

CPM-AoN

Generated by Doxygen 1.6.1

Sat Jul 29 12:10:16 2017

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	activity Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Data Documentation	5
3.1.2.1	dur	5
3.1.2.2	next	5
3.1.2.3	num	5
3.2	ARC Struct Reference	7
3.2.1	Detailed Description	7
3.2.2	Member Data Documentation	7
3.2.2.1	arcnumber	7
3.2.2.2	bstar	7
3.2.2.3	fromnode	7
3.2.2.4	fstar	7
3.2.2.5	tonode	8
3.3	NODE Struct Reference	9
3.3.1	Detailed Description	9

3.3.2	Member Data Documentation	9
3.3.2.1	bcount	9
3.3.2.2	bstar	9
3.3.2.3	EF	10
3.3.2.4	ES	10
3.3.2.5	fcount	10
3.3.2.6	fstar	10
3.3.2.7	label	10
3.3.2.8	LF	10
3.3.2.9	LS	10
3.3.2.10	nodeDesc	11
3.3.2.11	nodeid	11
3.3.2.12	nodename	11
3.3.2.13	nodenumber	11
3.3.2.14	preds	11
3.3.2.15	rank	11
3.3.2.16	slack	11
3.3.2.17	station	11
3.3.2.18	successors	12
3.3.2.19	tasktime	12
3.3.2.20	unsetpreds	12
3.3.2.21	unsetsuccs	12
3.4	project Class Reference	13
3.4.1	Detailed Description	14
3.4.2	Constructor & Destructor Documentation	14
3.4.2.1	project	14
3.4.3	Member Function Documentation	14
3.4.3.1	AddPreds	14
3.4.3.2	AddSuccessors	15
3.4.3.3	ArcReport	15
3.4.3.4	AssignAllTasks	16

3.4.3.5	AssignmentReport	16
3.4.3.6	Better	17
3.4.3.7	CountPreds	17
3.4.3.8	CountSuccessors	17
3.4.3.9	CPM	18
3.4.3.10	DoNode	19
3.4.3.11	Driver	19
3.4.3.12	Evaluate	19
3.4.3.13	FindTaskFit	20
3.4.3.14	IsTie	21
3.4.3.15	MakeStation	21
3.4.3.16	NewNodeNum	22
3.4.3.17	NodeNum	22
3.4.3.18	NodeReport	22
3.4.3.19	OutNodeLabel	23
3.4.3.20	Printout	23
3.4.3.21	PrintRanking	23
3.4.3.22	PutArc	23
3.4.3.23	PutNode	23
3.4.3.24	RankTasks	23
3.4.3.25	ReadProblem	24
3.4.3.26	ReadProblem2	24
3.4.3.27	SkipRestLine	25
3.4.3.28	ZeroNodeLabels	25
3.4.3.29	ZeroNodePreds	25
3.4.3.30	ZeroNodeSuccessors	26
3.4.4	Member Data Documentation	26
3.4.4.1	arc	26
3.4.4.2	iend	26
3.4.4.3	istart	26
3.4.4.4	min	26

3.4.4.5	n	26
3.4.4.6	node	26
3.4.4.7	nodestack	27
3.4.4.8	numArcs	27
3.4.4.9	numCritical	27
3.4.4.10	numNodes	27
3.4.4.11	numStations	27
3.4.4.12	root	27
3.4.4.13	station	28
3.4.4.14	tMax	28
3.4.4.15	totaltime	28
3.4.4.16	tsort	28
3.5	STATION Struct Reference	29
3.5.1	Detailed Description	29
3.5.2	Member Data Documentation	29
3.5.2.1	numTasks	29
3.5.2.2	stationNum	29
3.5.2.3	stationtime	29
4	File Documentation	31
4.1	aon.cpp File Reference	31
4.1.1	Define Documentation	32
4.1.1.1	debug	32
4.1.1.2	MAX	32
4.1.1.3	MAXARCS	32
4.1.1.4	MAXDESC	32
4.1.1.5	MAXID	32
4.1.1.6	MAXNODES	33
4.1.1.7	NAMELEN	33
4.1.2	Function Documentation	33
4.1.2.1	main	33

Chapter 1

Class Index

1.1 Class List

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

activity	5
ARC	7
NODE	9
project	13
STATION	29

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

[aon.cpp](#) 31

Chapter 3

Class Documentation

3.1 activity Struct Reference

Public Attributes

- long [num](#)
- long [dur](#)
- [activity](#) * [next](#)

3.1.1 Detailed Description

Definition at line 30 of file aon.cpp.

3.1.2 Member Data Documentation

3.1.2.1 long activity::dur

Definition at line 30 of file aon.cpp.

3.1.2.2 activity* activity::next

Definition at line 30 of file aon.cpp.

3.1.2.3 long activity::num

Definition at line 30 of file aon.cpp.

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

- [aon.cpp](#)

3.2 ARC Struct Reference

Public Attributes

- long [arcnumber](#)
- long [fromnode](#)
- long [tonode](#)
- long [fstar](#)
- long [bstar](#)

3.2.1 Detailed Description

Definition at line 55 of file aon.cpp.

3.2.2 Member Data Documentation

3.2.2.1 long ARC::arcnumber

Definition at line 56 of file aon.cpp.

3.2.2.2 long ARC::bstar

Definition at line 60 of file aon.cpp.

Referenced by `project::AddSuccessors()`, `project::DoNode()`, and `project::ReadProblem2()`.

3.2.2.3 long ARC::fromnode

Definition at line 57 of file aon.cpp.

Referenced by `project::CPM()`, and `project::ReadProblem2()`.

3.2.2.4 long ARC::fstar

Definition at line 59 of file aon.cpp.

Referenced by `project::AddPreds()`, `project::MakeStation()`, and `project::ReadProblem2()`.

3.2.2.5 long ARC::tonode

Definition at line 58 of file aon.cpp.

Referenced by project::CPM(), project::MakeStation(), and project::ReadProblem2().

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

- [aon.cpp](#)

3.3 NODE Struct Reference

Public Attributes

- long [nodenumber](#)
- char [nodename](#) [NAMELEN+1]
- long [tasktime](#)
- long [fstar](#)
- long [bstar](#)
- long [fcount](#)
- long [bcount](#)
- long [preds](#)
- long [successors](#)
- long [rank](#)
- long [station](#)
- long [label](#)
- long [ES](#)
- long [EF](#)
- long [LS](#)
- long [LF](#)
- long [slack](#)
- long [unsetpreds](#)
- long [unsetsuccs](#)
- char [nodeid](#) [MAXID+1]
- char [nodeDesc](#) [MAXDESC+1]

3.3.1 Detailed Description

Definition at line 31 of file aon.cpp.

3.3.2 Member Data Documentation

3.3.2.1 long NODE::bcount

Definition at line 38 of file aon.cpp.

Referenced by `project::CPM()`, and `project::ReadProblem2()`.

3.3.2.2 long NODE::bstar

Definition at line 36 of file aon.cpp.

Referenced by `project::AddSuccessors()`, `project::CPM()`, `project::DoNode()`, `project::Evaluate()`, and `project::ReadProblem2()`.

3.3.2.3 long NODE::EF

Definition at line 45 of file aon.cpp.

Referenced by project::CPM().

3.3.2.4 long NODE::ES

Definition at line 44 of file aon.cpp.

Referenced by project::CPM(), and project::NewNodeNum().

3.3.2.5 long NODE::fcount

Definition at line 37 of file aon.cpp.

Referenced by project::CPM(), and project::ReadProblem2().

3.3.2.6 long NODE::fstar

Definition at line 35 of file aon.cpp.

Referenced by project::AddPreds(), project::CPM(), project::Evaluate(), project::MakeStation(), and project::ReadProblem2().

3.3.2.7 long NODE::label

Definition at line 43 of file aon.cpp.

Referenced by project::AddPreds(), project::AddSuccessors(), project::Evaluate(), and project::MakeStation().

3.3.2.8 long NODE::LF

Definition at line 47 of file aon.cpp.

Referenced by project::CPM().

3.3.2.9 long NODE::LS

Definition at line 46 of file aon.cpp.

Referenced by project::CPM(), and project::NewNodeNum().

3.3.2.10 char NODE::nodeDesc[MAXDESC+1]

Definition at line 52 of file aon.cpp.

3.3.2.11 char NODE::nodeid[MAXID+1]

Definition at line 51 of file aon.cpp.

3.3.2.12 char NODE::nodename[NAMELEN+1]

Definition at line 33 of file aon.cpp.

Referenced by project::MakeStation(), project::NewNodeNum(), and project::PrintRanking().

3.3.2.13 long NODE::nodenumber

Definition at line 32 of file aon.cpp.

3.3.2.14 long NODE::preds

Definition at line 39 of file aon.cpp.

Referenced by project::AddPreds().

3.3.2.15 long NODE::rank

Definition at line 41 of file aon.cpp.

3.3.2.16 long NODE::slack

Definition at line 48 of file aon.cpp.

Referenced by project::CPM().

3.3.2.17 long NODE::station

Definition at line 42 of file aon.cpp.

Referenced by project::MakeStation().

3.3.2.18 long NODE::successors

Definition at line 40 of file aon.cpp.

Referenced by project::AddSuccessors().

3.3.2.19 long NODE::tasktime

Definition at line 34 of file aon.cpp.

Referenced by project::CPM(), project::MakeStation(), and project::ReadProblem2().

3.3.2.20 long NODE::unsetpreds

Definition at line 49 of file aon.cpp.

Referenced by project::CPM().

3.3.2.21 long NODE::unsetsuccs

Definition at line 50 of file aon.cpp.

Referenced by project::CPM().

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

- [aon.cpp](#)

3.4 project Class Reference

Public Member Functions

- [project](#) ()
- void [ReadProblem](#) (const char *fname)
- void [ReadProblem2](#) (const char *fname)
- void [AssignAllTasks](#) (long, long)
- void [AssignmentReport](#) (long, long)
- long [NodeNum](#) (char *)
- long [NewNodeNum](#) (char *)
- long [MakeStation](#) (long, long)
- long [IsTie](#) (long, long)
- long [FindTaskFit](#) (long)
- void [RankTasks](#) (long)
- long [Better](#) (long, long, long)
- void [PrintRanking](#) (long)
- void [CountPreds](#) ()
- void [CountSuccessors](#) ()
- void [AddPreds](#) (long)
- void [AddSuccessors](#) (long)
- void [ZeroNodeLabels](#) ()
- void [ZeroNodePreds](#) ()
- void [ZeroNodeSuccessors](#) ()
- void [Printout](#) ()
- void [NodeReport](#) ()
- void [ArcReport](#) ()
- void [Evaluate](#) ()
- void [DoNode](#) (long, string)
- void [PutNode](#) (long)
- void [OutNodeLabel](#) (long)
- void [PutArc](#) (long, string, long)
- void [Driver](#) (long, char **)
- void [CPM](#) ()

Public Attributes

- long [min](#)
- long [n](#)

Private Member Functions

- void [SkipRestLine](#) (ifstream &file)

Private Attributes

- long [tMax](#)
- long [numNodes](#)
- long [numArcs](#)
- long [numStations](#)
- long [numCritical](#)
- long [root](#)
- long [totaltime](#)
- long [istart](#)
- long [iend](#)
- [NODE](#) [node](#) [MAXNODES+1]
- [ARC](#) [arc](#) [MAXARCS+1]
- [STATION](#) [station](#) [MAXNODES+1]
- long [tsort](#) [MAXNODES+1]
- stack< long > [nodestack](#)

3.4.1 Detailed Description

Definition at line 69 of file aon.cpp.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 `project::project () [inline]`

Definition at line 71 of file aon.cpp.

```
71 {tMax = 0; numNodes=numArcs=0;}
```

3.4.3 Member Function Documentation

3.4.3.1 `void project::AddPreds (long i)`

Definition at line 374 of file aon.cpp.

Referenced by `CountPreds()`.

```

374         {
375         long a;
376         if (node[i].label) return;
377         ++node[i].preds;
378         node[i].label = 1;
379         a = node[i].fstar;
380         while(a) {
381             AddPreds(arc[a].tonode);
382             a = arc[a].fstar;
383         }
384         return;
385     }

```

3.4.3.2 void project::AddSuccessors (long i)

Definition at line 406 of file aon.cpp.

Referenced by CountSuccessors().

```

406         {
407         long a;
408         if (node[i].label) return;
409         ++node[i].successors;
410         node[i].label = 1;
411         a = node[i].bstar;
412         while(a) {
413             AddSuccessors(arc[a].fromnode);
414             a = arc[a].bstar;
415         }
416         return;
417     }

```

3.4.3.3 void project::ArcReport ()

Definition at line 473 of file aon.cpp.

```

473         {
474         long i;
475
476         cout << endl << " *** TASK PRECEDENCE REQUIREMENTS ***" << endl << endl;
477         printf(" Arc From To Fstar Bstar\n");
478         printf("-----\n");
479
480         for(i=1;i<=numArcs; ++i) {
481             printf("%3d %3d %3d %5d %5d\n",
482                 i, arc[i].fromnode, arc[i].tonode, arc[i].fstar, arc[i].bstar);
483         }
484     }

```

3.4.3.4 void project::AssignAllTasks (long *ct*, long *rule*)

Definition at line 153 of file aon.cpp.

```

153                                     {
154     long i, tasksleft;
155     cout << endl << "   *** WORKSTATION ASSIGNMENTS ***" << endl << endl;
156     cout << "For cycle time = " << ct << ", using rule " << rule << endl ;
157     if (ct < tMax) {
158         cout << "Infeasible cycle time, reset to " << tMax << ", the larg
159         est task time"<< endl;
160         ct = tMax;
161     }
162     cout << endl;
163     for (i=1; i <= numNodes; ++i) node[i].label = node[i].bcount;
164     for (numStations=0,tasksleft=numNodes; numStations <= numNodes && tasksle
165     ft; ) {
166         ++numStations;
167         MakeStation(numStations,ct);
168         tasksleft -= station[numStations].numTasks;
169     }
170 }
```

3.4.3.5 void project::AssignmentReport (long *ct*, long *rule*)

Definition at line 121 of file aon.cpp.

```

121                                     {
122     long i, idle, calc;
123     double e;
124     cout << endl << "   *** ASSEMBLY LINE SUMMARY ***" << endl << endl;
125
126     printf("Problem:\n");
127     printf("-----\n");
128     printf("Number of tasks          %d\n",numNodes);
129     printf("Total task time             %d\n",totaltime);
130     printf("Longest task time           %d\n",tMax);
131     printf("Target cycle time          %d\n\n",ct);
132
133     printf("Assignment:\n");
134     printf("-----\n");
135     printf("Rule                          %d\n",rule);
136     for(i=1, calc=0; i <= numStations; ++i) {calc = MAX(calc,station[i].stati
137     ontime);}
138     ct = calc;
139     printf("Computed cycle time      %d\n",ct);
140     printf("Stations used, N           %d\n",numStations);
141     idle = 0;
142     for(i=1; i <= numStations; ++i) {idle += (ct-station[i].stationtime);}
143     printf("Output per hour, N'      %.2lf (if seconds)\n",3600.0/(double)ct);
```

```

144         printf("                %.2lf (if minutes)\n",60.0/(double)ct);
145         printf("Station time/cycle    %d\n",numStations*ct);
146         printf("Idle time/cycle      %d\n",idle);
147         e = (double)totaltime/((double) numStations*(double)ct);
148         printf("Efficiency, E        %lf\n",e);
149         printf("Fraction idle, 1-E    %lf\n",1.0-e);
150         printf("Balance delay        %lf\n",100.0*(1.0-e));
151     }

```

3.4.3.6 long project::Better (long j, long i, long rule)

Definition at line 244 of file aon.cpp.

Referenced by RankTasks().

```

244                                     { // returns i or j, whichever is b
    etter
245         switch (rule) {
246             case 1: if (node[j].preds > node[i].preds) return 0;
247                     else if (node[j].preds < node[i].preds) return 1;
248
249                     break;
250             case 2: if (node[j].successors < node[i].successors) return 0;
251                     else if (node[j].successors > node[i].successors)
252                         return 1;
253                     break;
254             }
255         // Tie-breaker: longer task time first
256         if (node[j].tasktime > node[i].tasktime) return 1;
257         else return 0;
258     }

```

3.4.3.7 void project::CountPreds ()

Definition at line 364 of file aon.cpp.

```

364         {
365             long i;
366             ZeroNodePreds();
367             for (i=1; i <= numNodes; ++i) {
368                 ZeroNodeLabels();
369                 AddPreds(i);
370             }
371             return;
372         }

```

3.4.3.8 void project::CountSuccessors ()

Definition at line 396 of file aon.cpp.

```

396         {
397         long i;
398         ZeroNodeSuccessors();
399         for (i=1; i <= numNodes; ++i) {
400             ZeroNodeLabels();
401             AddSuccessors(i);
402         }
403         return;
404     }

```

3.4.3.9 void project::CPM ()

Definition at line 507 of file aon.cpp.

```

508     {
509     // Forward pass
510     long i,j,n,EFi,LSi;
511
512     for (i=1; i < numNodes; ++i) {
513         node[i].unsetpreds = node[i].bcount;
514         node[i].unsetsuccs = node[i].fcount;
515     }
516
517     node[istart].ES = 0;
518     nodestack.push(istart);
519     do {
520         i = nodestack.top();
521         nodestack.pop();
522         EFi = node[i].EF = node[i].ES + node[i].tasktime;
523         // Explore fstar list for node i and add complete ones to stack
524         if ((n = node[i].fcount) > 0) {
525             j = arc[node[i].fstar].tonode; // next successor
526             for(; n > 0; --n){
527                 if(EFi > node[j].EF) node[j].EF = EFi;
528                 if((--node[j].unsetpreds) == 0) nodestack.push(j)
529             }
530             j = arc[node[j].fstar].tonode; // next successor
531         }
532     } while(!nodestack.empty());
533
534     // Backward pass
535
536     node[iend].LF = node[iend].EF;
537     nodestack.push(iend);
538     do {
539         i = nodestack.top();
540         nodestack.pop();
541         LSi = node[i].LS = node[i].LF - node[i].tasktime;
542         if((n = node[i].bcount) > 0) {
543             j = arc[node[i].bstar].fromnode; // next predecessor n
544         }
545         for (; n > 0; --n) {
546             if(LSi < node[j].LS) node[j].LS = LSi; // new m

```

```

    in LSj
546                                     if((--node[j].unsetsuccs) == 0) nodestack.push(j)
    ;
547                                     j = arc[node[j].bstar].fromnode;           // next p
    red in bstar list
548                                     }
549                                     }
550                                } while(!nodestack.empty());
551
552 //    Compute slacks
553
554    for(i=1,numCritical=0; i<numNodes; ++i) {
555        node[i].slack = node[i].EF - node[i].ES;
556        if(node[i].slack == 0) ++numCritical;
557    }
558

```

3.4.3.10 void project::DoNode (long i, string tab1)

Definition at line 274 of file aon.cpp.

```

276    {
277        long a,j,lv1,t;
278        string postfix;
279
280        a = node[i].bstar;
281        t = 0;
282        while (a) {
283            PutArc(a,tab1,t);
284            DoNode(arc[a].tonode,postfix);
285            a = arc[a].bstar;
286            t = 1;
287        }
288        cout << tab1 << endl;
289        return;
290    }

```

3.4.3.11 void project::Driver (long, char **)

Referenced by main().

3.4.3.12 void project::Evaluate ()

Definition at line 423 of file aon.cpp.

```

423    {
424        long i;
425        long queue[MAXNODES+1];
426        long qlen;

```

```

427         long evnode; // node being evaluated
428         long evarc;
429         long pred;
430         long evcount;
431
432         qlen = 0;
433         for(i = 1; i <= numNodes; ++i) {
434             }
435
436         evcount = numNodes;
437         while (qlen > 0) {
438             evnode = queue[qlen--];
439             --evcount;
440             evarc = node[evnode].fstar;
441             pred = node[evnode].bstar;
442             --node[pred].label;
443             if (debug) {cout << "Evaluating node " << evnode << "   Q = {";
444                 for (i=1; i<= qlen; ++i) cout << queue[i] << " ";
445                 cout << "}" << endl;
446             }
447
448             node[pred].bstar = evarc; // add to chain of child arcs
449         for pred node
450             if ((node[pred].label) <= 0) queue[++qlen] = pred;
451         }
452         if (evcount) cout << "*** There are " << evcount << " unevaluated nodes" <
453         < endl;
454         root = evnode;
455     }

```

3.4.3.13 long project::FindTaskFit (long *timeleft*)

Definition at line 202 of file aon.cpp.

Referenced by MakeStation().

```

202                                     { // Return next task that fits or 0
203         long i,t;
204         for (i=1; i <= numNodes; ++i) {
205             t = tsort[i];
206             if (t <= 0) continue; // skip if already assigned
207             if (node[t].label) continue; // skip if all preds not assigned
208             if (node[t].tasktime <= timeleft) {
209                 tsort[i] = -tsort[i];
210                 return t;
211             }
212         }
213         return 0;
214     }

```

3.4.3.14 long project::IsTie (long *i*, long *j*)

Definition at line 228 of file aon.cpp.

Referenced by PrintRanking().

```

228                                     {
229         if(i <1 || i > numNodes || j < 1 || j > numNodes) return 0;
230         if (node[i].preds == node[j].preds && node[i].tasktime == node[j].tasktim
    e) return 1;
231         else return 0;
232     }

```

3.4.3.15 long project::MakeStation (long *s*, long *cycletime*)

Definition at line 172 of file aon.cpp.

Referenced by AssignAllTasks().

```

172                                     {          // assign tasks to worksta
    tion s, return # assigned
173         long t,a,timeleft,cumtime;
174         static STATION initstation = {0,0,0};
175
176         timeleft = cycletime;
177         cumtime = 0;
178         station[s] = initstation;
179         // NodeReport();
180         // ArcReport();
181
182         // cout << endl<<"Station " << s << ":" << endl;
183         // cout << "Task   Time   CumTime" << endl;
184         // cout << "----   ----   ----" << endl;
185         cout << "Station " << s << " tasks:" ;
186
187         while ((timeleft > 0) && (t=FindTaskFit(timeleft)) > 0) {
188             node[t].station = s;
189             ++station[s].numTasks;
190             station[s].stationtime += node[t].tasktime;
191             timeleft -= node[t].tasktime;
192             // printf("%4d%6d%9d\n", t, node[t].tasktime, station[s].stationt
    ime);
193             cout << " " << node[t].nodename;
194             a = node[t].fstar;
195             while(a) { --node[arc[a].tonode].label; a = arc[a].fstar;}
196             }
197
198         cout << " (time = " << station[s].stationtime << ")" << endl;
199         // cout << "-----" << endl;
200     }

```

3.4.3.16 long project::NewNodeNum (char * s)

Definition at line 305 of file aon.cpp.

Referenced by NodeNum().

```

305                                     {
306         ++numNodes;
307         memcpy (node[numNodes].nodename, s, sizeof(char)*(NAMELEN));
308         node[numNodes].nodename[NAMELEN+1] = '\0';
309         node[numNodes].ES      = 0;
310         node[numNodes].LS      = LONG_MAX;
311         return numNodes;
312     }
```

3.4.3.17 long project::NodeNum (char * s)

Definition at line 299 of file aon.cpp.

Referenced by ReadProblem2().

```

299                                     {
300         if (numNodes == 0) return NewNodeNum(s);
301         for (long i=1; i <= numNodes; ++i) { if (!strcmp(s,node[i].nodename,
302         NAMELEN)) return i; }
303         return NewNodeNum(s);
304     }
```

3.4.3.18 void project::NodeReport ()

Definition at line 456 of file aon.cpp.

```

456                                     {
457     long i;
458     char buf[25];
459
460     cout <<endl<<"      *** TASK SUMMARY ***"<< endl << endl ;
461     cout << "      Task    Time    Preds Succs";
462     if (debug) cout << "Fstar Bstar    Fcount Bcount    Label";
463     cout << endl;
464     for(i=1;i<=numNodes; ++i) {
465         printf("%8.8s    %4d    %5d %5d ",node[i].nodename,node[i].tasktime
466         , node[i].preds, node[i].successors);
467         if (debug) printf("%    %5d %5d    %5d",
468         node[i].fstar, node[i].bstar, node[i].fcount, node[i].bco
469         unt, node[i].label);
470         cout << endl;
471     }
```

3.4.3.19 void project::OutNodeLabel (long *i*)

Definition at line 258 of file aon.cpp.

```

258                                     {
259     char buf[20];
260
261     cout << i;
262 }
```

3.4.3.20 void project::Printout ()**3.4.3.21 void project::PrintRanking (long *rule*)**

Definition at line 216 of file aon.cpp.

```

216                                     {
217     long ties = 0;
218     cout << "Tasks ranked by rule " << rule << ":" << endl;
219     for (long i=1; i <= numNodes; ++i) {
220         cout << " " << node[tsort[i]].nodename;
221         if (IsTie(tsort[i],tsort[i-1])) {++ties; cout << "*"; }
222         // if (IsTie(tsort[i],tsort[i-1]) || IsTie(tsort[i],tsort[i+1]))
223         {++ties; cout << "*"; }
224     }
225     cout << endl;
226     if (ties) cout << " (*task tied with prior task in ranking)" << endl;
227 }
```

3.4.3.22 void project::PutArc (long *a*, string *tab*, long *usetab*)

Definition at line 264 of file aon.cpp.

Referenced by DoNode().

```

266     {
267     const char* cp;
268     char buf[10];
269
270     cp = tab.c_str();
271     return;
272 }
```

3.4.3.23 void project::PutNode (long)**3.4.3.24 void project::RankTasks (long *rule*)**

Definition at line 234 of file aon.cpp.

```

234                                     {
235         long i,j,best,hold;
236         for (i=1; i <= numNodes; ++i) tsort[i] = i;
237         for (i=1; i < numNodes; ++i) {
238             best = i;
239             for (j=i+1; j <= numNodes; ++j) { if(Better(tsort[j],tsort[best],
rule)) best = j; }
240             if (best != i) {hold = tsort[i]; tsort[i]=tsort[best]; tsort[best
] = hold;} // Swap
241             }
242         }

```

3.4.3.25 void project::ReadProblem (const char *fname)

3.4.3.26 void project::ReadProblem2 (const char *fname)

Definition at line 314 of file aon.cpp.

```

315 {         long ifrom, jto, tasktime;
316         int nstart, nend;    // number of start/end nodes (with 0 preds/succs)
317         char buf[256], save[256];
318         char *token;
319
320         ifstream file;
321         file.open(fname, ios_base::in);
322         if (file.fail()) { cout << "Cannot open input file.\n"; exit(1); }
323
324         for (numArcs=0;;)
325         {
326             file.getline (buf,256);
327             if (file.eof()) break;
328             memcpy(save,buf,sizeof(char)*256);
329
330             token = strtok(buf," \t");    // Node/task name
331             jto = NodeNum(token);
332
333             token = strtok(NULL," \t");
334             node[jto].tasktime = atoi(token);
335             if (node[jto].tasktime > tMax) tMax = node[jto].tasktime;
336             totaltime += node[jto].tasktime;
337
338             while (token = strtok(NULL," \t") ) {
339                 ifrom = NodeNum(token);
340                 ++numArcs;
341                 arc[numArcs].fromnode = ifrom;
342                 arc[numArcs].tonode    = jto;
343                 arc[numArcs].bstar     = node[jto].bstar;
344                 node[jto].bstar        = numArcs;
345                 arc[numArcs].fstar     = node[ifrom].fstar;
346                 node[ifrom].fstar      = numArcs;
347                 ++node[ifrom].fcount;
348                 ++node[jto].bcount;
349             }
350             for (int i=1, nstart=nend=0; i <= numNodes; ++i) {

```

```

351             if (node[i].preds == 0) { istart = i; ++nstart;}
352             if (node[i].successors == 0) { iend = i; ++nend;}
353         }
354     }
355 }
356
357 file.close();
358 cout << numNodes << " task nodes, " << numArcs << " precedence arcs";
359 cout << ", and " << totaltime << " time units for all tasks" << endl;
360 if(istart != 1) { cout << "Exactly 1 start node (with no predecessors) is requ
    ired.\n"; exit(1); }
361 if(iend != 1) { cout << "Exactly 1 end node (with no successors) is required.\
    n"; exit(1); }
362 }

```

3.4.3.27 void project::SkipRestLine (ifstream &file) [private]

Definition at line 293 of file aon.cpp.

```

294 { char ch;
295     do file.get(ch); while (!file.fail() && ch != '\n');
296 }

```

3.4.3.28 void project::ZeroNodeLabels ()

Definition at line 387 of file aon.cpp.

Referenced by CountPreds(), and CountSuccessors().

```

387     {
388         for(long i=1; i<= numNodes; ++i) node[i].label = 0;
389     }

```

3.4.3.29 void project::ZeroNodePreds ()

Definition at line 391 of file aon.cpp.

Referenced by CountPreds().

```

391     {
392         for(long i=1; i<= numNodes; ++i) node[i].preds = -1;
393     }

```

3.4.3.30 void project::ZeroNodeSuccessors ()

Definition at line 419 of file aon.cpp.

Referenced by CountSuccessors().

```
419                                     {  
420         for(long i=1; i<= numNodes; ++i) node[i].successors = -1;  
421     }
```

3.4.4 Member Data Documentation

3.4.4.1 ARC project::arc[MAXARCS+1] [private]

Definition at line 113 of file aon.cpp.

Referenced by AddPreds(), AddSuccessors(), ArcReport(), CPM(), DoNode(), MakeStation(), and ReadProblem2().

3.4.4.2 long project::iend [private]

Definition at line 111 of file aon.cpp.

Referenced by CPM(), and ReadProblem2().

3.4.4.3 long project::istart [private]

Definition at line 110 of file aon.cpp.

Referenced by CPM(), and ReadProblem2().

3.4.4.4 long project::min

Definition at line 72 of file aon.cpp.

3.4.4.5 long project::n

Definition at line 72 of file aon.cpp.

Referenced by CPM().

3.4.4.6 NODE project::node[MAXNODES+1] [private]

Definition at line 112 of file aon.cpp.

Referenced by AddPreds(), AddSuccessors(), AssignAllTasks(), Better(), CPM(), DoNode(), Evaluate(), FindTaskFit(), IsTie(), MakeStation(), NewNodeNum(), NodeNum(), NodeReport(), PrintRanking(), ReadProblem2(), ZeroNodeLabels(), ZeroNodePreds(), and ZeroNodeSuccessors().

3.4.4.7 `stack<long> project::nodestack [private]`

Definition at line 117 of file aon.cpp.

Referenced by CPM().

3.4.4.8 `long project::numArcs [private]`

Definition at line 105 of file aon.cpp.

Referenced by ArcReport(), project(), and ReadProblem2().

3.4.4.9 `long project::numCritical [private]`

Definition at line 107 of file aon.cpp.

Referenced by CPM().

3.4.4.10 `long project::numNodes [private]`

Definition at line 104 of file aon.cpp.

Referenced by AssignAllTasks(), AssignmentReport(), CountPreds(), CountSuccessors(), CPM(), Evaluate(), FindTaskFit(), IsTie(), NewNodeNum(), NodeNum(), NodeReport(), PrintRanking(), project(), RankTasks(), ReadProblem2(), ZeroNodeLabels(), ZeroNodePreds(), and ZeroNodeSuccessors().

3.4.4.11 `long project::numStations [private]`

Definition at line 106 of file aon.cpp.

Referenced by AssignAllTasks(), and AssignmentReport().

3.4.4.12 `long project::root [private]`

Definition at line 108 of file aon.cpp.

Referenced by Evaluate().

3.4.4.13 STATION project::station[MAXNODES+1] [private]

Definition at line 114 of file aon.cpp.

Referenced by AssignAllTasks(), AssignmentReport(), and MakeStation().

3.4.4.14 long project::tMax [private]

Definition at line 103 of file aon.cpp.

Referenced by AssignAllTasks(), AssignmentReport(), project(), and ReadProblem2().

3.4.4.15 long project::totaltime [private]

Definition at line 109 of file aon.cpp.

Referenced by AssignmentReport(), and ReadProblem2().

3.4.4.16 long project::tsort[MAXNODES+1] [private]

Definition at line 115 of file aon.cpp.

Referenced by FindTaskFit(), PrintRanking(), and RankTasks().

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

- [aon.cpp](#)

3.5 STATION Struct Reference

Public Attributes

- long [stationtime](#)
- long [numTasks](#)
- long [stationNum](#)

3.5.1 Detailed Description

Definition at line 63 of file aon.cpp.

3.5.2 Member Data Documentation

3.5.2.1 long STATION::numTasks

Definition at line 65 of file aon.cpp.

Referenced by `project::AssignAllTasks()`, and `project::MakeStation()`.

3.5.2.2 long STATION::stationNum

Definition at line 66 of file aon.cpp.

3.5.2.3 long STATION::stationtime

Definition at line 64 of file aon.cpp.

Referenced by `project::AssignmentReport()`, and `project::MakeStation()`.

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

- [aon.cpp](#)

Chapter 4

File Documentation

4.1 aon.cpp File Reference

```
#include <fstream>
#include "stdio.h"
#include <cmath>
#include <iostream>
#include <sstream>
#include <climits>
#include <stack>
#include <string>
#include <math.h>
#include <cstring>
#include <cstdlib>
```

Classes

- struct [activity](#)
- struct [NODE](#)
- struct [ARC](#)
- struct [STATION](#)
- class [project](#)

Defines

- #define `MAX(AA, BB)` (((AA)>(BB))?(AA):(BB))
- #define `MAXNODES` 100
- #define `MAXARCS` 500
- #define `MAXDESC` 50
- #define `MAXID` 4
- #define `NAMELEN` 8
- #define `debug` 0

Functions

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

4.1.1 Define Documentation

4.1.1.1 #define debug 0

Definition at line 25 of file aon.cpp.

Referenced by project::Evaluate(), and project::NodeReport().

4.1.1.2 #define MAX(AA, BB) (((AA)>(BB))?(AA):(BB))

Definition at line 19 of file aon.cpp.

Referenced by project::AssignmentReport().

4.1.1.3 #define MAXARCS 500

Definition at line 21 of file aon.cpp.

4.1.1.4 #define MAXDESC 50

Definition at line 22 of file aon.cpp.

4.1.1.5 #define MAXID 4

Definition at line 23 of file aon.cpp.

4.1.1.6 #define MAXNODES 100

Definition at line 20 of file aon.cpp.

Referenced by project::Evaluate().

4.1.1.7 #define NAMELEN 8

Definition at line 24 of file aon.cpp.

Referenced by project::NewNodeNum(), and project::NodeNum().

4.1.2 Function Documentation

4.1.2.1 int main (int *argc*, char * *argv*[])

Definition at line 560 of file aon.cpp.

```
561 { project g;  
562  
563     g.Driver(argc,argv);  
564     return 0;  
565 }
```

Index

- activity, [5](#)
 - dur, [5](#)
 - next, [5](#)
 - num, [5](#)
- AddPreds
 - project, [14](#)
- AddSuccessors
 - project, [15](#)
- aon.cpp, [31](#)
 - debug, [32](#)
 - main, [33](#)
 - MAX, [32](#)
 - MAXARCS, [32](#)
 - MAXDESC, [32](#)
 - MAXID, [32](#)
 - MAXNODES, [32](#)
 - NAMELEN, [33](#)
- ARC, [7](#)
 - arcnumber, [7](#)
 - bstar, [7](#)
 - fromnode, [7](#)
 - fstar, [7](#)
 - tonode, [7](#)
- arc
 - project, [26](#)
- arcnumber
 - ARC, [7](#)
- ArcReport
 - project, [15](#)
- AssignAllTasks
 - project, [15](#)
- AssignmentReport
 - project, [16](#)
- bcount
 - NODE, [9](#)
- Better
 - project, [17](#)
- bstar
 - ARC, [7](#)
 - NODE, [9](#)
- CountPreds
 - project, [17](#)
- CountSuccessors
 - project, [17](#)
- CPM
 - project, [18](#)
- debug
 - aon.cpp, [32](#)
- DoNode
 - project, [19](#)
- Driver
 - project, [19](#)
- dur
 - activity, [5](#)
- EF
 - NODE, [9](#)
- ES
 - NODE, [10](#)
- Evaluate
 - project, [19](#)
- fcounat
 - NODE, [10](#)
- FindTaskFit
 - project, [20](#)
- fromnode
 - ARC, [7](#)
- fstar
 - ARC, [7](#)
 - NODE, [10](#)

- iend
 - project, 26
- istart
 - project, 26
- IsTie
 - project, 20
- label
 - NODE, 10
- LF
 - NODE, 10
- LS
 - NODE, 10
- main
 - aon.cpp, 33
- MakeStation
 - project, 21
- MAX
 - aon.cpp, 32
- MAXARCS
 - aon.cpp, 32
- MAXDESC
 - aon.cpp, 32
- MAXID
 - aon.cpp, 32
- MAXNODES
 - aon.cpp, 32
- min
 - project, 26
- n
 - project, 26
- NAMELEN
 - aon.cpp, 33
- NewNodeNum
 - project, 21
- next
 - activity, 5
- NODE, 9
 - bcount, 9
 - bstar, 9
 - EF, 9
 - ES, 10
 - fcount, 10
 - fstar, 10
 - label, 10
 - LF, 10
 - LS, 10
 - nodeDesc, 10
 - nodeid, 11
 - nodename, 11
 - nodenumber, 11
 - preds, 11
 - rank, 11
 - slack, 11
 - station, 11
 - successors, 11
 - tasktime, 12
 - unsetpreds, 12
 - unsetsuccs, 12
- node
 - project, 26
- nodeDesc
 - NODE, 10
- nodeid
 - NODE, 11
- nodename
 - NODE, 11
- NodeNum
 - project, 22
- nodenumber
 - NODE, 11
- NodeReport
 - project, 22
- nodestack
 - project, 27
- num
 - activity, 5
- numArcs
 - project, 27
- numCritical
 - project, 27
- numNodes
 - project, 27
- numStations
 - project, 27
- numTasks
 - STATION, 29
- OutNodeLabel
 - project, 22

- preds
 - NODE, 11
- Printout
 - project, 23
- PrintRanking
 - project, 23
- project, 13
 - AddPreds, 14
 - AddSuccessors, 15
 - arc, 26
 - ArcReport, 15
 - AssignAllTasks, 15
 - AssignmentReport, 16
 - Better, 17
 - CountPreds, 17
 - CountSuccessors, 17
 - CPM, 18
 - DoNode, 19
 - Driver, 19
 - Evaluate, 19
 - FindTaskFit, 20
 - iend, 26
 - istart, 26
 - IsTie, 20
 - MakeStation, 21
 - min, 26
 - n, 26
 - NewNodeNum, 21
 - node, 26
 - NodeNum, 22
 - NodeReport, 22
 - nodestack, 27
 - numArcs, 27
 - numCritical, 27
 - numNodes, 27
 - numStations, 27
 - OutNodeLabel, 22
 - Printout, 23
 - PrintRanking, 23
 - project, 14
 - PutArc, 23
 - PutNode, 23
 - RankTasks, 23
 - ReadProblem, 24
 - ReadProblem2, 24
 - root, 27
 - SkipRestLine, 25
 - station, 27
 - tMax, 28
 - totaltime, 28
 - tsort, 28
 - ZeroNodeLabels, 25
 - ZeroNodePreds, 25
 - ZeroNodeSuccessors, 25
- PutArc
 - project, 23
- PutNode
 - project, 23
- rank
 - NODE, 11
- RankTasks
 - project, 23
- ReadProblem
 - project, 24
- ReadProblem2
 - project, 24
- root
 - project, 27
- SkipRestLine
 - project, 25
- slack
 - NODE, 11
- STATION, 29
 - numTasks, 29
 - stationNum, 29
 - stationtime, 29
- station
 - NODE, 11
 - project, 27
- stationNum
 - STATION, 29
- stationtime
 - STATION, 29
- successors
 - NODE, 11
- tasktime
 - NODE, 12
- tMax
 - project, 28

tonode
 ARC, [7](#)
totaltime
 project, [28](#)
tsort
 project, [28](#)

unsetpreds
 NODE, [12](#)
unsetsuccs
 NODE, [12](#)

ZeroNodeLabels
 project, [25](#)
ZeroNodePreds
 project, [25](#)
ZeroNodeSuccessors
 project, [25](#)