  
00519 sale, or importing the Program or any portion of it.

00520  
00521 11. Patents.  
00522  
00523 A "contributor" is a copyright holder who authorizes use under this  
00524 License of the Program or a work on which the Program is based. The  
00525 work thus licensed is called the contributor's "contributor version".  
00526  
00527 A contributor's "essential patent claims" are all patent claims  
00528 owned or controlled by the contributor, whether already acquired or  
00529 hereafter acquired, that would be infringed by some manner, permitted  
00530 by this License, of making, using, or selling its contributor version,  
00531 but do not include claims that would be infringed only as a  
00532 consequence of further modification of the contributor version. For  
00533 purposes of this definition, "control" includes the right to grant  
00534 patent sublicenses in a manner consistent with the requirements of  
00535 this License.  
00536  
00537 Each contributor grants you a non-exclusive, worldwide, royalty-free  
00538 patent license under the contributor's essential patent claims, to  
00539 make, use, sell, offer for sale, import and otherwise run, modify and  
00540 propagate the contents of its contributor version.  
00541  
00542 In the following three paragraphs, a "patent license" is any express  
00543 agreement or commitment, however denominated, not to enforce a patent  
00544 (such as an express permission to practice a patent or covenant not to  
00545 sue for patent infringement). To "grant" such a patent license to a  
00546 party means to make such an agreement or commitment not to enforce a  
00547 patent against the party.  
00548  
00549 If you convey a covered work, knowingly relying on a patent license,  
00550 and the Corresponding Source of the work is not available for anyone  
00551 to copy, free of charge and under the terms of this License, through a  
00552 publicly available network server or other readily accessible means,  
00553 then you must either (1) cause the Corresponding Source to be so  
00554 available, or (2) arrange to deprive yourself of the benefit of the  
00555 patent license for this particular work, or (3) arrange, in a manner  
00556 consistent with the requirements of this License, to extend the patent  
00557 license to downstream recipients. "Knowingly relying" means you have  
00558 actual knowledge that, but for the patent license, your conveying the  
00559 covered work in a country, or your recipient's use of the covered work  
00560 in a country, would infringe one or more identifiable patents in that  
00561 country that you have reason to believe are valid.  
00562  
00563 If, pursuant to or in connection with a single transaction or  
00564 arrangement, you convey, or propagate by procuring conveyance of, a  
00565 covered work, and grant a patent license to some of the parties  
00566 receiving the covered work authorizing them to use, propagate, modify  
00567 or convey a specific copy of the covered work, then the patent license  
00568 you grant is automatically extended to all recipients of the covered  
00569 work and works based on it.  
00570  
00571 A patent license is "discriminatory" if it does not include within  
00572 the scope of its coverage, prohibits the exercise of, or is  
00573 conditioned on the non-exercise of one or more of the rights that are  
00574 specifically granted under this License. You may not convey a covered  
00575 work if you are a party to an arrangement with a third party that is  
00576 in the business of distributing software, under which you make payment  
00577 to the third party based on the extent of your activity of conveying  
00578 the work, and under which the third party grants, to any of the  
00579 parties who would receive the covered work from you, a discriminatory  
00580 patent license (a) in connection with copies of the covered work  
00581 conveyed by you (or copies made from those copies), or (b) primarily  
00582 for and in connection with specific products or compilations that  
00583 contain the covered work, unless you entered into that arrangement,  
00584 or that patent license was granted, prior to 28 March 2007.  
00585  
00586 Nothing in this License shall be construed as excluding or limiting  
00587 any implied license or other defenses to infringement that may  
00588 otherwise be available to you under applicable patent law.  
00589  
00590 12. No Surrender of Others' Freedom.  
00591  
00592 If conditions are imposed on you (whether by court order, agreement or  
00593 otherwise) that contradict the conditions of this License, they do not  
00594 excuse you from the conditions of this License. If you cannot convey a  
00595 covered work so as to satisfy simultaneously your obligations under this  
00596 License and any other pertinent obligations, then as a consequence you may  
00597 not convey it at all. For example, if you agree to terms that obligate you  
00598 to collect a royalty for further conveying from those to whom you convey  
00599 the Program, the only way you could satisfy both those terms and this  
00600 License would be to refrain entirely from conveying the Program.  
00601  
00602 13. Use with the GNU Affero General Public License.  
00603  
00604 Notwithstanding any other provision of this License, you have  
00605 permission to link or combine any covered work with a work licensed  
00606 under version 3 of the GNU Affero General Public License into a single

00607 combined work, and to convey the resulting work. The terms of this  
00608 License will continue to apply to the part which is the covered work,  
00609 but the special requirements of the GNU Affero General Public License,  
00610 section 13, concerning interaction through a network will apply to the  
00611 combination as such.

00612

00613 14. Revised Versions of this License.

00614

00615 The Free Software Foundation may publish revised and/or new versions of  
00616 the GNU General Public License from time to time. Such new versions will  
00617 be similar in spirit to the present version, but may differ in detail to  
00618 address new problems or concerns.

00619

00620 Each version is given a distinguishing version number. If the  
00621 Program specifies that a certain numbered version of the GNU General  
00622 Public License "or any later version" applies to it, you have the  
00623 option of following the terms and conditions either of that numbered  
00624 version or of any later version published by the Free Software  
00625 Foundation. If the Program does not specify a version number of the  
00626 GNU General Public License, you may choose any version ever published  
00627 by the Free Software Foundation.

00628

00629 If the Program specifies that a proxy can decide which future  
00630 versions of the GNU General Public License can be used, that proxy's  
00631 public statement of acceptance of a version permanently authorizes you  
00632 to choose that version for the Program.

00633

00634 Later license versions may give you additional or different  
00635 permissions. However, no additional obligations are imposed on any  
00636 author or copyright holder as a result of your choosing to follow a  
00637 later version.

00638

00639 15. Disclaimer of Warranty.

00640

00641 THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY  
00642 APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT  
00643 HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY  
00644 OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,  
00645 THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR  
00646 PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM  
00647 IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF  
00648 ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

00649

00650 16. Limitation of Liability.

00651

00652 IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING  
00653 WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS  
00654 THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY  
00655 GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE  
00656 USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF  
00657 DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD  
00658 PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),  
00659 EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF  
00660 SUCH DAMAGES.

00661

00662 17. Interpretation of Sections 15 and 16.

00663

00664 If the disclaimer of warranty and limitation of liability provided  
00665 above cannot be given local legal effect according to their terms,  
00666 reviewing courts shall apply local law that most closely approximates  
00667 an absolute waiver of all civil liability in connection with the  
00668 Program, unless a warranty or assumption of liability accompanies a  
00669 copy of the Program in return for a fee.

00670

00671 END OF TERMS AND CONDITIONS

00672

00673 How to Apply These Terms to Your New Programs

00674

00675 If you develop a new program, and you want it to be of the greatest  
00676 possible use to the public, the best way to achieve this is to make it  
00677 free software which everyone can redistribute and change under these terms.

00678

00679 To do so, attach the following notices to the program. It is safest  
00680 to attach them to the start of each source file to most effectively  
00681 state the exclusion of warranty; and each file should have at least  
00682 the "copyright" line and a pointer to where the full notice is found.

00683

00684 {one line to give the program's name and a brief idea of what it does.}  
00685 Copyright (C) {year} {name of author}

00686

00687 This program is free software: you can redistribute it and/or modify  
00688 it under the terms of the GNU General Public License as published by  
00689 the Free Software Foundation, either version 3 of the License, or  
00690 (at your option) any later version.

00691

00692 This program is distributed in the hope that it will be useful,  
00693 but WITHOUT ANY WARRANTY; without even the implied warranty of

```

00694     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00695     GNU General Public License for more details.
00696
00697     You should have received a copy of the GNU General Public License
00698     along with this program.  If not, see <http://www.gnu.org/licenses/>.
00699
00700 Also add information on how to contact you by electronic and paper mail.
00701
00702 If the program does terminal interaction, make it output a short
00703 notice like this when it starts in an interactive mode:
00704
00705     {project} Copyright (C) {year} {fullname}
00706     This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
00707     This is free software, and you are welcome to redistribute it
00708     under certain conditions; type `show c' for details.
00709
00710 The hypothetical commands `show w' and `show c' should show the appropriate
00711 parts of the General Public License.  Of course, your program's commands
00712 might be different; for a GUI interface, you would use an "about box".
00713
00714 You should also get your employer (if you work as a programmer) or school,
00715 if any, to sign a "copyright disclaimer" for the program, if necessary.
00716 For more information on this, and how to apply and follow the GNU GPL, see
00717 <http://www.gnu.org/licenses/>.
00718
00719 The GNU General Public License does not permit incorporating your program
00720 into proprietary programs.  If your program is a subroutine library, you
00721 may consider it more useful to permit linking proprietary applications with
00722 the library.  If this is what you want to do, use the GNU Lesser General
00723 Public License instead of this License.  But first, please read
00724 <http://www.gnu.org/philosophy/why-not-lgpl.html>.
00725

```

## 8.7 src/definitions.hpp File Reference

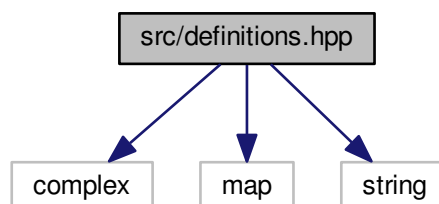
Contains declarations of system-independant (universal size) integers and float types, shortened type names for some commonly used types, and enumerations.

```

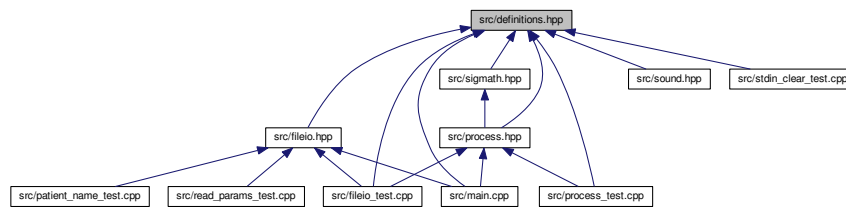
#include <complex>
#include <map>
#include <string>

```

Include dependency graph for definitions.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- struct [DataParams](#)
- struct [Maximum](#)

## Namespaces

- [avda](#)

## Macros

- #define [ENUM](#) signed char

## Typedefs

- typedef unsigned char [byte](#)
- typedef unsigned char [uint8](#)
- typedef signed char [sint8](#)
- typedef unsigned short [uint16](#)
- typedef signed short [sint16](#)
- typedef unsigned int [uint32](#)
- typedef signed int [sint32](#)
- typedef unsigned long long [uint64](#)
- typedef signed long long [sint64](#)
- typedef float [float32](#)
- typedef double [float64](#)
- typedef std::complex< [float32](#) > [cfloat32](#)

## Enumerations

- enum [avda::Side](#) { [avda::Side::Left](#), [avda::Side::Right](#) }

## Variables

- const [uint16](#) [DET\\_THRESH](#) = 5000
- const [uint8](#) [DURATION](#) = 6
- const [sint8](#) [ERROR](#) = -1
- const [uint16](#) [MAX\\_DROP\\_FREQ](#) = 7000
- const [uint8](#) [REC\\_COUNT](#) = 6

- const `uint32 SAMPLE_COUNT` = 131072
- const `uint16 SAMPLE_FREQ` = 24000
- const `std::string TEMP_FILE` = ".temp"
- const `uint32 BUFFER_SIZE` = `SAMPLE_COUNT` \* `sizeof(float32)`

### 8.7.1 Detailed Description

Contains declarations of system-independant (universal size) integers and float types, shortened type names for some commonly used types, and enumerations.

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [definitions.hpp](#).

### 8.7.2 Macro Definition Documentation

#### 8.7.2.1 #define ENUM signed char

Definition at line 16 of file [definitions.hpp](#).

### 8.7.3 Typedef Documentation

#### 8.7.3.1 typedef unsigned char byte

Definition at line 20 of file [definitions.hpp](#).

#### 8.7.3.2 typedef std::complex<float32> cfloat32

Complex float32's.

Definition at line 92 of file [definitions.hpp](#).

#### 8.7.3.3 typedef float float32

Definition at line 33 of file [definitions.hpp](#).

#### 8.7.3.4 typedef double float64

Definition at line 34 of file [definitions.hpp](#).

#### 8.7.3.5 typedef signed short sint16

Definition at line 25 of file [definitions.hpp](#).

#### 8.7.3.6 typedef signed int sint32

Definition at line 28 of file [definitions.hpp](#).

#### 8.7.3.7 typedef signed long long sint64

Definition at line 31 of file [definitions.hpp](#).

#### 8.7.3.8 typedef signed char sint8

Definition at line 22 of file [definitions.hpp](#).

#### 8.7.3.9 typedef unsigned short uint16

Definition at line 24 of file [definitions.hpp](#).

#### 8.7.3.10 typedef unsigned int uint32

Definition at line 27 of file [definitions.hpp](#).

#### 8.7.3.11 typedef unsigned long long uint64

Definition at line 30 of file [definitions.hpp](#).

#### 8.7.3.12 typedef unsigned char uint8

Definition at line 21 of file [definitions.hpp](#).

### 8.7.4 Variable Documentation

#### 8.7.4.1 const uint32 BUFFER\_SIZE = SAMPLE\_COUNT \* sizeof(float32)

Size of the sample buffer.

Definition at line 84 of file [definitions.hpp](#).

#### 8.7.4.2 const uint16 DET\_THRESH = 5000

Threshold for the differential-parameters product to be considered indicative of a vasospasm.

Definition at line 43 of file [definitions.hpp](#).

#### 8.7.4.3 const uint8 DURATION = 6

Duration of recording in seconds.

Definition at line 48 of file [definitions.hpp](#).

#### 8.7.4.4 const sint8 ERROR = -1

Error integer returned when the program must exit with an error.

Definition at line 53 of file [definitions.hpp](#).



**8.7.4.5** `const uint16 MAX_DROP_FREQ = 7000`

Maximum drop-off frequency considered valid.

Definition at line 58 of file [definitions.hpp](#).

**8.7.4.6** `const uint8 REC_COUNT = 6`

Number of recordings (both left and right) to make.

Definition at line 63 of file [definitions.hpp](#).

**8.7.4.7** `const uint32 SAMPLE_COUNT = 131072`

Number of samples to use in processing the recordings. Must be a power of two.  $SAMPLE\_COUNT / SAMPLE\_FREQ < DURATION$  must be true.

Definition at line 69 of file [definitions.hpp](#).

**8.7.4.8** `const uint16 SAMPLE_FREQ = 24000`

Recording sampling rate in Hz (NOT kHz).

Definition at line 74 of file [definitions.hpp](#).

**8.7.4.9** `const std::string TEMP_FILE = ".temp"`

Filename of the temporary recording file.

Definition at line 79 of file [definitions.hpp](#).

**8.8** `definitions.hpp`

```

00001
00009 #ifndef definitions_H
00010 #define definitions_H
00011
00012 #include <complex>
00013 #include <map>
00014 #include <string>
00015
00016 #define ENUM signed char
00017
00018 // Type definitions
00019
00020 typedef unsigned char byte;
00021 typedef unsigned char uint8;
00022 typedef signed char sint8;
00023
00024 typedef unsigned short uint16;
00025 typedef signed short sint16;
00026
00027 typedef unsigned int uint32;
00028 typedef signed int sint32;
00029
00030 typedef unsigned long long uint64;
00031 typedef signed long long sint64;
00032
00033 typedef float float32;
00034 typedef double float64;
00035
00036
00037 // Constants
00038
00043 const uint16 DET_THRESH = 5000;
00044
00048 const uint8 DURATION = 6;
00049

```

```

00053 const sint8 ERROR = -1;
00054
00058 const uint16 MAX_DROP_FREQ = 7000;
00059
00063 const uint8 REC_COUNT = 6;
00064
00069 const uint32 SAMPLE_COUNT = 131072; //262144;
00070
00074 const uint16 SAMPLE_FREQ = 24000;
00075
00079 const std::string TEMP_FILE = ".temp";
00080
00084 const uint32 BUFFER_SIZE = SAMPLE_COUNT * sizeof(
    float32);
00085
00086
00087 // Objective/structural type definitions
00088
00092 typedef std::complex<float32> cfloat32;
00093
00097 typedef struct {
00098     float32 freq = 0;
00099     float32 noise = 0;
00100 } DataParams;
00101
00106 typedef struct {
00107     float32 value = 0;
00108     uint32 index = 0;
00109 } Maximum;
00110
00111
00112 // Enumerations
00113
00117 namespace avda {
00121     enum class Side { Left, Right };
00122 }
00123
00124
00125 // Doxygen documentation for other files.
00126
00147 #endif

```

## 8.9 src/fileio.hpp File Reference

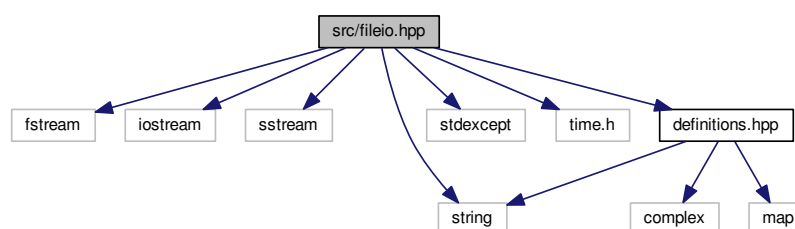
contains functions related to the file I/O use in this program

```

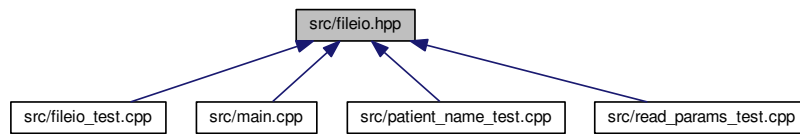
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <stdexcept>
#include <time.h>
#include "definitions.hpp"

```

Include dependency graph for fileio.hpp:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [avda](#)

## Functions

- `std::string avda::PatientName ()`
- `std::map< Side, DataParams > avda::ReadParams (auto filename)`
- `void avda::WriteParams (std::map< Side, DataParams > myMap, auto filename)`

## Variables

- `const std::string avda::CSV\_HEADER = "Time,Side,Frequency,Noise Level"`
- `const std::string avda::PATIENT\_PATH = "/home/pi/patients/"`

### 8.9.1 Detailed Description

contains functions related to the file I/O use in this program

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)  
 Nicholas K. Nolan

**Bug** file is overly complicated and much more bug-prone

Definition in file [fileio.hpp](#).

## 8.10 fileio.hpp

```

00001
00009 #ifndef fileio_H
00010 #define fileio_H
00011
00012 #include <fstream>
00013 #include <iostream>
00014 #include <sstream>
00015 #include <string>
00016 #include <stdexcept>
00017 #include <time.h>
00018
00019 #include "definitions.hpp"
00020
00021 namespace avda {
00025     const std::string CSV_HEADER = "Time,Side,Frequency,Noise Level";
00026
  
```

```

00030     const std::string PATIENT_PATH = "/home/pi/patients/";
00031
00043     std::string PatientName() {
00044         std::string fname = "";
00045         std::string mname = "";
00046         std::string lname = "";
00047         std::string patfil = "";
00048         std::string patientname = "";
00049         uint32 track1 = 0;
00050         uint32 track2 = 0;
00051         uint32 track3 = 0;
00052
00053         do {
00054             std::cout << "Please enter the patients name." << std::endl;
00055             std::cout << "First name: ";
00056             std::cin >> fname;
00057             std::cout << "Middle name: ";
00058             std::cin >> mname;
00059             std::cout << "Last name: ";
00060             std::cin >> lname;
00061
00062             // creates new std::string with path to patient file
00063             patientname = PATIENT_PATH + lname + ", " + fname
00064                 + " " + mname + ".csv";
00065
00066             // prints out patientname. shows user the path to the patient file
00067             //std::cout << patientname << std::endl << std::endl;
00068             std::ifstream file(patientname.c_str());
00069
00070             if (file.good()) {
00071                 track1 = 1;
00072             }
00073
00074             /*
00075              * Compares patientname to existing files and lets user know
00076              * if the file does not exist.
00077              */
00078             else if (!file.good()) {
00079                 /*
00080                  * Do while statement to continue asking user about the file
00081                  * if their input is not acceptable
00082                  */
00083                 do {
00084                     std::cout << "Patient file does not exist, would you like "
00085                         "to create file or re-enter their name?" << std::endl;
00086                     std::cout << " *Type 'create' and press enter key "
00087                         "to create the patient file." << std::endl;
00088                     std::cout << " *Type 'reenter' and press enter key "
00089                         "to re-enter the patients name." << std::endl;
00090                     std::cout << std::endl;
00091                     std::cin >> patfil;
00092
00093                     /*
00094                      * patfil equals create, track1 and 2 will increase
00095                      * escaping both do while loops
00096                      */
00097                     if(patfil == "create") {
00098                         std::ofstream createfile(patientname.c_str());
00099                         track1 = 1;
00100                         track2 = 1;
00101                         track3 = 1;
00102                         createfile << CSV_HEADER << std::endl;
00103                         createfile.flush();
00104                         createfile.close();
00105                     }
00106
00107                     /*
00108                      *patfil equals reenter, track1 will remain zero allowing
00109                      *user to reenter the patient name.
00110                      */
00111                     else if(patfil == "reenter") {
00112                         track1 = 0;
00113                         track2 = 1;
00114                     }
00115
00116                     /*
00117                      *The users input was neither create or reenter. User
00118                      *must enter patient name again.
00119                      */
00120                     else {
00121                         std::cout << std::endl;
00122                         std::cout << "Your input is not acceptable." << std::endl;
00123                         std::cout << std::endl;
00124                     }
00125                 }while(track2 == 0);
00126             }
00127         } while (track1 == 0);

```

```

00128
00129     return patientname; //returns the path to the patient file
00130 }
00131
00141 std::map<Side, DataParams> ReadParams(auto filename) {
00142     std::map<Side, DataParams> myMap;
00143     DataParams leftparams;
00144     DataParams rightparams;
00145
00146     std::ifstream file(filename.c_str());
00147     std::string leftline;
00148     std::string rightline;
00149     std::string leftsearch = "Left";
00150     std::string rightsearch = "Right";
00151     std::string paramstring;
00152     std::string lfreqstr;
00153     std::string lnoisestr;
00154     std::string rfreqstr;
00155     std::string rnoisestr;
00156     uint32 lcnt = 0;
00157     uint32 rcnt = 0;
00158     float32 lfreqval;
00159     float32 lnoiseval;
00160     float32 rfreqval;
00161     float32 rnoiseval;
00162
00163     /*
00164     * if statement which uses ifstream function to open patient file
00165     * filename)
00166     */
00167     if(file.is_open()) {
00168         /*
00169         * While statement to find the first Left line and save to
00170         * leftline as string.
00171         */
00172         while (getline(file, leftline)) {
00173             if(leftline.find(leftsearch, 0) != std::string::npos) {
00174                 break;
00175             }
00176         }
00177     }
00178
00179     /*
00180     * While statement to find first right line and save to rightline
00181     * as string.
00182     */
00183     while (getline(file, rightline)) {
00184         if(rightline.find(rightsearch, 0) != std::string::npos) {
00185             break;
00186         }
00187     }
00188
00189     // Code to break leftline and rightline into its parts
00190     std::stringstream lss(leftline);
00191     std::stringstream rss(rightline);
00192
00193     while(getline(lss, paramstring, ',')) {
00194         lcnt++;
00195
00196         if(lcnt == 3) {
00197             lfreqstr = paramstring;
00198         }
00199
00200         else if(lcnt == 4) {
00201             lnoisestr = paramstring;
00202         }
00203     }
00204
00205     while(getline(rss, paramstring, ',')) {
00206         rcnt++;
00207
00208         if(rcnt == 3) {
00209             rfreqstr = paramstring;
00210         }
00211
00212         else if(rcnt == 4) {
00213             rnoisestr = paramstring;
00214         }
00215     }
00216
00217     /*
00218     * Statement to convert lfreq, lnoise, rfreq, and rnoise from
00219     * strings to floats.
00220     */
00221     lfreqval = atof(lfreqstr.c_str());
00222     lnoiseval = atof(lnoisestr.c_str());
00223     rfreqval = atof(rfreqstr.c_str());

```

```

00224         rnoiseval = atof(rnoisestr.c_str());
00225
00226         file.close();
00227     }
00228
00229     else {
00230         throw std::runtime_error("The patient file could not be opened.");
00231     }
00232
00233     leftparams.freq = lfreqval;
00234     leftparams.noise = lnoiseval;
00235     rightparams.freq = rfreqval;
00236     rightparams.noise = rnoiseval;
00237
00238     myMap[Side::Left] = leftparams;
00239     myMap[Side::Right] = rightparams;
00240
00241     return myMap;
00242 }
00243
00251 void WriteParams(std::map<Side, DataParams> myMap, auto filename) {
00252     char temp[80];
00253     std::ofstream file(filename.c_str(),
00254         std::ofstream::out | std::ofstream::app);
00255
00256     //Gives pointer measurementtime a data type of time_t.
00257     time_t measurementtime;
00258     time(&measurementtime); //Gets the current time.
00259     strftime(temp, 80, "%c", localtime(&measurementtime));
00260     std::string fTime = std::string(temp);
00261
00262     //if statement to print the Left side parameters to the patient file.
00263     if(file.is_open()) {
00264         file << fTime + "," + "Left" + ","
00265             + std::to_string(myMap[Side::Left].freq)
00266             + ", " + std::to_string(myMap[Side::Left].noise) << std::endl;
00267     }
00268
00269     //if statement to print the Right side parameters to the patient file.
00270     if(file.is_open()) {
00271         file << fTime + "," + "Right" + ","
00272             + std::to_string(myMap[Side::Right].freq)
00273             + ", " + std::to_string(myMap[Side::Right].noise) << std::endl;
00274     }
00275
00276     else {
00277         std::cout << "Patient file can not be opened!" << std::endl;
00278     }
00279
00280     file.close();
00281 }
00282 }
00283
00284 #endif

```

## 8.11 src/fileio\_test.cpp File Reference

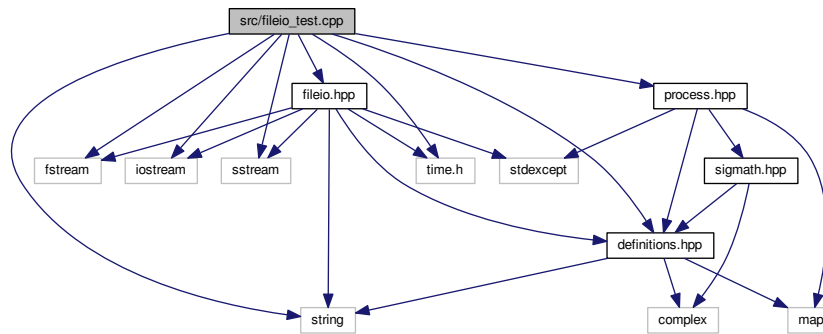
Contains program that tests the functions in [fileio.hpp](#).

```

#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <time.h>
#include "definitions.hpp"
#include "fileio.hpp"
#include "process.hpp"

```

Include dependency graph for fileio\_test.cpp:



## Functions

- int [main](#) ()

### 8.11.1 Detailed Description

Contains program that tests the functions in [fileio.hpp](#).

#### Author

Samuel Andrew Wisner  
Nicholas K. Nolan

Definition in file [fileio\\_test.cpp](#).

### 8.11.2 Function Documentation

#### 8.11.2.1 int main ( )

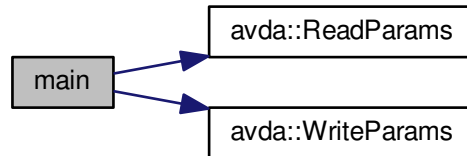
Tests the functions in [fileio.hpp](#).

Definition at line [24](#) of file [fileio\\_test.cpp](#).

```

00024         {
00025             string path = PATIENT_PATH + "wizmack, sammy andy.csv";
00026             map<Side, DataParams> laMap = ReadParams(path);
00027             cout << laMap[Side::Right].freq << endl;
00028             cout << laMap[Side::Right].noise << endl;
00029
00030             WriteParams(laMap, path);
00031         }
  
```

Here is the call graph for this function:



## 8.12 fileio\_test.cpp

```

00001
00008 #include <fstream>
00009 #include <iostream>
00010 #include <sstream>
00011 #include <string>
00012 #include <time.h>
00013
00014 #include "definitions.hpp"
00015 #include "fileio.hpp"
00016 #include "process.hpp"
00017
00018 using namespace std;
00019 using namespace avda;
00020
00024 int main() {
00025     string path = PATIENT_PATH + "wizmack, sammy andy.csv";
00026     map<Side, DataParams> laMap = ReadParams(path);
00027     cout << laMap[Side::Right].freq << endl;
00028     cout << laMap[Side::Right].noise << endl;
00029
00030     WriteParams(laMap, path);
00031 }
  
```

## 8.13 src/main.cpp File Reference

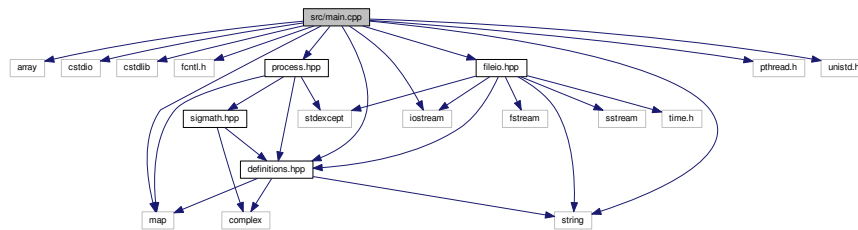
Contains the main program.

```

#include <array>
#include <cstdio>
#include <cstdlib>
#include <fcntl.h>
#include <iostream>
#include <map>
#include <pthread.h>
#include <string>
#include <unistd.h>
#include "definitions.hpp"
#include "fileio.hpp"
#include "process.hpp"
  
```



Include dependency graph for main.cpp:



## Functions

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

### 8.13.1 Detailed Description

Contains the main program.

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)  
Nicholas K. Nolan

**Bug** extra newline character inserted into stdin buffer after `PatientName()` is run

Definition in file `main.cpp`.

### 8.13.2 Function Documentation

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

The main program for this project. It will detect avdaspasms over a period of days.

Definition at line 31 of file `main.cpp`.

```

00031                                     {
00032     // Recorded audio buffer
00033     float32* buffer = (float32*)std::malloc(BUFFER_SIZE);
00034     bool cont = true; // whether to continue in the recording loop
00035     DataParams params[REC_COUNT]; // holds DataParam's from recordings
00036     string filename = PatientName(); // generate name for patient's file
00037     map<Side, DataParams> results; // parameters by side
00038
00039     // arecord command
00040     const string recCommand = string("arecord -t raw -d ")
00041     + to_string(DURATION) + string(" -D plughw:1,0 -f FLOAT -q -r ")
00042     + to_string(SAMPLE_FREQ) + string(" ") + TEMP_FILE;
00043
00044     // Recording
00045     while(cont) {
00046         for(uint8 i = 0; i < REC_COUNT; i++) {
00047             // prompt
00048             cout << "Press [ ENTER ] to begin analysis for the "
00049             << (i < REC_COUNT / 2 ? "left" : "right") << " side, depth #"
00050             << ((i >= REC_COUNT / 2) ? (i - REC_COUNT / 2) : i) + 1)
00051             << " ";
00052             getchar(); // wait for ENTER to be pressed
00053             cout << "Analyzing..." << endl;
00054
00055             system(recCommand.c_str());
00056             usleep(DURATION*1000000 + 1500000); // sleep DURATION + 1.5 seconds

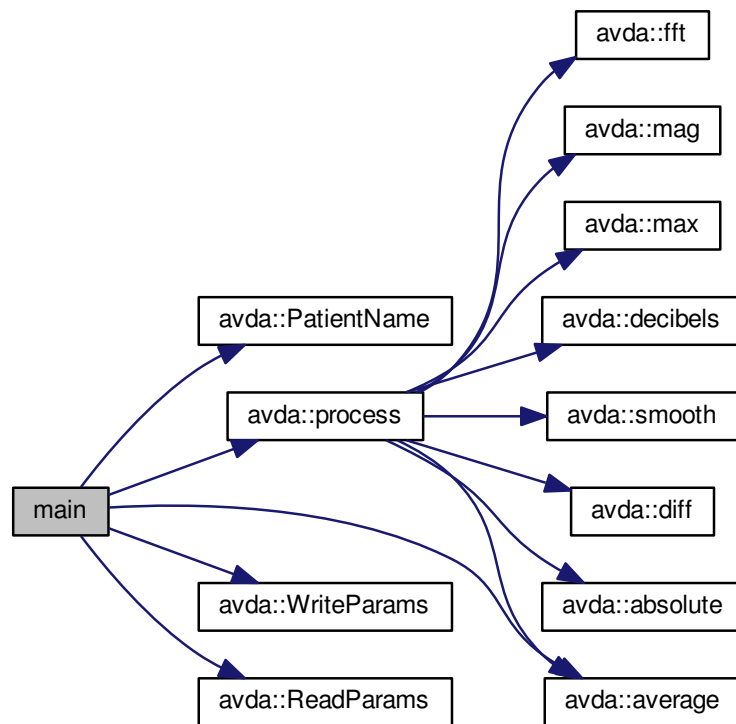
```

```

00057
00058     int file = open(TEMP_FILE.c_str(), O_RDONLY); // open temp file
00059     int retRead = read(file, buffer, BUFFER_SIZE); // copy to buffer
00060     close(file); // close temp file
00061     remove(TEMP_FILE.c_str()); // delete temp file
00062
00063     // if something goes wrong reading the temp file, program exits
00064     if(file < 0 || retRead < BUFFER_SIZE) {
00065         cerr << "An error occurred reading the doppler audio! "
00066              "The program will now exit." << endl;
00067         return ERROR;
00068     }
00069
00070     // process and store parameters
00071     params[i] = process(buffer, SAMPLE_COUNT,
SAMPLE_FREQ);
00072     cout << "The analysis is complete." << endl << endl;
00073 }
00074
00075 // calculate averaged parameters
00076 results[Side::Left] = average(params, REC_COUNT / 2);
00077 results[Side::Right] = average(&params[REC_COUNT / 2], REC_COUNT / 2);
00078
00079 cout << "Analysis is complete." << endl << endl;
00080
00081 // print averaged side analysis
00082 for(int i = 0; i < 2; i++) {
00083     Side side = (Side)i;
00084     cout << (side == Side::Left ? "[LEFT]" : "[RIGHT]") << endl;
00085     cout << "Drop-off frequency: " << (uint16)(results[side].freq + 0.5)
00086          << " Hz" << endl;
00087     cout << "Average relative noiseband power: "
00088          << (sint16)(results[side].noise - 0.5) << " dB" << endl << endl;
00089 }
00090
00091 cont = results[Side::Left].freq > MAX_DROP_FREQ
00092      || results[Side::Right].freq > MAX_DROP_FREQ;
00093
00094 if(cont) {
00095     cout << "An error in aquisition of the doppler audio has occurred! "
00096          "Ensure the connection from the doppler machine to this device "
00097          "is secure and the connection uninterruptable." << endl << endl;
00098 }
00099 }
00100
00101 free(buffer); // free buffer to prevent memory leak
00102 WriteParams(results, filename);
00103
00104 // examine likelihood of avdaspasm
00105 try {
00106     map<Side, DataParams> baseParams = ReadParams(filename);
00107     map<Side, bool> comparison;
00108
00109     for(uint8 i = 0; i < 2; i++) {
00110         Side side = (Side)i;
00111         float comp = (results[side].freq - baseParams[side].freq)
00112              * (baseParams[side].noise - results[side].noise);
00113         comparison[side] = comp > DET_THRESH;
00114     }
00115
00116     string which;
00117
00118     if(comparison[Side::Left] && !comparison[Side::Right]) {
00119         which = "The left";
00120     } else if(!comparison[Side::Left] && comparison[Side::Right]) {
00121         which = "The right";
00122     } else if (comparison[Side::Left] && comparison[Side::Right]) {
00123         which = "Both";
00124     } else {
00125         which = "Neither";
00126     }
00127
00128     cout << which << " side seems to show evidence of a vasospasm." << endl;
00129 } catch(runtime_error ex) {
00130     cout << "These values will be stored as the baseline parameters to "
00131          "which all future parameters are compared." << endl;
00132 }
00133 }

```

Here is the call graph for this function:



## 8.14 main.cpp

```

00001
00010 #include <array>
00011 #include <cstdio>
00012 #include <cstdlib>
00013 #include <fcntl.h>
00014 #include <iostream>
00015 #include <map>
00016 #include <pthread.h>
00017 #include <string>
00018 #include <unistd.h>
00019
00020 #include "definitions.hpp"
00021 #include "fileio.hpp"
00022 #include "process.hpp"
00023
00024 using namespace std;
00025 using namespace avda;
00026
00031 int main(int argc, char** argv) {
00032     // Recorded audio buffer
00033     float32* buffer = (float32*)std::malloc(BUFFER_SIZE);
00034     bool cont = true; // whether to continue in the recording loop
00035     DataParams params[REC_COUNT]; // holds DataParam's from recordings
00036     string filename = PatientName(); // generate name for patient's file
00037     map<Side, DataParams> results; // parameters by side
00038
00039     // arecord command
00040     const string recCommand = string("arecord -t raw -d ")
00041         + to_string(DURATION) + string(" -D plughw:1,0 -f FLOAT -q -r ")
00042         + to_string(SAMPLE_FREQ) + string(" ") + TEMP_FILE;
00043
00044     // Recording
00045     while(cont) {

```

```

00046     for(uint8 i = 0; i < REC_COUNT; i++) {
00047         // prompt
00048         cout << "Press [ ENTER ] to begin analysis for the "
00049             << (i < REC_COUNT / 2 ? "left" : "right") << " side, depth #"
00050             << ((i >= REC_COUNT / 2) ? (i - REC_COUNT / 2) : i) + 1)
00051             << " ";
00052         getchar(); // wait for ENTER to be pressed
00053         cout << "Analyzing..." << endl;
00054
00055         system(recCommand.c_str());
00056         usleep(DURATION*1000000 + 1500000); // sleep DURATION + 1.5 seconds
00057
00058         int file = open(TEMP_FILE.c_str(), O_RDONLY); // open temp file
00059         int retRead = read(file, buffer, BUFFER_SIZE); // copy to buffer
00060         close(file); // close temp file
00061         remove(TEMP_FILE.c_str()); // delete temp file
00062
00063         // if something goes wrong reading the temp file, program exits
00064         if(file < 0 || retRead < BUFFER_SIZE) {
00065             cerr << "An error occurred reading the doppler audio! "
00066                 << "The program will now exit." << endl;
00067             return ERROR;
00068         }
00069
00070         // process and store parameters
00071         params[i] = process(buffer, SAMPLE_COUNT,
SAMPLE_FREQ);
00072         cout << "The analysis is complete." << endl << endl;
00073     }
00074
00075     // calculate averaged parameters
00076     results[Side::Left] = average(params, REC_COUNT / 2);
00077     results[Side::Right] = average(&params[REC_COUNT / 2], REC_COUNT / 2);
00078
00079     cout << "Analysis is complete." << endl << endl;
00080
00081     // print averaged side analysis
00082     for(int i = 0; i < 2; i++) {
00083         Side side = (Side)i;
00084         cout << (side == Side::Left ? "[LEFT]" : "[RIGHT]") << endl;
00085         cout << "Drop-off frequency: " << (uint16)(results[side].freq + 0.5)
00086             << " Hz" << endl;
00087         cout << "Average relative noiseband power: "
00088             << (sint16)(results[side].noise - 0.5) << " dB" << endl << endl;
00089     }
00090
00091     cont = results[Side::Left].freq > MAX_DROP_FREQ
00092         || results[Side::Right].freq > MAX_DROP_FREQ;
00093
00094     if(cont) {
00095         cout << "An error in aquisition of the doppler audio has occurred! "
00096             << "Ensure the connection from the doppler machine to this device "
00097             << "is secure and the connection uninterruptable." << endl << endl;
00098     }
00099 }
00100
00101 free(buffer); // free buffer to prevent memory leak
00102 WriteParams(results, filename);
00103
00104 // examine likelihood of avdaspasm
00105 try {
00106     map<Side, DataParams> baseParams = ReadParams(filename);
00107     map<Side, bool> comparison;
00108
00109     for(uint8 i = 0; i < 2; i++) {
00110         Side side = (Side)i;
00111         float comp = (results[side].freq - baseParams[side].freq)
00112             * (baseParams[side].noise - results[side].noise);
00113         comparison[side] = comp > DET_THRESH;
00114     }
00115
00116     string which;
00117
00118     if(comparison[Side::Left] && !comparison[Side::Right]) {
00119         which = "The left";
00120     } else if(!comparison[Side::Left] && comparison[Side::Right]) {
00121         which = "The right";
00122     } else if (comparison[Side::Left] && comparison[Side::Right]) {
00123         which = "Both";
00124     } else {
00125         which = "Neither";
00126     }
00127
00128     cout << which << " side seems to show evidence of a vasospasm." << endl;
00129 } catch(runtime_error ex) {
00130     cout << "These values will be stored as the baseline parameters to "
00131         << "which all future parameters are compared." << endl;

```

```

00132     }
00133 }

```

## 8.15 src/patient\_name\_test.cpp File Reference

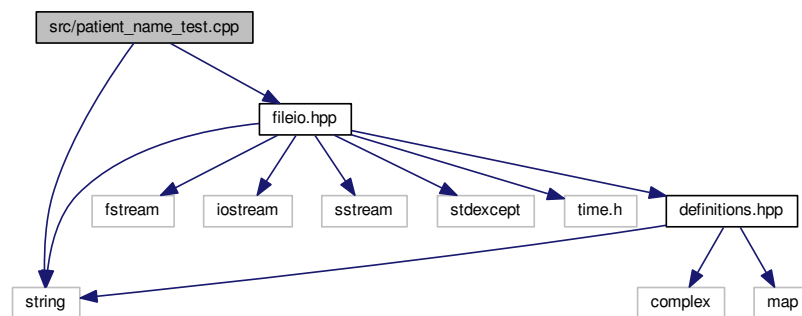
Contains a program to test the [PatientName\(\)](#) function.

```

#include <string>
#include "fileio.hpp"

```

Include dependency graph for patient\_name\_test.cpp:



### Functions

- int [main](#) (int argc, char \*\*argv)

#### 8.15.1 Detailed Description

Contains a program to test the [PatientName\(\)](#) function.

##### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [patient\\_name\\_test.cpp](#).

#### 8.15.2 Function Documentation

##### 8.15.2.1 int main ( int argc, char \*\* argv )

Tests the [PatientName\(\)](#) function from [fileio.hpp](#).

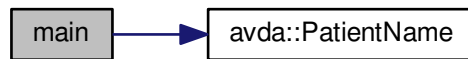
Definition at line 17 of file [patient\\_name\\_test.cpp](#).

```

00017     {
00018         string filename = PatientName();
00019         cout << filename;
00020     }

```

Here is the call graph for this function:



## 8.16 patient\_name\_test.cpp

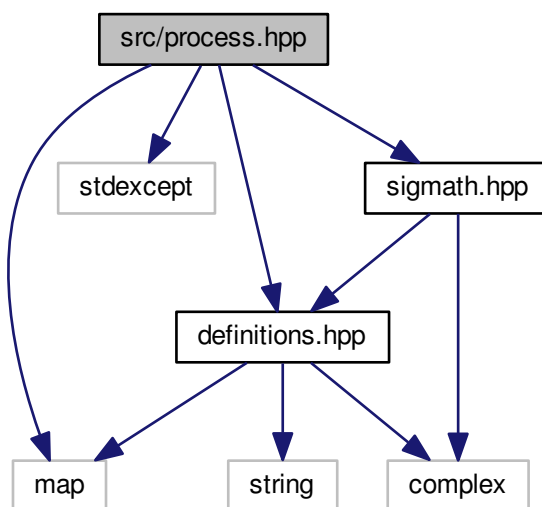
```
00001
00007 #include <string>
00008
00009 #include "fileio.hpp"
00010
00011 using namespace std;
00012 using namespace avda;
00013
00017 int main(int argc, char** argv) {
00018     string filename = PatientName();
00019     cout << filename;
00020 }
```

## 8.17 src/process.hpp File Reference

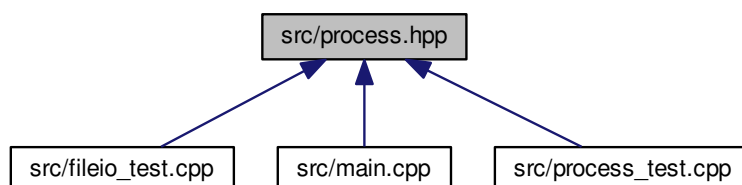
Contains functions related to the program's threaded processing of audio data.

```
#include <map>
#include <stdexcept>
#include "definitions.hpp"
#include "sigmath.hpp"
```

Include dependency graph for process.hpp:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [avda](#)

## Functions

- [DataParams avda::process](#) ([float32](#) \*data, [uint32](#) size, [float32](#) samplingRate)

### 8.17.1 Detailed Description

Contains functions related to the program's threaded processing of audio data.

## Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [process.hpp](#).

## 8.18 process.hpp

```

00001
00008 #ifndef process_H
00009 #define process_H
00010
00011 #include <map>
00012 #include <stdexcept>
00013
00014 #include "definitions.hpp"
00015 #include "sigmath.hpp"
00016
00017 namespace avda {
00048     DataParams process(float32* data, uint32 size,
00049 float32 samplingRate) {
00049         if((size & (size - 1) != 0) || size < 2) {
00050             throw std::invalid_argument(
00051                 "The number of samples is not a power of two!");
00052         }
00053
00054         // declare function-scoped variables
00055         uint32 freqSize = size / 2;
00056         cfloat32* cdata = (cfloat32*)std::malloc(size * sizeof(
00057 cfloat32));
00057         float32* fdata = (float32*)std::malloc(freqSize * sizeof(
00058 float32));
00058         float32* origdata = (float32*)std::malloc(freqSize * sizeof(
00059 float32));
00059
00060         // convert data to complex numbers for fft()
00061         for(uint32 i = 0; i < size; i++) {
00062             cdata[i] = data[i];
00063         }
00064
00065         // find frequency spectrum in relative decibels
00066         fft(cdata, size);
00067         mag(cdata, fdata, freqSize);
00068         Maximum maximum = max(fdata, freqSize);
00069
00070         for(uint32 i = 0; i < freqSize; i++) {
00071             fdata[i] /= maximum.value;
00072         }
00073
00074         decibels(fdata, freqSize);
00075
00076         for(uint32 i = 0; i < freqSize; i++) {
00077             origdata[i] = fdata[i];
00078         }
00079
00080         /*
00081          * Run spectrum values through moving-average filter to smooth the
00082          * curve and make it easier to determine the derivative.
00083          */
00084         smooth(fdata, freqSize, 20);
00085
00086         /*
00087          * Find the derivative of the smoothed spectrum. Note that both this
00088          * filter and the previous are necessary to the algorithm.
00089          */
00090         diff(fdata, freqSize);
00091         smooth(fdata, freqSize, 100);
00092         absolute(fdata, freqSize);
00093
00094         // find the parameters of this specific recording
00095         uint16 offset = 1000;
00096         absolute(&fdata[offset], freqSize - offset);
00097         maximum = max(&fdata[offset], freqSize - offset);
00098         uint32 index = maximum.index + offset;
00099
00100         DataParams params;
00101         params.freq = index * (float)SAMPLE_FREQ / freqSize / 2;
00102         params.noise = average(&origdata[index + offset],
00103             freqSize - offset - index);
00104
00105         free(cdata);
00106         free(fdata);

```



```

00107
00108         return params;
00109     }
00110 }
00111 }
00112
00113 #endif

```

## 8.19 src/process\_test.cpp File Reference

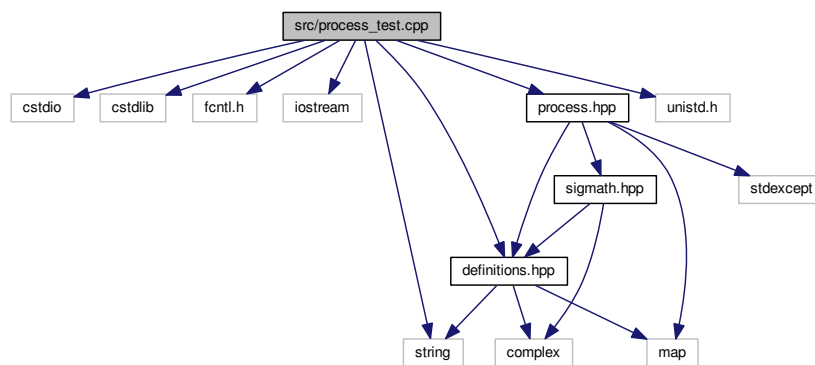
Contains a program to test the [process\(\)](#) function.

```

#include <cstdio>
#include <cstdlib>
#include <fcntl.h>
#include <iostream>
#include <string>
#include <unistd.h>
#include "definitions.hpp"
#include "process.hpp"

```

Include dependency graph for process\_test.cpp:



### Macros

- `#define` [COUNT](#) 131072

### Functions

- `int` [main](#) (`int argc`, `char **argv`)

#### 8.19.1 Detailed Description

Contains a program to test the [process\(\)](#) function.

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)  
 Nicholas K. Nolan

Definition in file [process\\_test.cpp](#).

## 8.19.2 Macro Definition Documentation

### 8.19.2.1 #define COUNT 131072

Definition at line 18 of file [process\\_test.cpp](#).

## 8.19.3 Function Documentation

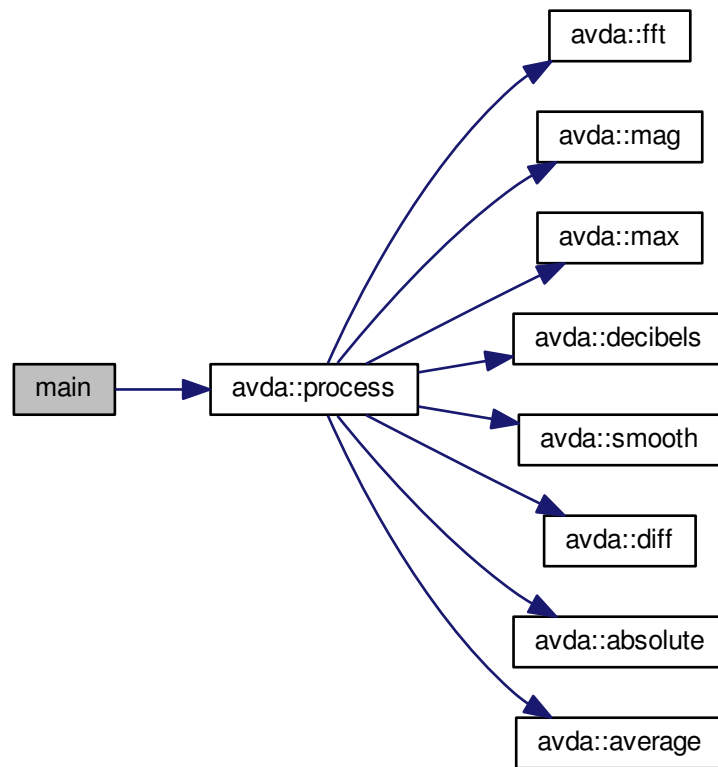
### 8.19.3.1 int main ( int argc, char \*\* argv )

Tests the [process\(\)](#) function from [process.hpp](#).

Definition at line 26 of file [process\\_test.cpp](#).

```
00026         {
00027     int file = open("/home/pi/avda/etc/audio/test.raw", O_RDONLY);
00028
00029     if(file < 0) {
00030         cerr << "File unreadable!" << endl;
00031         return -1;
00032     }
00033
00034     float32* buffer = (float32*)malloc(COUNT * sizeof(float32));
00035     int charRead = read(file, buffer, COUNT * sizeof(float32));
00036
00037     if(charRead < COUNT) {
00038         cerr << "Too few bytes read!" << endl;
00039         return -1;
00040     }
00041
00042     close(file);
00043
00044     DataParams params = process(buffer, COUNT, SAMPLE_FREQ);
00045     free(buffer);
00046     cout << "Cutoff: " << params.freq << endl;
00047     cout << "Noise: " << params.noise << endl;
00048 }
```

Here is the call graph for this function:



## 8.20 process\_test.cpp

```

00001
00008 #include <cstdio>
00009 #include <cstdlib>
00010 #include <fcntl.h>
00011 #include <iostream>
00012 #include <string>
00013 #include <unistd.h>
00014
00015 #include "definitions.hpp"
00016 #include "process.hpp"
00017
00018 #define COUNT 131072
00019
00020 using namespace std;
00021 using namespace avda;
00022
00026 int main(int argc, char** argv) {
00027     int file = open("/home/pi/avda/etc/audio/test.raw", O_RDONLY);
00028
00029     if(file < 0) {
00030         cerr << "File unreadable!" << endl;
00031         return -1;
00032     }
00033
00034     float32* buffer = (float32*)malloc(COUNT * sizeof(float32));
00035     int charRead = read(file, buffer, COUNT * sizeof(float32));
00036
00037     if(charRead < COUNT) {
00038         cerr << "Too few bytes read!" << endl;
00039         return -1;

```

```

00040     }
00041
00042     close(file);
00043
00044     DataParams params = process(buffer, COUNT, SAMPLE_FREQ);
00045     free(buffer);
00046     cout << "Cutoff: " << params.freq << endl;
00047     cout << "Noise: " << params.noise << endl;
00048 }

```

## 8.21 src/read\_params\_test.cpp File Reference

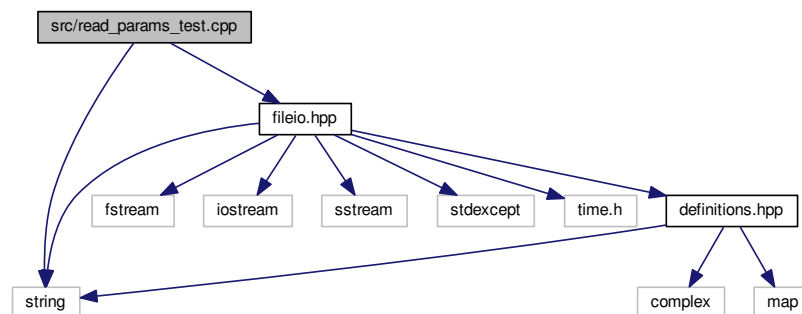
Contains a program test the [PatientName\(\)](#) function.

```

#include <string>
#include "fileio.hpp"

```

Include dependency graph for read\_params\_test.cpp:



## Functions

- int [main](#) (int argc, char \*\*argv)

### 8.21.1 Detailed Description

Contains a program test the [PatientName\(\)](#) function.

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [read\\_params\\_test.cpp](#).

### 8.21.2 Function Documentation

#### 8.21.2.1 int main ( int argc, char \*\* argv )

Tests the [PatientName\(\)](#) function in `fileio.hpp`.

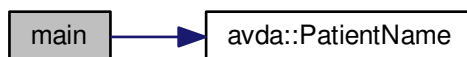
Definition at line 17 of file [read\\_params\\_test.cpp](#).

```

00017     {
00018     string filename = PatientName();
00019     cout << filename;
00020 }

```

Here is the call graph for this function:



## 8.22 read\_params\_test.cpp

```
00001
00007 #include <string>
00008
00009 #include "fileio.hpp"
00010
00011 using namespace std;
00012 using namespace avda;
00013
00017 int main(int argc, char** argv) {
00018     string filename = PatientName();
00019     cout << filename;
00020 }
```

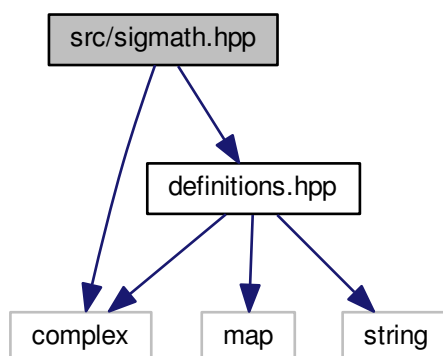
## 8.23 src/sigmath.hpp File Reference

contains the functions necessary to perform the mathematical operations required by this program

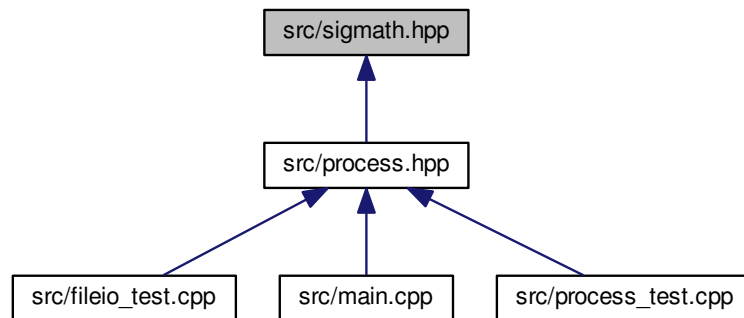
```
#include <complex>
```

```
#include "definitions.hpp"
```

Include dependency graph for sigmath.hpp:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [avda](#)

## Functions

- void [avda::absolute](#) ([float32](#) \*data, [uint32](#) size)
- [float32](#) [avda::average](#) ([float32](#) \*data, [uint32](#) size)
- [DataParams](#) [avda::average](#) ([DataParams](#) \*params, [uint8](#) size)
- void [avda::decibels](#) ([float32](#) \*data, [uint32](#) size)
- void [avda::diff](#) ([float32](#) \*data, [uint32](#) size)
- void [avda::fft](#) ([cfloat32](#) \*data, [uint32](#) size)
- void [avda::mag](#) ([cfloat32](#) \*orig, [float32](#) \*newmags, [uint32](#) size)
- [Maximum](#) [avda::max](#) ([float32](#) \*data, [uint32](#) size)
- void [avda::smooth](#) ([float32](#) \*data, [uint32](#) size, [uint16](#) order)

### 8.23.1 Detailed Description

contains the functions necessary to perform the mathematical operations required by this program

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)  
 Nicholas K. Nolan

Definition in file [sigmath.hpp](#).

## 8.24 sigmath.hpp

```

00001
00009 #ifndef sigmath_H
00010 #define sigmath_H
00011
00012 #include <complex>
00013 #include "definitions.hpp"
00014

```

```

00015 namespace avda {
00016     // PROTOTYPES
00017
00026     void absolute(float32* data, uint32 size);
00027
00037     float32 average(float32* data, uint32 size);
00038
00049     DataParams average(DataParams* params, uint8 size);
00050
00062     void decibels(float32* data, uint32 size);
00063
00072     void diff(float32* data, uint32 size);
00073
00085     void fft(cffloat32* data, uint32 size);
00086
00096     void mag(cffloat32* orig, float32* newmags, uint32 size);
00097
00107     Maximum max(float32* data, uint32 size);
00108
00119     void smooth(float32* data, uint32 size, uint16 order);
00120
00121     // DEFINITIONS
00122
00123     void absolute(float32* data, uint32 size) {
00124         for(uint32 i = 0; i < size; i++) {
00125             data[i] = fabsf(data[i]);
00126         }
00127     }
00128
00129     float32 average(float32* data, uint32 size) {
00130         float32 ave;
00131
00132         for(uint32 i = 0; i < size; i++) {
00133             ave += data[i];
00134         }
00135
00136         ave = ave / size;
00137         return ave;
00138     }
00139
00140     DataParams average(DataParams* params, uint8 size) {
00141         DataParams ave;
00142
00143         for(uint8 i = 0; i < size; i++) {
00144             //freq is an attribute. this is how to add structure attributes
00145             ave.freq += params[i].freq;
00146             ave.noise += params[i].noise;
00147         }
00148
00149         ave.freq /= size;
00150         ave.noise /= size;
00151         return ave;
00152     }
00153
00154     void decibels(float32* data, uint32 size) {
00155         for(uint32 i = 0; i < size; i++) {
00156             data[i] = 20 * log10(data[i]);
00157         }
00158     }
00159
00160     void diff(float32* data, uint32 size) {
00161         float32 temp[size];
00162         temp[0] = 0;
00163
00164         for(uint32 i = 1; i < size; i++) {
00165             temp[i] = data[i] - data[i-1];
00166         }
00167
00168         for(uint32 i = 0; i < size; i++) {
00169             data[i] = temp[i];
00170         }
00171     }
00172
00173     void fft(cffloat32* data, uint32 size) {
00174         // DFT
00175         uint32 k = size;
00176         uint32 n;
00177         float32 thetaT = M_PI / size;
00178         cffloat32 phiT(cos(thetaT), sin(thetaT));
00179         cffloat32 T;
00180
00181         while(k > 1) {
00182             n = k;
00183             k >>= 1;
00184             phiT = phiT * phiT;
00185             T = 1.0L;
00186

```

```

00187         for(uint32 l = 0; l < k; l++) {
00188             for(uint32 a = l; a < size; a += n) {
00189                 uint32 b = a + k;
00190                 cfloat32 t = data[a] - data[b];
00191                 data[a] += data[b];
00192                 data[b] = t * T;
00193             }
00194         }
00195         T *= phiT;
00196     }
00197 }
00198
00199 // Decimate
00200 uint32 m = (uint32)log2(size);
00201
00202 for(uint32 a = 0; a < size; a++) {
00203     uint32 b = a;
00204
00205     // Reverse bits
00206     b = ((b & 0xaaaaaaaa) >> 1) | ((b & 0x55555555) << 1);
00207     b = ((b & 0xcccccccc) >> 2) | ((b & 0x33333333) << 2);
00208     b = ((b & 0xf0f0f0f0) >> 4) | ((b & 0x0f0f0f0f) << 4);
00209     b = ((b & 0xff00ff00) >> 8) | ((b & 0x00ff00ff) << 8);
00210     b = ((b >> 16) | (b << 16)) >> (32 - m);
00211
00212     if (b > a)
00213     {
00214         cfloat32 t = data[a];
00215         data[a] = data[b];
00216         data[b] = t;
00217     }
00218 }
00219 }
00220
00221 void mag(cfloat32* orig, float32* newmags, uint32 size) {
00222     //loop to run through the length of array orig
00223     for(uint32 n = 0; n < size; n++) {
00224         /*
00225          * abs should calculate the magnitude of complex array elements.
00226          * saves to new array
00227          */
00228         newmags[n] = std::abs(orig[n]);
00229     }
00230 }
00231
00232 Maximum max(float32* data, uint32 size) {
00233     Maximum m;
00234
00235     //loop to run through the length of array data
00236     for (uint32 i = 0; i < size; i++) {
00237         /*
00238          * when value at data[i] is above max.value,
00239          * sets max.value equal to data[i] and max.index equal to i
00240          */
00241         if (data[i] > m.value) {
00242             m.value = data[i];
00243             m.index = i;
00244         }
00245     }
00246
00247     return m;
00248 }
00249
00250 void smooth(float32* data, uint32 size, uint16 order) {
00251     float32 coeff = 1 / (float32)order;
00252     float32 temp[size];
00253
00254     for(uint32 i = 0; i < size; i++) {
00255         temp[i] = 0;
00256
00257         for(uint16 j = 0; j < order && j <= i; j++) {
00258             temp[i] += data[i - j];
00259         }
00260
00261         temp[i] *= coeff;
00262     }
00263
00264     for(uint32 i = 0; i < size; i++) {
00265         data[i] = temp[i];
00266     }
00267 }
00268 }
00269
00270 #endif

```



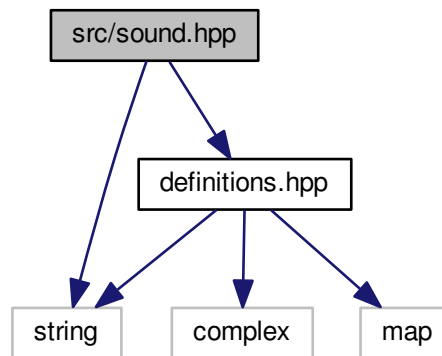
## 8.25 src/sound.hpp File Reference

contains the function(s) relating to sound

```
#include <string>
```

```
#include "definitions.hpp"
```

Include dependency graph for sound.hpp:



### Namespaces

- [avda](#)

### Functions

- void [avda::play](#) (auto filename)

#### 8.25.1 Detailed Description

contains the function(s) relating to sound

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)

Definition in file [sound.hpp](#).

## 8.26 sound.hpp

```
00001
00007 #ifndef sound_H
00008 #define sound_H
00009
00010 #include <string>
00011
00012 #include "definitions.hpp"
00013
00014 namespace avda {
00020     void play(auto filename) {
```

```

00021
00022     }
00023 }
00024
00025 #endif

```

## 8.27 src/stdin\_clear\_test.cpp File Reference

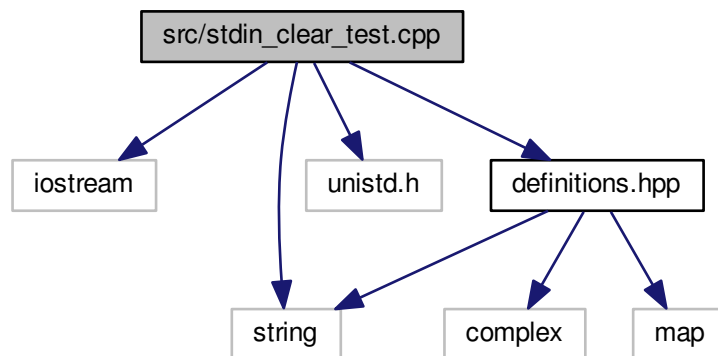
Contains a program to test clearing the stdin buffer.

```

#include <iostream>
#include <string>
#include <unistd.h>
#include "definitions.hpp"

```

Include dependency graph for stdin\_clear\_test.cpp:



### Macros

- `#define` `COUNT` 80

### Functions

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

#### 8.27.1 Detailed Description

Contains a program to test clearing the stdin buffer.

#### Author

Samuel Andrew Wisner, [awisner94@gmail.com](mailto:awisner94@gmail.com)  
 Nicholas K. Nolan

Definition in file `stdin_clear_test.cpp`.

## 8.27.2 Macro Definition Documentation

### 8.27.2.1 #define COUNT 80

Definition at line 14 of file [stdin\\_clear\\_test.cpp](#).

## 8.27.3 Function Documentation

### 8.27.3.1 int main ( int argc, char \*\* argv )

Tests the ability to clear the stdin buffer.

Definition at line 22 of file [stdin\\_clear\\_test.cpp](#).

```
00022         {
00023     char text1[COUNT];
00024     char text2[COUNT];
00025
00026     cout << "Enter text to ignore: ";
00027     cout.flush();
00028     read(STDIN_FILENO, &text1, COUNT);
00029     // fflush(stdin);
00030     cout << endl << "Enter text to print: ";
00031     cout.flush();
00032     read(STDIN_FILENO, &text2, COUNT);
00033     cout << endl << "In buffer: " << text2 << endl;
00034 }
```

## 8.28 stdin\_clear\_test.cpp

```
00001
00008 #include <iostream>
00009 #include <string>
00010 #include <unistd.h>
00011
00012 #include "definitions.hpp"
00013
00014 #define COUNT 80
00015
00016 using namespace std;
00017 using namespace avda;
00018
00022 int main(int argc, char** argv) {
00023     char text1[COUNT];
00024     char text2[COUNT];
00025
00026     cout << "Enter text to ignore: ";
00027     cout.flush();
00028     read(STDIN_FILENO, &text1, COUNT);
00029     // fflush(stdin);
00030     cout << endl << "Enter text to print: ";
00031     cout.flush();
00032     read(STDIN_FILENO, &text2, COUNT);
00033     cout << endl << "In buffer: " << text2 << endl;
00034 }
```

# Index

- absolute
  - avda, [20](#)
- avda, [19](#)
  - absolute, [20](#)
  - average, [20](#), [21](#)
  - decibels, [21](#)
  - diff, [22](#)
  - fft, [22](#)
  - Left, [20](#)
  - mag, [23](#)
  - max, [25](#)
  - play, [27](#)
  - process, [27](#)
  - Right, [20](#)
  - Side, [19](#)
  - smooth, [32](#)
- average
  - avda, [20](#), [21](#)
- decibels
  - avda, [21](#)
- diff
  - avda, [22](#)
- fft
  - avda, [22](#)
- index
  - Maximum, [36](#)
- Left
  - avda, [20](#)
- mag
  - avda, [23](#)
- makefile, [38](#)
- max
  - avda, [25](#)
- Maximum, [35](#)
  - index, [36](#)
  - value, [36](#)
- play
  - avda, [27](#)
- process
  - avda, [27](#)
- Right
  - avda, [20](#)
- Side
  - avda, [19](#)
- smooth
  - avda, [32](#)
- value
  - Maximum, [36](#)