Lab 1

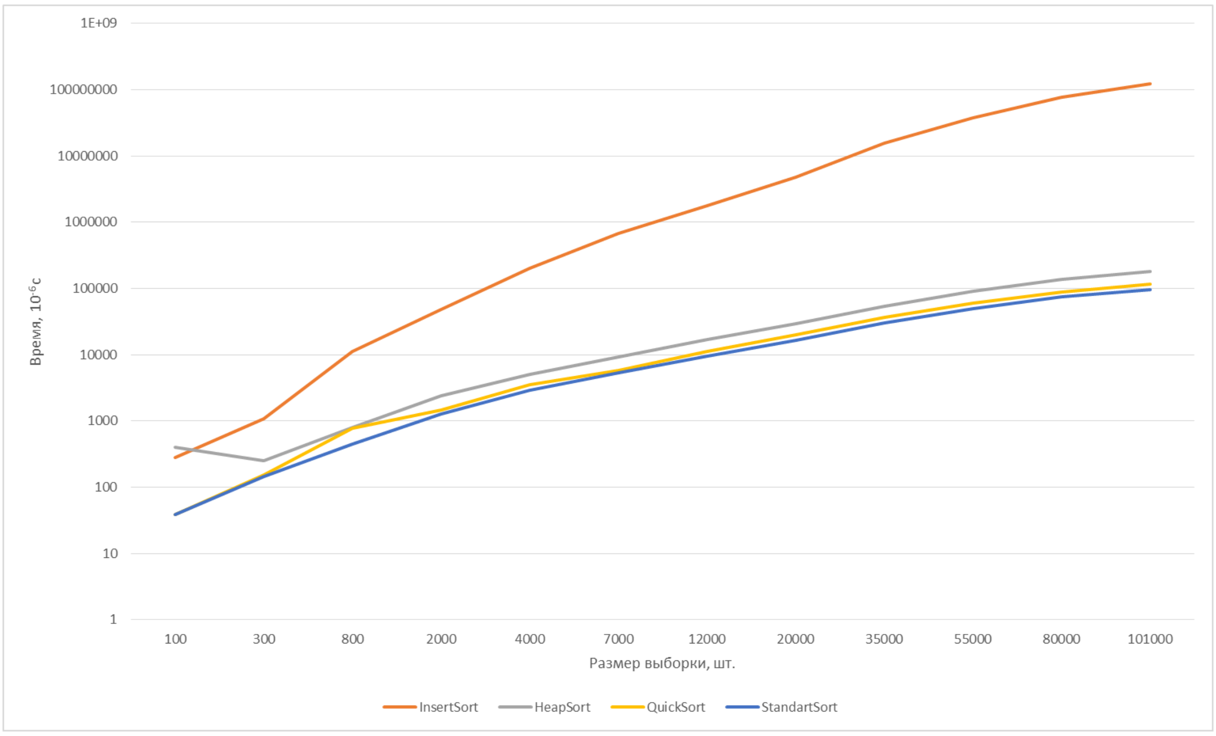
Generated by Doxygen 1.9.4

Chapter 1

Lab 1

Link to repository: https://github.com/mrzrow/pm-labs

Results



Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Athlete Class Reference

Athlete Class.

```
#include <athlete.h>
```

Public Member Functions

- Athlete ()

    Default constructor.
- Athlete (const std::string &n, int a, int h, int w, const std::string &s)
- ∼Athlete ()

    Destructor.
- bool operator== (const Athlete &other)

    overloading equal
- bool operator< (const Athlete &other)

    overloading less
- bool operator!= (const Athlete &other)

    overloading not equal
- bool operator<= (const Athlete &other)

    overloading less or equal
- bool operator> (const Athlete &other)

    overloading greater
- bool operator>= (const Athlete &other)

    overloading greater or equal

Public Attributes

- std::string name
- int age
- int height
- int weight
- std::string sport

4.1.1 Detailed Description

Athlete Class.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Athlete() [1/2]

```
Athlete::Athlete ( )  [inline]
```

Default constructor.

4.1.2.2 Athlete() [2/2]

```
Athlete::Athlete (
              const std::string & n,
              int a,
              int h,
              int w,
              const std::string & s )  [inline]
```

Constructor

Parameters

n	name
a	age
h	height
w	weight
s	sport

4.1.2.3 ∼Athlete()

```
Athlete::∼Athlete ( )  [inline]
```

Destructor.

4.1.3 Member Function Documentation

4.1.3.1 operator"!=()

```
bool Athlete::operator!= (
             const Athlete & other )  [inline]
```

overloading not equal

4.1.3.2 operator<()

```
bool Athlete::operator< (
             const Athlete & other )  [inline]
```

overloading less

4.1.3.3 operator<=()

```
bool Athlete::operator<= (
             const Athlete & other )  [inline]
```

overloading less or equal

4.1.3.4 operator==()

```
bool Athlete::operator== (
             const Athlete & other )  [inline]
```

overloading equal

4.1.3.5 operator>()

```
bool Athlete::operator> (
             const Athlete & other )  [inline]
```

overloading greater

4.1.3.6 operator>=()

```
bool Athlete::operator>= (
             const Athlete & other )  [inline]
```

overloading greater or equal

4.1.4 Member Data Documentation

4.1.4.1 age

```
int Athlete::age
```

4.1.4.2 height

```
int Athlete::height
```

4.1.4.3 name

```
std::string Athlete::name
```

4.1.4.4 sport

```
std::string Athlete::sport
```

4.1.4.5 weight

```
int Athlete::weight
```

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

- athlete.h

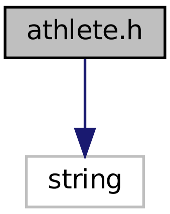
Chapter 5

File Documentation

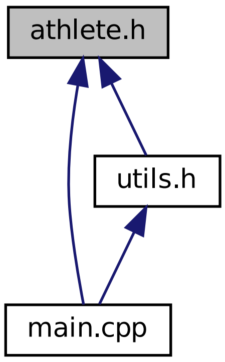
5.1 athlete.h File Reference

```
#include <string>
```

Include dependency graph for athlete.h:



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



Classes

- class Athlete

  Athlete Class.

5.2 athlete.h

Go to the documentation of this file.

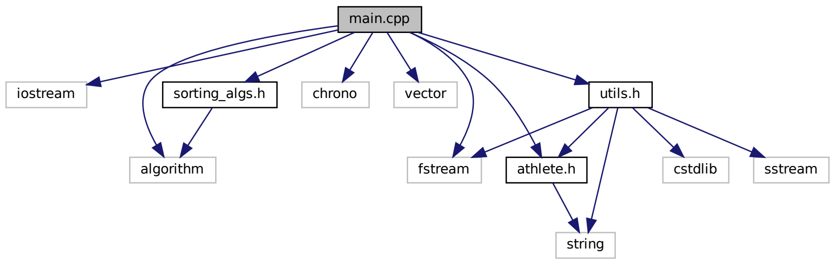
```
#ifndef ATHLETE_H_
#define ATHLETE_H_

#include <string>


class Athlete {
public:
    std::string name;
    int age;
    int height;
    int weight;
    std::string sport;

public:
    Athlete() {}
    Athlete(const std::string& n, int a, int h, int w, const std::string& s):
        name(n), age(a), height(h), weight(w), sport(s) {}
    ~Athlete() {}

    bool operator==(const Athlete& other) {
        return sport == other.sport &&
               name  == other.name  &&
               age   == other.age;
    }

    bool operator<(const Athlete& other) {
        if ( sport != other.sport )
            return sport < other.sport;
        if ( name != other.name )
            return name < other.name;
        return age < other.age;
    }

    bool operator!=(const Athlete& other) { return !(*this == other);               }
    bool operator<=(const Athlete& other) { return *this == other || *this < other; }
    bool operator> (const Athlete& other) { return !(*this <= other);               }
    bool operator>=(const Athlete& other) { return *this > other || *this == other; }
};

#endif
```

5.3 main.cpp File Reference

```
#include <iostream>
#include <algorithm>
#include <chrono>
#include <vector>
#include <fstream>
#include "athlete.h"
#include "sorting_algs.h"
```

```
#include "utils.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

5.3.1 Function Documentation

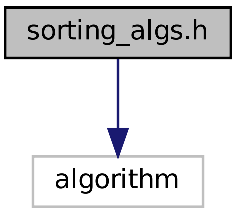
5.3.1.1 main()

```
int main ( )
```

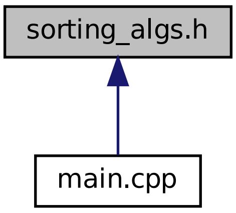
5.4 sorting_algs.h File Reference

```
#include <algorithm>
```

Include dependency graph for sorting_algs.h:



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



Functions

- template<class T >
  void insertSort (T ∗a, long size)
- template<class T >
  void downHeap (T ∗a, long k, long n)
- template<class T >
  void heapSort (T ∗a, long size)
- template<class T >
  void quickSortR (T ∗a, long size)
- template<class T >
  void standartSort (T ∗a, long size)

5.4.1 Function Documentation

5.4.1.1 downHeap()

```
template<class T >
void downHeap (
             T * a,
             long k,
             long n )
```

Building pyramid

Parameters

k	start index
n	end index

5.4.1.2 heapSort()

```
template<class T >
void heapSort (
             T * a,
             long size )
```

Heap sort

Parameters

a	array to sort
size	size of the array

5.4.1.3 insertSort()

```
template<class T >
void insertSort (
             T * a,
             long size )
```

Insert sort

Parameters

a	array to sort
size	size of the array

5.4.1.4 quickSortR()

```
template<class T >
void quickSortR (
             T * a,
             long size )
```

Quick sort

Parameters

a	array to sort
size	size of the array

5.4.1.5 standartSort()

```
template<class T >
void standartSort (
            T * a,
            long size )
```

Wrapper for std::sort

Parameters

a	array to sort
size	size of the array

5.5 sorting_algs.h

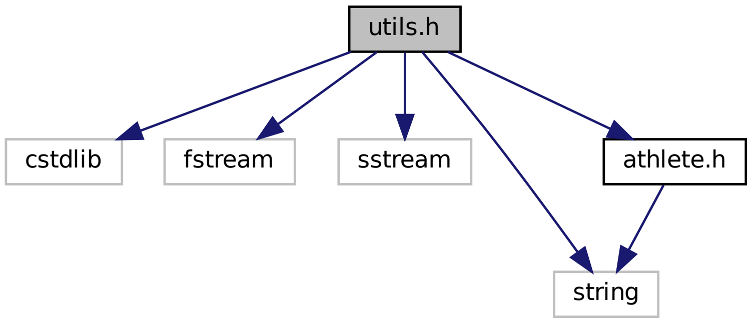
Go to the documentation of this file.

```
#ifndef SORTING_ALGS_H_
#define SORTING_ALGS_H_

#include <algorithm>

template<class T> void insertSort(T* a, long size) {
    T x;
    long i, j;

    for (i = 0; i < size; i++) {
        x = a[i];

        for ( j=i-1; j>=0 && a[j] > x; j--)
            a[j+1] = a[j];

        a[j+1] = x;
  }
}

template<class T> void downHeap(T* a, long k, long n) {
    T new_elem;
    long child;
    new_elem = a[k];

    while(k <= n/2) {
        child = 2*k;
        if( child < n && a[child] < a[child+1] ) child++;
        if( new_elem >= a[child] ) break;
        a[k] = a[child];
        k = child;
    }
    a[k] = new_elem;
}
template<class T> void heapSort(T* a, long size) {
    long i;
    T temp;

    for(i=size/2-1; i >= 0; i--)
        downHeap(a, i, size-1);

    for(i=size-1; i > 0; i--) {
        std::swap(a[i], a[0]);
        downHeap(a, 0, i-1);
    }
}

template<class T> void quickSortR(T* a, long size)
{
    long i = 0, j = size-1;
    T p = a[ size»1 ];

    do {
        while ( a[i] < p ) i++;
        while ( a[j] > p ) j--;

        if (i <= j) {
            std::swap(a[i], a[j]);
            i++; j--;
        }
    } while ( i<=j );

    if ( j > 0 ) quickSortR(a, j);
    if ( size > i ) quickSortR(a+i, size-i);
}

template<class T> void standartSort(T* a, long size) {
    std::sort(a, a + size);
}

#endif
```

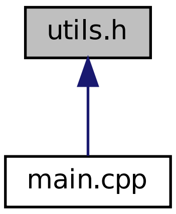
5.6 utils.h File Reference

```
#include <cstdlib>
#include <fstream>
#include <sstream>
#include <string>
#include "athlete.h"
```

Include dependency graph for utils.h:



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



Functions

- std::string randomString (int length)
- void generateAthletesCSV (const std::string &filename, int total=101000)
- void fillAthleteArray (Athlete ∗arr, int n, const std::string &filename="athletes.csv")

5.6.1 Function Documentation

5.6.1.1 fillAthleteArray()

```
void fillAthleteArray (
            Athlete * arr,
            int n,
            const std::string & filename = "athletes.csv" )
```

Fill array with Athlete objects

Parameters

arr	array to fill
n	number of athletes
filename	name of the file to read

5.6.1.2 generateAthletesCSV()

```
void generateAthletesCSV (
            const std::string & filename,
            int total = 101000 )
```

Generates file of athletes data

Parameters

filename	name of the file to save
total	number of lines of data

5.6.1.3 randomString()

```
std::string randomString (
            int length )
```

Generates random string

Parameters

length	length of the string

Returns

generated string

5.7 utils.h

Go to the documentation of this file.

```cpp
#ifndef UTILS_H_
#define UTILS_H_

#include <cstdlib>
#include <fstream>
#include <sstream>
#include <string>

#include "athlete.h"

std::string randomString(int length) {
    const std::string chars = "abcdefghijklmnopqrstuvwxyz";

    std::string result;
    for (int i = 0; i < length; ++i)
        result += chars[rand() % chars.size()];
    return result;
}


void generateAthletesCSV(const std::string& filename, int total = 101000) {
    const std::string sports[] = {"Football", "Basketball", "Tennis", "Swimming", "Running"};
    int numSport = 5;

    std::ofstream file(filename);
    if (!file.is_open()) return;

    for (int i = 0; i < total; ++i) {
        std::string name = randomString(5 + rand() % 5);
        int age    = 18  + rand() % 23;
        int height = 160 + rand() % 41;
        int weight = 50  + rand() % 51;
        std::string sport = sports[rand() % numSport];

        file « name « "," « age « "," « height « "," « weight « "," « sport « "\n";
    }

    file.close();
}


void fillAthleteArray(Athlete* arr, int n, const std::string& filename = "athletes.csv") {
    std::ifstream file(filename);
    if (!file.is_open()) return;

    std::string line;
    int count = 0;

    while (count < n && std::getline(file, line)) {
        std::stringstream ss(line);
        std::string name, ageStr, heightStr, weightStr, sport;

        std::getline(ss, name, ',');
        std::getline(ss, ageStr, ',');
        std::getline(ss, heightStr, ',');
        std::getline(ss, weightStr, ',');
        std::getline(ss, sport);

        int age    = std::stoi(ageStr);
        int height = std::stoi(heightStr);
        int weight = std::stoi(weightStr);

        arr[count++] = Athlete(name, age, height, weight, sport);
    }

    file.close();
}

#endif
```