A412

0.1

Genereret af Doxygen 1.8.8

Tir Dec 9 2014 09:46:38

Indhold

Kapitel 1

Indeks over datastrukturer

1.1 Datastrukturer

Her er datastrukturerne med korte beskrivelser:

Kapitel 2

Fil-indeks

2.1 Filoversigt

Her er en liste over alle filer med korte beskrivelser:

Kapitel 3

Datastruktur-documentation

3.1 data Datastruktur-reference

Datafelter

- unsigned int tempo
- mode mode

3.1.1 Detaljeret beskrivelse

Defineret på linje 31 i filen main.c.

3.1.2 Felt-dokumentation

3.1.2.1 mode data::mode

Defineret på linje 33 i filen main.c.

3.1.2.2 unsigned int data::tempo

Defineret på linje 32 i filen main.c.

Dokumentationen for denne datastruktur blev genereret ud fra filen:

- main.c

3.2 moodWeighting Datastruktur-reference

Datafelter

- int mode
- int tempo
- int toneLength
- int pitch

3.2.1 Detaljeret beskrivelse

Defineret på linje 41 i filen main.c.

3.2.2 Felt-dokumentation

3.2.2.1 int moodWeighting::mode

Defineret på linje 42 i filen main.c.

3.2.2.2 int moodWeighting::pitch

Defineret på linje 45 i filen main.c.

3.2.2.3 int moodWeighting::tempo

Defineret på linje 43 i filen main.c.

3.2.2.4 int moodWeighting::toneLength

Defineret på linje 44 i filen main.c.

Dokumentationen for denne datastruktur blev genereret ud fra filen:

- main.c

3.3 note Datastruktur-reference

Datafelter

- int tone
- int octave
- int lenght

3.3.1 Detaljeret beskrivelse

Defineret på linje 25 i filen main.c.

3.3.2 Felt-dokumentation

3.3.2.1 int note::lenght

Defineret på linje 28 i filen main.c.

3.3.2.2 int note::octave

Defineret på linje 27 i filen main.c.

3.3.2.3 int note::tone

Defineret på linje 26 i filen main.c.

Dokumentationen for denne datastruktur blev genereret ud fra filen:

- main.c

3.4 points Datastruktur-reference

Datafelter

- char ∗ parameter
- int point

3.4.1 Detaljeret beskrivelse

Defineret på linje 36 i filen main.c.

3.4.2 Felt-dokumentation

3.4.2.1 char∗ points::parameter

Defineret på linje 37 i filen main.c.

3.4.2.2 int points::point

Defineret på linje 38 i filen main.c.

Dokumentationen for denne datastruktur blev genereret ud fra filen:

- main.c

Kapitel 4

Fil-dokumentation

4.1 main.c filreference

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

Datastrukturer

- struct note
- struct data
- struct points
- struct moodWeighting

#Defines

- #define CHARS 1000
- #define AMOUNT_OF_MOODS 2

Typedefinitioner

- typedef enum mode mode
- typedef enum tone tone
- typedef enum mood mood

Enumerationer

- enum mode { major, minor }
- enum tone {
  C, Csharp, D, Dsharp,
  E, F = 6, Fsharp, G,
  Gsharp, A, Asharp, B }
- enum mood { glad, sad }

Funktioner

- void findNoteLength (double x, int ∗, int ∗)
- void printNote (note)
- int getHex (FILE ∗, int[ ])
- void fillSongData (data ∗, int[ ], int)
- int countNotes (int[ ], int)
- void fillNote (int, note ∗)
- void printSongData (data)
- void insertMoods (moodWeighting[ ])
- int weightingMatrix (moodWeighting[ ], int, int, int, int)
- void findEvents (int, int[ ], note[ ])
- int sortResult (const void ∗, const void ∗)
- int deltaTimeToNoteLength (int, int)
- int main (int argc, const char ∗argv[ ])
- void settingPoints (int ∗mode, int ∗tempo, int ∗length, int ∗octave, data data)

4.1.1 #Define-dokumentation

4.1.1.1 #define AMOUNT_OF_MOODS 2

Defineret på linje 18 i filen main.c.

4.1.1.2 #define CHARS 1000

Defineret på linje 17 i filen main.c.

4.1.2 Dokumentation af typedefinitioner

4.1.2.1 typedef enum **mode mode**

4.1.2.2 typedef enum **mood mood**

4.1.2.3 typedef enum **tone tone**

4.1.3 Dokumentation af enumerations-typer

4.1.3.1 enum **mode**

Enumerationsværdier

major

minor

Defineret på linje 21 i filen main.c.

```
21 {major, minor} mode;
```

4.1.3.2 enum mood

Enumerationsværdier

glad

sad

Defineret på linje 23 i filen main.c.

```
{glad, sad} mood;
```

4.1.3.3 enum tone

Enumerationsværdier

C

Csharp

D

Dsharp

E

F

Fsharp

G

Gsharp

A

Asharp

B

Defineret på linje 22 i filen main.c.

```
{C, Csharp, D, Dsharp, E, F = 6, Fsharp, G, Gsharp,
    A, Asharp, B} tone;
```

4.1.4 Funktions-dokumentation

4.1.4.1 int countNotes (int *hex[ ]*, int *amount*)

A function to count the number of notes in the entire song

Parametre

int]	hex[]: an array with the stored information from the file
int]	amount: an integer holding the total number of characters in the array

Defineret på linje 119 i filen main.c.

```
                                    {
  int i = 0, res = 0;
  for(i = 0; i < amount; i++){
    if(hex[i] == 0x90){
      res++;
    }
  }
  return res;
}
```

4.1.4.2 int deltaTimeToNoteLength (int *ticks,* int *ppqn*)

Defineret på linje 317 i filen main.c.

```
                                              {
  return (int) (round((4*ticks)/ppqn));
}
```

4.1.4.3 void fillNote (int *inputTone,* note ∗ *note*)

A function to fill out each of the structures of type note

Parametre

int]	inputTone: the value of the hexadecimal collected on the "tone"-spot
note∗]	note: a pointer to a note-structure

Defineret på linje 201 i filen main.c.

```
                                       {
  note->tone = inputTone % 12;
  note->octave = inputTone / 12;
}
```

4.1.4.4 void fillSongData (data ∗ *data,* int *hex[ ],* int *numbersInText*)

A function, that fills out the song data

Parametre

data∗]	data: a pointer to a structure containing the tempo and mode of the song
int]	hex[]:the array of integers read from the file
int]	numbersInText: the total amount of integers in the array

Defineret på linje 134 i filen main.c.

```
                                                           {
  int j;
  /*Find the mode of the song, initialised as minor atm*/
  data->mode = minor;
  for(j = 0; j < numbersInText; j++){
    /* finds the tempo */
    if(hex[j] == 0xff){
      if(hex[j+1] == 0x51){
        data->tempo =  60000000/((hex[j+3] << 16) | (hex[j+4] << 8) | (hex[j+5]));
      }
    }
  }
}
```

4.1.4.5 void findEvents (int *numbersInText,* int *hex[ ],* note *noteAr[ ]*)

Defineret på linje 148 i filen main.c.

```
                                                             {
  int note = 0x01, eventType = 0x01, counter = 0, i = 0;
  /*Read and proces the hex array*/
  for(int j = 0; j < numbersInText; j++){
    /* Hops over any noto-on, note-off or metaevent start
       Also stores the tones read after a note-on          */
    if(hex[j] == 0x00 && (hex[j + 1] == 0x90 || hex[j + 1] == 0xff)){
      counter = 1;
      j += 4;
      if(hex[j - 3] == 0x90){
        note = hex[j - 2];
```

```
159         fillNote(hex[j - 2], &noteAr[i]);
160         i++;
161       }
162       else{
163         eventType = hex[j - 2];
164       }
165     }
166     else if(hex[j] == 0x80 && hex[j + 1] == note){
167       j += 2;
168       note = 0x01;
169       counter = 0;
170     }
171     if(counter){
172       /* Here you can check for parameters inside a meta-event or MIDI-event */
173     }
174     else{
175       /* Here you can check for parameters outside a meta-event or MIDI-event
176          e.g. between a note-off and the next MIDI-event or a meta-event      */
177     }
178   }
179 }
```

4.1.4.6 void findNoteLength (double *x,* int ∗ *high,* int ∗ *low*)

A function to calculate the notelenght - tba

Defineret på linje 183 i filen main.c.

```
183                                                 {
184   double func = 16*((x*x)*(0.0000676318287050830)+(0.0128675448628599*x)-2.7216713227147);
185   double temp = func;
186   double temp2 = (int) temp;
187 
188   if (!(temp - (double) temp2 < 0.5)){
189     func += 1;
190   }
191 
192   printf("x: %f og func: %f\n", x, func);
193   *high = (int) func;
194   *low = 16;
195 }
```

4.1.4.7 int getHex (FILE ∗ *f,* int *hexAr[ ]*)

A function, that retrieves the hexadecimals from the files and also returns the number of files

Parametre

FILE∗]	f: a pointer to the file the program is reading from
int]	hexAr[]: an array of integers, that the information is stored in

Defineret på linje 104 i filen main.c.

```
104                              {
105   int i = 0, c;
106 
107   while( (c = fgetc(f)) != EOF && i < CHARS){
108     hexAr[i] = c;
109     i++;
110   }
111 
112   return i;
113 }
```

4.1.4.8 void insertMoods (moodWeighting *moodArray[ ]*)

Defineret på linje 284 i filen main.c.

```
284                                             {
285   moodArray[glad].mode             = 3;
286   moodArray[glad].tempo          = 4;
287   moodArray[glad].toneLength     = 2;
288   moodArray[glad].pitch          = 1;
289
290   moodArray[sad].mode            = -4;
291   moodArray[sad].tempo           = -5;
292   moodArray[sad].toneLength      = -3;
293   moodArray[sad].pitch           = 0;
294 }
```

4.1.4.9 int main (int *argc,* const char ∗ *argv[]*)

Defineret på linje 62 i filen main.c.

```
62                                        {
63   /*Variables*/
64   int numbersInText = 0, notes, i = 0, moodOfMelodi = 0;
65   /* PLACEHOLDER FIX THIS */
66   int mode = 5, tempo = 5, toneLength = 5, pitch = 5;
67   moodWeighting moodArray[AMOUNT_OF_MOODS];
68   data data;
69   FILE *f = fopen(argv[1],"r");
70   int *hex = (int *) malloc(CHARS * sizeof(int));
71   if(hex == NULL){
72     printf("Memory allokation failed, bye!");
73     exit(EXIT_FAILURE);
74   }
75
76   /*Reading the data from the file*/
77   numbersInText = getHex(f, hex);
78   fillSongData(&data, hex, numbersInText);
79   notes = countNotes(hex, numbersInText);
80   note *noteAr = (note*) malloc(notes * sizeof(note));
81   if(noteAr == NULL){
82     printf("Memory allocation failed, bye!");
83     exit(EXIT_FAILURE);
84   }
85   findEvents(numbersInText, hex, noteAr);
86   insertMoods(moodArray);
87   for(i = 0; i < notes; i++)
88     printNote(noteAr[i]);
89   printSongData(data);
90   moodOfMelodi = weightingMatrix(moodArray, mode, tempo, toneLength, pitch);
91
92   /*Clean up and close*/
93   fclose(f);
94   free(hex);
95   free(noteAr);
96
97   return 0;
98 }
```

4.1.4.10 void printNote (note *note*)

A function to print the note

Parametre

note]	note: the note structure to be printed

Defineret på linje 209 i filen main.c.

```
209                         {
210   printf("Tone: ");
211
212   switch (note.tone){
213     case C      : printf("C") ; break;
214     case Csharp: printf("C#"); break;
215     case D      : printf("D") ; break;
216     case Dsharp: printf("D#"); break;
217     case E      : printf("E") ; break;
218     case F      : printf("F") ; break;
219     case Fsharp: printf("F#"); break;
220     case G      : printf("G") ; break;
221     case Gsharp: printf("G#"); break;
```

```
    case A     : printf("A") ; break;
    case Asharp: printf("A#"); break;
    case B     : printf("B") ; break;
    default    : printf("Undefined note"); break;
  }
  printf(", octave: %d\n", note.octave);
}
```

4.1.4.11 void printSongData (data *data*)

A function to print out the overall data of the song, tempo and mode

Parametre

data]	data: the data to be printed

Defineret på linje 233 i filen main.c.

```
                            {
  printf("Tempo: %d\nMode: ", data.tempo);
  switch(data.mode){
    case minor: printf("minor"); break;
    case major: printf("major"); break;
    default: printf("unknown mode"); break;
  }
  putchar('\n');
}
```

4.1.4.12 void settingPoints (int ∗ *mode,* int ∗ *tempo,* int ∗ *length,* int ∗ *octave,* data *data*)

Defineret på linje 243 i filen main.c.

```
                                                                     {
  int deltaTime = deltaTimeToNoteLength(480, 960);
  switch(data.mode){
    case minor: *mode = -5; break;
    case major: *mode = 5; break;
  }
  if(data.tempo < 60)
    *tempo = -5;
  else if(data.tempo >= 60 && data.tempo < 70)
    *tempo = -4;
  else if(data.tempo >= 70 && data.tempo < 80)
    *tempo = -3;
  else if(data.tempo >= 80 && data.tempo < 90)
    *tempo = -2;
  else if(data.tempo >= 90 && data.tempo < 100)
    *tempo = -1;
  else if(data.tempo >= 100 && data.tempo < 120)
    *tempo =  0;
  else if(data.tempo >= 120 && data.tempo < 130)
    *tempo =  1;
  else if(data.tempo >= 130 && data.tempo < 140)
    *tempo =  2;
  else if(data.tempo >= 140 && data.tempo < 150)
    *tempo =  3;
  else if(data.tempo >= 150 && data.tempo < 160)
    *tempo =  4;
  else if(data.tempo >=  160)
    *tempo =  5;

  switch(deltaTime){
    case 1: *length = -5; break;
    case 2: *length = -4; break;
    case 4: *length = -2; break;
    case 8: *length =  0; break;
    case 16: *length = 3; break;
    case 32: *length = 5; break;
  }
}
```

4.1.4.13 int sortResult (const void ∗ *pa,* const void ∗ *pb*)

Defineret på linje 310 i filen main.c.

```
                                              {
    int a = *(const int*)pa;
    int b = *(const int*)pb;
    return (b-a);
}
```

4.1.4.14 int weightingMatrix (moodWeighting *moodArray[ ],* int *mode,* int *tempo,* int *toneLength,* int *pitch*)

Defineret på linje 297 i filen main.c.

```
                                                                                                   {
    int result[AMOUNT_OF_MOODS] = {0};
    for(int i = 0; i < AMOUNT_OF_MOODS; i++){
      result[i] += (moodArray[i].mode * mode);
      result[i] += (moodArray[i].tempo * tempo);
      result[i] += (moodArray[i].toneLength * toneLength);
      result[i] += (moodArray[i].pitch * pitch);
    }
    qsort(result, AMOUNT_OF_MOODS, sizeof(int), sortResult);
    return result[0];
}
```

4.2 test.c filreference

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

Funktioner

- int main (void)
- void testFunk (void)

4.2.1 Funktions-dokumentation

4.2.1.1 int main (void)

Defineret på linje 3 i filen test.c.

```
              {
  printf("Jonas er en kagemand!\nOg han har lange løg.\n");

  return 0;
}
```

4.2.1.2 void testFunk (void)

Defineret på linje 12 i filen test.c.

```
                    {
  int stuff = 1337;
}
```

Indeks