平成21年度
北九州市立大学特別研究推進費対象研究

Webによる卒業単位チェックシステムの開発

松尾太加志

平成22年4月1日

目 次

1 はじめに

大学のカリキュラムはさまざまな科目区分に分かれ，卒業に必要な要件は科目区分ごとの必要単位数だけではなく，複数の科目区分に渡った要件が設定されており，卒業要件の計算が複雑になっていることが多い．そのため，学生が計算を間違ってしまい卒業できなくなるケースもある．また，卒業できた学生であっても，卒業できるまでは，卒業に必要な単位を満たしているのかどうか不安を感じている学生も多い．

学務情報システム等が完備されている大学においては，不足単位数を示した成績表を配布したり，コンピュータ画面上で確認できるようになっているところもある．ただし，要件の条件が比較的単純な場合には科目区分ごとの不足単位数を示すことができるが，少し複雑になると的確に不足単位数を示すことができない．また，学務情報システムの導入に費用がかかるため，すべての大学においてこのようなシステムが利用できるわけではない．そのため，卒業要件のチェック確認表を配布するなどの対応のところも少なくない．しかし，確認表も履修便覧の記載内容をわかりやすく工夫しただけで，自分でチェックをすることに変わりはなく，間違いが生じる可能性はある．

そこで本研究では，コストがかからない汎用的な卒業単位チェックシステムの開発を検討した．学務情報システムは，履修成績や履修登録と連動させることによって有用なシステムとなっており，さらにコース管理システムとの連携が実現されている（梶田・角所・中澤・竹村・美濃・間瀬，2007）．しかし，カリキュラムに合わせたカスタマイズの必要性，成績と連動するためのセキュリティの確保（吉牟田，2004）の問題などが生じる．そのため，導入コストや運用コストは，金銭面だけではなく，システム構築や保守運用のための人的資源においてもかなり必要となる．

ところが，卒業の単位チェックだけに限れば，学生が自分の成績表を見ながら，単位数を入力すると，卒業判定ができるようなシステムとすることによって，学生のニーズは十分に満たすことができる．その場合，卒業要件の条件だけを設計すればよいためコストはあまりかからない上に，成績との連動もないため，セキュリティ上の問題も考慮する必要がない．したがって，

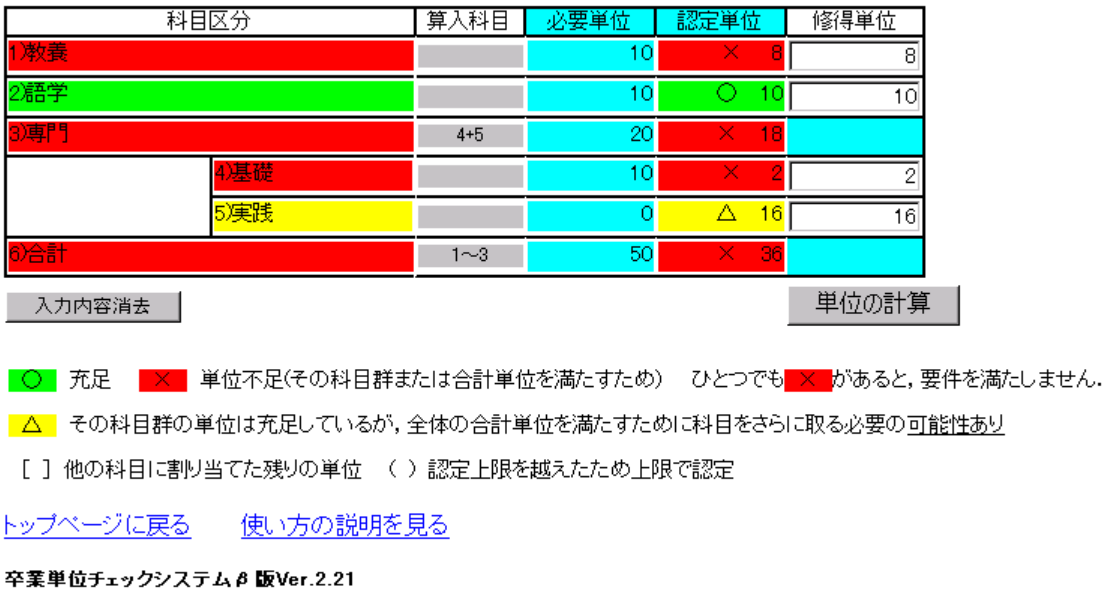


図 1. 卒業単位チェックシステムの画面例．

Web 上での構築も容易で，誰でも利用できるようにすることが可能である．

同様の単位チェックシステムはいくつかみられるが（栗原，2007; 岩下・片桐・ 天田・ 南・中居，2006），汎用的なものは存在しない．大学の卒業単位要件は，科目区分ごとに必要な単位数が設定されるという点においてはどの大学においても共通であるため，アルゴリズムの工夫によって，汎用的な卒業単位チェックシステムの開発は可能であり，本研究では，それを実現することが目的である．

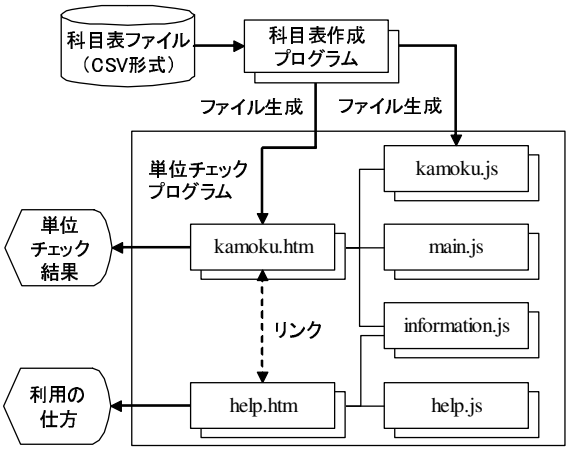


図 2. システムのプログラム構成図

2 システム概要

図 1 に示したものが，実際に完成した卒業単位チェックシステムの画面例である．学生は，Web 上で自分が取得した単位を科目区分ごとに入力をする．そして「単位の計算」のボタンをクリックすると，科目区分での合計，全体の合計等を算出し，科目区分ごとに判定を行ってくれる．その判定は 3 段階（○緑判定，×赤判定，△黄判定）で，○は新たな取得不要，×は追加取得が必要，△は他の区分で取らなければ追加取得が必要であることを示している（詳細は後述）．学生はひと目でどの科目区分の単位が不足しているのか知ることができるようになっている．

本システムは，大きく分けて 2 つのプログラムから構成される（図 2）．ひとつは実際にエンドユーザ（学生，教職員）が利用する単位チェックプログラムで，Web ブラウザで利用される．もうひとつは，個々のカリキュラムに合わせた卒業単位チェックシステムを構築するためのプログラム（科目表作成プログラム）であり，教務担当者がシステム構築のために利用するものである．

2.1 単位チェックプログラム

図 2 の実践の枠全体が単位チェックプログラムであり，以下のプログラムから構成されている．

メインプログラムファイル (kamoku.htm)

エンドユーザが実際に呼び出す HTML ファイルであり，後述の科目表作成プログラムによって自動生成される．内容は，画面上に表示される科目表の Table の定義がほとんどで，本プログラムの主要部となるプログラムは，以下の 2 つの JavaScript ファイルに収められており，それらのファイルはこのファイルにインクルードされる．

科目情報ファイル (kamoku.js)

科目に関する情報が収められているファイルで，後述の科目表作成プログラムによって自動生成される JavaScript ファイルである．実際の中身は，配列の宣言と変数の値の代入だけのプログラムである．これらの変数は，卒業判定プログラムで参照される．

卒業判定プログラムファイル (main.js)

卒業判定のプログラムを記述したファイルである．このプログラムでは，ユーザが科目区分ごとに入力した取得単位数に基づいて，科目区分単位でチェックを行う．チェックの結果は，次の 3 つの段階で表示を行う．

- 当該の科目区分において単位充足し，かつ，合計単位数を満たすために取得する必要がない．（○；緑表示）
- 当該の科目区分において単位が充足していないか，全体の合計単位数を満たすた

	A	B	C	D	E	F	G	H	I	J	K	L
1	科目番号	科目区分	科目コメント	様式	最低単位数	最大取得可能単位数	入力か合計か	合計項目番号1	算入最低単位	合計項目番号2	算入最低単位	合計項目番号3
2	表題											
3	コメント											
4	1	合計		1	80	0	sum	2		6	12	10
5	2	教養		1	20	0	sum	3		4		5
6	60			1			space					
7	3	人文		2	4	40	input					
8	4	社会		2	4	40	input					
9	5	自然		2	4	40	input					
10	6	語学		1	12	12	sum	7		8		9
11	90			1			space					
12	7	英語		2	8	8	input					
13	8	独語		2	0	6	input					
14	9	仏語		2	0	6	input					
15	10	専門		1	30	0	sum	11		12		
16	80			1			space					
17	11	基礎		2	10	16	input					
18	12	実践		2	0	0	input					

図 3. CSV ファイルの例

科目区分		算入科目	必要単位	認定単位	修得単位
1)合計		2+6(12)+10	80		
2)教養		3~5	20		
	3)人文		4		
	4)社会		4		
	5)自然		4		
6)語学		7~9	12		
	7)英語		8		
	8)独語		0		
	9)仏語		0		
10)専門		11+12	30		
	11)基礎		10		
	12)実践		0		

図 4. 卒業単位チェックシステムで構築された科目表の例

めに必ず取得の必要がある.（×；赤表示）

- 当該の科目区分においては単位が充足しているが，全体の合計単位数を満たしてない．ただし，他の科目区分によっても合計単位数を満たすことが可能であるため，当該の科目区分で必ずしも取得する必要はない.（△；黄表示）

卒業要件を満たすためには，すべての科目区分において，緑表示（○）にならなければならない.

バージョン情報ファイル (information.js)

プログラムの改定情報を記述したファイルでバージョン表示を行う．kamoku.htm と help.htm から呼び出して使われる.

ヘルププログラムファイル (help.htm)

卒業単位チェックプログラムの使い方を説明するプログラムで，メインプログラムからリンクされている.

ヘルプ用科目情報ファイル (help.js)

卒業単位チェックプログラムの使い方を説明するために例示する科目表に関する情報が収められているファイルで，実際の中身は，配列の宣言と変数の値の代入だけのプログラムである．このプログラムも後述の科目表作成プログラムによって自動生成された JavaScript ファイルである．hepl.htm にインクルードされて使われる.

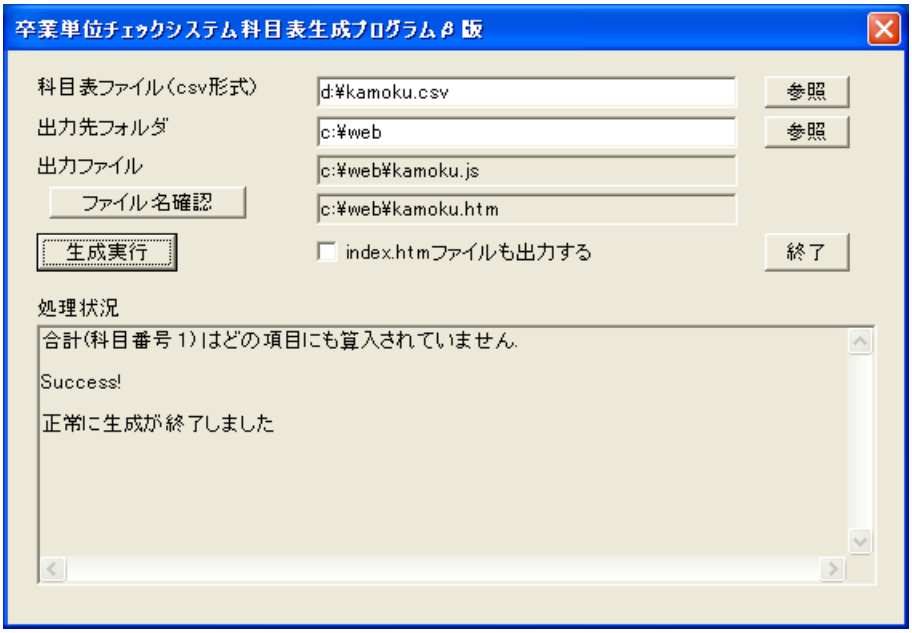


図 5. 科目表作成プログラムの実行例

2.2 科目表作成プログラム

CSV ファイルで記述された科目表ファイルを読み込み，単位チェックプログラムのメインプログラムファイル (kamoku.htm) と科目情報ファイル (kamoku.js) を自動生成するプログラムである．C 言語 (コンパイラ Borland C++ Ver.5.5.1 for Win32) で記述されている Windows プログラムである．CSV ファイルで入力する内容は以下の通りである．

・科目区分名および科目区分コメント
・階層の様式 (表形式にする場合の階層の指定)
・最低単位数および最大取得許容単位
・入力項目であるか合計項目であるか（ユーザが単位数を入力するかどうかの指定）
・合計項目番号および参入最低単位（合計項目である場合．複数設定）

図 3 に示した CSV ファイルのように，Excel で科目表を作成していく．まず，1～99 の科目番号を任意に振る．そして，科目区分の名称を入力し，必要に応じてコメントを入力する．このコメントは科目区分の上にマウスを移動するとポップアップされる．様式は階層の深さを指定する．できあがりの科目表（図 4）の階層は右に広がっていくため，列番号で指定することになる．次に取得すべき最低単位数と最大取得単位数（上限がある場合）を入力する．次の欄は入力項目か合計項目かを示す．利用者が直接取得単位数を入力する科目区分の場合「input」，合計としての科目区分の場合「sum」と入力する．合計項目の場合，その後にどの科目区分を合計するのかを科目番号で示す．たとえば，科目区分「1．合計」の科目区分では「2．教養」，「6．語学」,「10．専門」を合計することを示している．さらに，算入する上限を指定する場合，その最低単位を指定する．図 3 の例では，語学の合計は 12 単位までしか算入させないことを示している．

このようにして，教務担当者が上記の内容を Excel で入力後，CSV 形式に保存し，そのファイルを科目表作成プログラムで実行すると，卒業単位チェックシステムで必要なファイル（kamoku.htm, kamoku.js）が自動生成される．図 5 には，科目表作成プログラムのダイアログを示している．科目表ファイルと出力フォルダを指定し，ファイル名を確認し生成実行を行えばよい．科目表ファイルの書き方に不具合があるとエラーが画面上に作成される．ただしエラーも警告レベルと致命的エラーがあり，致

命的エラーの場合，生成は実行されないが，警告レベルの場合生成実行はなされる．図5では，「合計（科目番号１）はどの項目にも算入されていません」というメッセージが出ているが，これは警告レベルのエラーである．合計は全体の合計であるため，どの項目にも合計算入されなくても問題はない．科目表作成プログラムでチェックされる主要なエラーは以下通りである．

・科目番号が１～99 の範囲でない．
・科目番号が重複している．
・どの項目にも算入されていない．
・合計算入科目として参照している科目番号が存在しない．
・合計がループになっている．
・合計算入科目として参照している科目番号が合計値でも入力値でもない．
・ある科目が複数の合計項目で割り当ての設定になっている．
・合計項目として指定してあるのに，算入科目の指定がない．
・必要最低単位数が合計項目の算入上限単位の合計よりも多くなっている
・階層列番号が１～49 の範囲でない．
・階層列番号が論理的に間違い
・入力・合計の指定が適切でない．

科目表作成プログラムの実行によって自動的に単位チェックプログラムで使われる２つのファイルが生成された後は，単位チェックプログラムに必要なファイル（図 2）を Web にアップロードすれば，卒業単位チェックプログラムが利用できる．

3 卒業単位チェックの判定アルゴリズム

卒業単位チェックプログラムに汎用性をもたせるために，データ構造が木構造となるため，単位計算などの探索において，再帰的なアルゴリズムの導入が必要であった．さらに，科目区分で定められた最低単位数を取得しただけでは上位の科目区分の合計を満たさないことがあり，このような場合の判定に動的パラメータの導入が必要であった．ここでは，再帰的アルゴリズムと動的パラメータの導入によって実現された卒業単位チェックの判定アルゴリズムについて説明する．

3.1 問題

3.1.1 階層的な科目体系の探索

卒業単位チェックシステムでは，各大学のカリキュラムに合わせて，どの科目区分の単位がどの科目区分の合計に算入されるのかを表した科目表を作成するだけで，自動的にその大学に合った単位チェックプログラムが作成される．そのデータ構造は，図 6 に示したようなポインタによる木構造となる（ヴィルト，1990）．各ノードには科目区分名や当該の科目区分における最低取得単位数がデータとして示される．卒業単位の全合計は根ノードとなり，最下位の科目区分は葉ノードとなり，ポインタは null で表される．

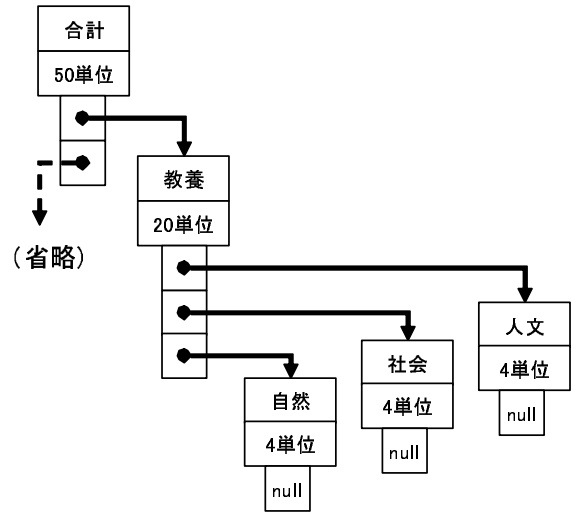


図 6. 階層的科目体系のポインタによる木構造

合計単位を計算するためには，この木構造を下に探索しながら単位数を合計していく必要がある．その際，合計単位は下位ノードの単位を合計するというアルゴリズムによって実現でき

る．下位の科目区分は部分木となるため，再帰的アルゴリズムによって実装することができる．

3.1.2 単位の合計数が一致しないケースの判定

一般に，ある科目区分の必要単位数は，その下位にある複数の科目区分の必要単位数の合計となる．ただし，場合によっては，科目区分ごとに必要な単位の最低数が定められているだけで，それらの必要最低数を合計しても，上位で必要な単位数に満たないことが多い．上位の合計数を満たすには，各科目区分での必要単位数を満たした上に，さらに合計数を満たすように，どこかの科目区分から単位を取得しなければならない．

各科目区分での必要単位数の制約を低くすることによって，学生は自分が取りたい科目区分での科目を多く履修し，それらを卒業単位に反映できるため，学生の科目履修の自由度を高めたカリキュラム体系になっている．このようなカリキュラムの場合，学生によって，科目区分の取得単位数が異なるため，卒業に必要な単位の計算誤りが生じやすくなる．そのため, このようなカリキュラム体系の場合，卒業単位チェックシステム導入の有用性が高い．

ここでは，以下のような用語の使い方をする．必要最低単位数は各科目区分で最低取得しなければならない単位数とする．また，開講科目数などによって認定される単位数の上限が決まっている場合，それを最大認定単位とよぶ．そして，ある科目区分の必要最低単位数がそれよりも下位の科目区分の必要最低単位数の合計と一致する場合，合計一致型とよび，そうでない場合を合計不一致型とよぶ．表１に合計一致型と合計不一致型の例を示した．

合計一致型の場合，当該の科目区分で定められた単位を超えて取得しても，超過分は合計に算入されない．合計一致型でなくても，合計に算入される単位数を限定している場合も，超過分が合計に算入されないため，合計一致型と等価となる．このような場合を擬似合計一致型とよぶこととし，合計の算入の上限を算入上限とよぶ．

表 1．合計一致型と合計不一致型の不足単位の計算例．上は一致型で下が不一致型．

科目区分	必要最低単位	取得単位	不足単位
教養（合計）	20	14	6
人文	8	6	2
社会	6	6	0
自然	6	2	2

科目区分	必要最低単位	取得単位	不足単位
教養（合計）	20	14	6
人文	4	6	0?
社会	4	6	0?
自然	4	2	2?

3.2 再帰的アルゴリズムの必要性

卒業単位チェックシステムとして有用なのは，最終的な卒業判定よりも，どの科目区分でさらに単位を取得する必要があるかを示すことである．合計一致型の場合，各科目区分での不足単位数がそのまま取らなければならない単位数となるため，単純に不足単位数を算出すればよい（表１上，図 7a）．

ところが，合計不一致型の場合，必要最低単位数を満たしていても，上位の合計を満たすために，さらに単位をとらなければならないことがあり，明確に何単位あと取得しなければならないか判定できないため，単純なアルゴリズムでは実現できない．表１下および図 7b に示した例では,「人文」や「社会」では単位を充足しているが,「教養（合計）」を満たすためには，まだ単位を取る必要が残っている．ただし,「自然」で２単位不足しているため，２単位分は確実に補うことができ,「教養（合計）」の不足分の６単位すべてを「人文」や「社会」で補う必要はなく，最大で４単位までとなる．しかも，そ

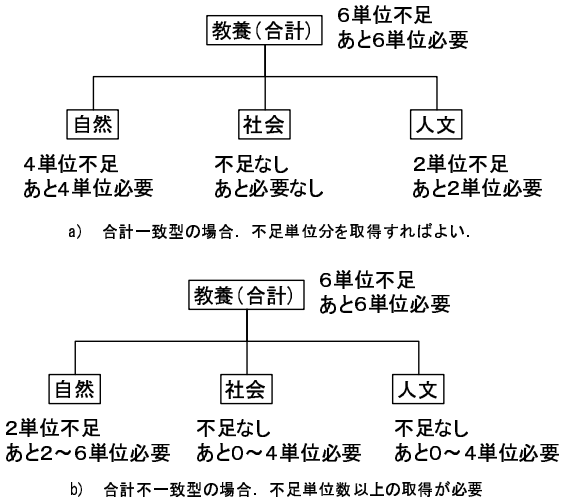


図 7. 各科目区分ごとの判定例

の４単位までというのも,「人文」,「社会」,「自然」のいずれかの科目区分で補ってしまえれば,新たに取る必要がないため,「人文」や「社会」で取るべき単位は，４単位が必須ではなく，最大で４単位にすぎない．そのため,「あと０～４単位必要」という幅をもった判定となってしまう.「自然」についても同様で，最低不足分の２単位は必ず必要であるが，追加４単位分は,「社会」や「人文」で取れれば必要がなくなるため,「あと２～６単位必要」ということになる．

図 7b で示した例では階層が２つで，同じ水準のノードの他の科目区分での必要単位数を計算すればよいが，階層が深くなると，その下位の科目区分に対しても探索が必要となる．そのような例を表 2 に示した．この例では「教養」から見た場合,「合計」の不足 14 単位を補うには,「専門」での２単位不足により２単位は確保できるため，同一水準のノードだけで見れば最大でも 12 単位でよい（図 8a）．しかし「専門」の下位科目区分をみると,「基礎」で８単位不足しているため，その上位の「専門」でも８単位は確保できる．したがって「教養」で補うべき単位の最大は 14 から８を減じた６単位となる（図 8b）.

このように，他の科目区分での不足単位数を計算しながら，当該の科目区分であと何単位必要なのかの計算を行う必要がある．木構造で考

表 2. 下位科目区分まで探索しなければならない例

科目区分	必要最低単位	取得単位数
教養	10	8
専門	20	18
基礎	10	2
実践	0	16
合計	40	26

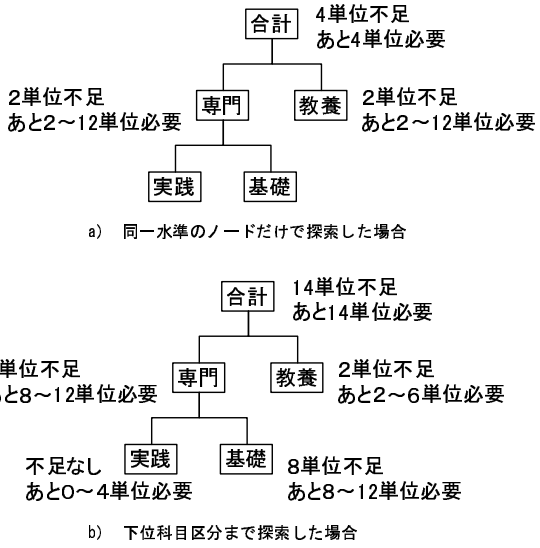


図 8. 下位科目区分まで探索した場合との比較

えると，下位の部分木の探索が必要となり，一種のバックトラックによる再帰的アルゴリズムが必要となる．

3.3 単位充足判定の方法

3.3.1 単位チェックの３段階判定

単位チェックの判定において，あと何単位とる必要があるかを範囲で示すと誤解を招いてしまう恐れがある．範囲の最大値まで取得しなければいけないと思ってしまったり，最小値でよいと思ってしまったりする可能性がある．そのため，単位数は明示せず先に示した３段階での判定を行うこととした．

合計一致型の場合，他の科目区分での単位取得状況とは関係なく単位の充足判定ができるため，当該の科目区分で単位が充足しているかどうかの判定になる．そのため黄判定が生じることなく，あらかじめ定められた最低必要単位数との比較で赤判定か緑判定かが決まってしまう．

主として問題になるのは，合計不一致型の場合，緑判定か黄判定のいずれになるかの判定である．判定には，他の科目区分の単位取得状況によって，どこまで単位を取る必要があるかの情報が必要となる．

3.3.2 動的パラメータの導入

そこで，各科目区分の単位の取得状況によって動的に変化する見込み単位数の最大と最小の値を算出することによって判定ができるようにした．前者を最大見込み単位数，後者を最小見込み単位数とよぶ．

・最大見込み単位数 p_max

合計不一致型の場合，合計単位数を満たすためには，最低必要単位数以上の単位を取得しなければならない．それは，合計に算入される他の科目区分の単位取得状況によって変化する．他の科目区分で必要単位が多く残っていれば，そこで確実に取得単位が増えるため，当該科目区分で新たに取得する単位数は少なくてよいが，そうでない場合は多く取得しなければならない．そのため，各科目区分において最大で何単位まで取得が見込まれるのかを算出する．たとえば，図 7b で「社会」や「人文」から見ると，「自然」はまだ不足単位を２単位残しているため，２単位は確実に増える．そのため，合計６単位不足を満たすのに，「社会」や「人文」では，あと最大４単位とればよく，最大見込み単位数はすでにそれぞれで取得している６単位を加え 10 単位となる（表 3）．一方，「自然」から見ると，「社会」や「人文」では不足単位がないため，「自然」では合計を満たすには最大であと６単位必要で，最大見込みは取得している２単位を加え８単位となる（表 3）．

・最小見込み単位数 p_min

各科目区分においての最低条件は，当該の科目区分で定められている最低必要単位数をまず満たすことである．したがって，最小見込み単位数は最低必要単位数を下回らない．図 7b の「自然」では，最小見込み単位数は４単位となる（表 3）．ただし，すでに最低必要単位数を超えている場合，見込みとしてはその取得単位を下回ることはあり得ないため，取得単位数が最小見込み単位数となる．「社会」や「人文」では必要最低単位はいずれも４単位であるがすでに６単位取得しているため，最小見込み単位数は６単位となる（表 3）．

ここで，最大認定単位 (max)，最低必要単位数（min），最大見込み単位数（p_max），最小見込み単位数（p_min）の関係を図 9 に表した．合計一致型の場合，これら４つのパラメータは同じになるが，合計不一致型の場合，２つの見込み数は max と min の間で動的に変化することになる．

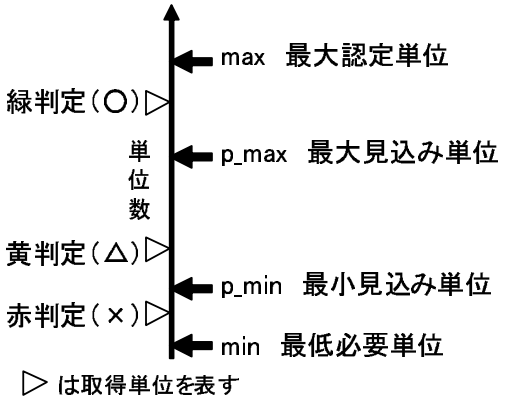


図 9. ２つの動的パラメータと単位充足判定ルールの関係．取得単位の数によって判定が異なる．

3.3.3 単位の充足判定のルール

２つの動的パラメータの導入によって，単位判定は以下のルールに集約される（図 9）．

- 緑判定　取得単位が最大見込み単位数以上の場合

表 3. p_max, p_min の計算と判定の例

科目区分	必要最低単位	取得単位	p_max	p_min	判定
教養（合計）	20	14	20	20	×赤判定
人文	4	6	10	6	△黄判定
社会	4	6	10	6	△黄判定
自然	4	2	8	4	×赤判定

- 赤判定　取得単位が最小見込み単位数に満たない場合
- 黄判定　取得単位が最大見込み単位数未満で最小見込み単位数以上の場合

取得単位数が最大見込み単位数以上であるということは新たに単位を取得する必要がないということである．また，最小見込み単位数に満たないということは，当該科目区分から必ず追加取得が必要であるため，赤判定になる．最大見込み単位数と最小見込み単位数の間に取得単位数がある場合は，上位の科目区分の合計を満たす必要があることを示しているが，それが当該区分でなくてもよく，他の科目区分で追加取得がなされれば，新たに取得の必要はなくなる．このような場合黄判定となる．黄判定は合計一致型の場合生じない．

表 3 に判定例を示した.「人文」と「社会」は，取得単位 6 単位が p_min と p_max の間にあるため黄判定となるが,「自然」と「教養（合計）」はいずれも p_min を取得単位が下回っているため，赤判定となる．このように 2 つの動的パラメータと実際に取得した取得単位を比較することによって，判定が可能となる．

3.4 動的パラメータの算出アルゴリズム

これまで述べてきたように，単位充足判定においては，動的パラメータの 2 つの値を算出することによって判定が可能となる．問題は動的パラメータをどのように算出するかである．それは以下の 3 つのフェーズで行われる．ここでは，表 2 に示した科目体系を用いて説明する．p_min や p_max の算出例を含めて，改めて表 4 に示した．なお，ここでは，取得単位という表現を認定単位と表している．科目体系によっては，取得単位すべてが卒業に必要な単位にならない場合があり，卒業に必要な単位として認定されるものだけを使って判定を行うため，認定単位としている．

表 4. 3 つのフェーズにおける p_max, p_min の更新例

科目区分	必要最低単位数	認定単位数	第 1 フェーズ p_max, p_min の仮設定	第 2 フェーズ p_max の更新	第 3 フェーズ p_min の更新
教養	10	8	10	14	10
専門	20	18	20	30	26
基礎	10	2	10	14	10
実践	0	16	16	20	16
合計	40	26	40	40	40

3.4.1 第１フェーズ　p_min と p_max の仮設定

まず，p_min と p_max の値を，他の科目区分を考慮せずに仮設定を行う．設定はいずれの値も，認定された単位と最低必要単位の大きいほうとする．p_min は，最低でも最低必要単位をとることになるので，最低必要単位以上になる．ただし，先に述べたように，すでに最低必要単位以上の単位が認定されていれば，見込みとしては，それを下回ることはなく，すでに認定された単位を p_min の値として設定する．

一方，p_max は見込みとして最低でも最低必要単位を取るが，すでにそれを越えた単位を取っている場合，見込みとしてはその値を下回ることがないため，p_min と同様すでに認定された値を設定する．

3.4.2 第２フェーズ　他の科目区分での最小取得見込みを考慮した p_max の更新

p_max 更新を再帰的なアルゴリズムとして実装したのが，図 10 である．この計算処理では，まず，他の科目区分で少なくとも何単位増えるのかを他の科目区分ごとに算出する必要がある．それが図 10a のアルゴリズムである．最小見込み単位数から認定単位数を減じれば何単位増えるのかがわかる（図 10a　）．ここで増える単位は，算入上限があれば，それを超えて上位の科目区分に算入されないため，それを考慮している．表 4 の例の「専門」で何単位増えるかは，第１フェーズでの p_min の値 20 から認定単位 18 を減じた値２単位となる．これが当該科目の増となる．

しかし，先に述べたように下位科目区分も探索する必要があり，下位科目区分での不足単位があれば，その分増えることになるため，下位科目区分での増の合計を再帰的に最下層までた

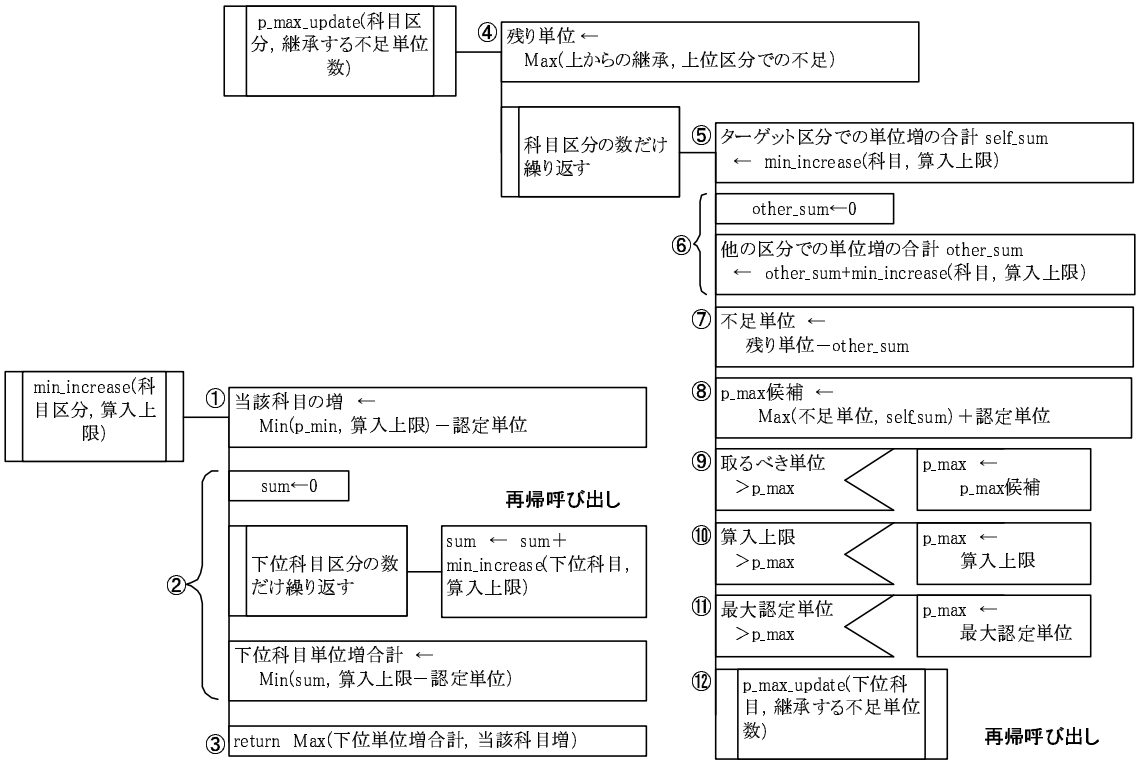


図 10. 最大見込み単位数 (p_max) 更新のアルゴリズム

どって合計を行う（図 10a ）．表４の例では，「基礎」と「実践」の科目の２つの下位科目区分の分だけ合計を算出する．「基礎」は８単位増が見込まれ，「実践」では単位増無しで，ここの合計 (sum) は８単位となる．ここでも算入上限を考慮して，下位科目単位増合計が算出される．この値と当該科目増の大きいほうの値がその当該科目区分での増分になる（図 10a ）．下位科目区分（「基礎」と「実践」）の合計８単位増と「専門」の２単位増を比較して，大きいほうの８単位が上位科目区分に増える単位数となる．

ここで再帰的に呼び出されている min_increase の関数は２つの引数を持っており，科目区分を指定する引数と算入上限の制約を指定することになっている．このアルゴリズムは実際には次に説明する p_max の更新のアルゴリズムで利用される．

p_max の更新では，上記で述べた他の科目区分で算出される単位増の値を見込んで，最大何単位取得を見込めばよいかを算出する．そのアルゴリズムが図 10b である．まず，上位から残単位を継承し，同時に上の科目区分での不足分と比較し，大きいほうの値を残り単位としてまかなう必要がある（図 10b ）．ここでは「教養」と「専門」に関して注目すると，その上位区分は「合計」になる．「合計」では，14 単位不足している．さらに「合計」の上位でもっと多く単位不足が生じている可能性があることも考慮して，その不足単位（上からの継承）も継承しなければならない．その単位数が多ければ，その単位数を下位でまかなうこととなる．ただし，表 4 の例の場合，「合計」の上は存在しないため，「合計」で不足している 14 単位を下位の「教養」と「専門」でまかなうことになる．

次に，ここでの科目区分の「教養」と「専門」の２つについて，自分のところの下位科目区分での単位増と他の科目区分の単位増を計算して，それらで不足分をどこまでまかなえるかどうかを算出する．そのために，以下の処理を繰り返すことになる．まず，ひとつの科目区分をターゲット（ターゲット区分）にして，その科目区分での不足単位（あと何単位増えるか）を計算する（self_sum）．たとえば，教養について計算する．ここでは図 10a に示したアルゴリズムを使い，それよりも下位の科目区分も探索して何単位増えるかを算出する（図 10b ）．「教養」には下位科目は存在しないため，この場合下位探索は行わなくてよく，「教養」では２単位増だけとなる．

次に他の科目区分でどれだけ単位が補えるかを合計する．ここでは他の科目区分は「専門」だけであるが，複数あれば複数の科目区分の合計（other_sum）を算出する（図 10b ）．ここでも，図 10a に示したアルゴリズムを使って再帰的に下位科目区分まで探索をしていく．ここでの科目区分「専門」の計算過程は，図 10a のアルゴリズムを説明したときにすでに述べた通りである．Other_sum は８単位となる．

この値は「合計」の科目区分の残り単位を減じるのに使え，その残り分が不足単位となる（図 10b ）．この場合，14 単位から８単位を減じた６単位となる．この値がターゲット科目区分「教養」で取るべき最大の単位数となる．ただし，先ほど算出した「教養」自身で増える２単位（self_sum）がそれを上回っていれば，そちらのほうが取るべき最大単位数となる．そして，その値に認定された単位を加えた単位が p_max の候補値となる（図 10b ）．つまり，６単位と２単位の大きい方の６単位にすでに認定されている８単位を加えて，14 単位が p_max の候補となる．そして，それがすでに設定されている p_max よりも大きい場合，この候補値で更新することとなる (図 10b)．後は，算入上限 (図 10b) や最大認定単位（図 10b ）を超えないようにする必要がある．そして，さらに，下位の科目区分でも同様の処理を行っていくために再帰的に行う（図 10b ）．

この一連の処理をターゲット科目区分を順々に変えて行う．ここでは，あとは「専門」をターゲット科目区分として処理するだけである．「専門」がターゲットとなると，さらにその下位科目区分「基礎」や「実践」でも同様の処理を行う．その結果，表４に示した値で p_max の更新

がなされる．

ここでの再帰的関数 p_max_update は科目区分の指定の引数と下に継承していく不足単位数を引数として指定することとなっている．

3.4.3 第３フェーズ　他の科目区分での最大取得見込みを考慮した p_min の更新

第２フェーズでは，他の科目区分での単位の取得可能性の最低を考慮したが，第３フェーズでは，他の科目区分での単位の取得が最大の可能性を考慮することになる．他の科目区分で単位を最大限取得しても，まだ足りない可能性があった場合，足りない単位分は当該の科目区分で取得しなければならない．その単位数がどの程度であるのかを計算して，当該科目区分での最小見込み単位数 p_min を更新する．

第３フェーズのアルゴリズムを図 11 に示したが，第２フェーズのアルゴリズムとほとんど類似しており．以下の２つの点のみが異なる．

まず，第２フェーズが他の科目区分での最小取得見込みを考慮するのに対して，第３フェーズでは，他の科目区分での最大取得見込みを考慮するため，何単位増えるかの算出において，第２フェーズで p_min と認定単位の差で計算していたのを p_max との差で計算するところが異なるだけである（図 11a　）．もうひとつは，ターゲット科目区分の合計を０とすることである（図 11b　）．これは，不足単位との比較において（図 11b　），不足単位がマイナスを示すこともあり，０を超えないようにするためにすぎず，ターゲット科目区分の合計を０とみなすこと自体には特別の意味はない．プログラムのメンテナンス上，第２フェーズと第３フェーズのプログラムを同じ形式にするメリットがあるためである．第２フェーズにおいて，すでに他の科目区分の p_max の値にターゲット科目区分での最小見込み単位増が織り込まれているため，ターゲット科目区分の単位増は考慮する

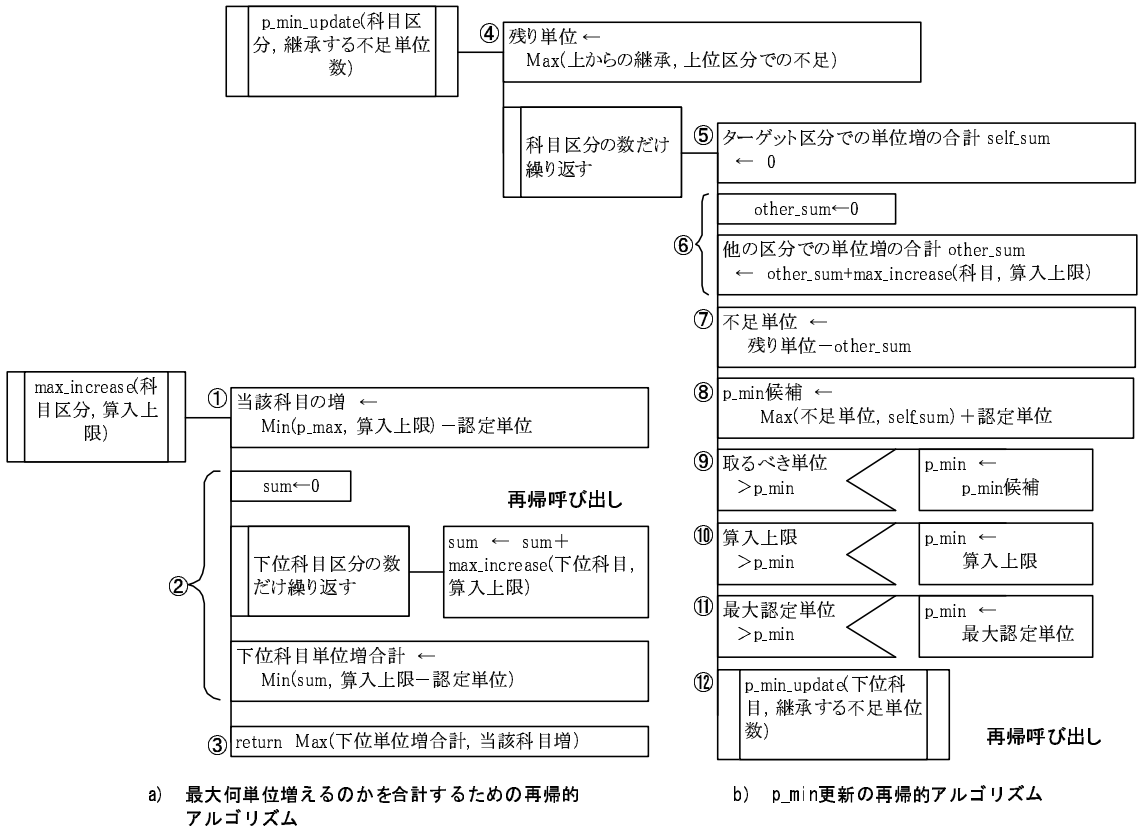


図 11. 最小見込み単位数 (p_min) 更新のアルゴリズム

必要がない．

表 4 における「専門」の p_min の更新のプロセスを追うと以下のようになる．残り単位は「合計」14 単位で第 2 フェーズの場合と変わらない．他の区分での単位増の合計（図 11b ）は，対象となるのが「教養」だけであり，今回は，p_max と認定単位との差（図 11a ）で算出され 6 単位となる．そして，残り単位 14 との差の不足単位が 8 となる．この値に「専門」の認定単位 18 を加えたのが，p_min の候補値で 26 単位となる．第 1 フェーズで設定された p_min20 単位よりも大きいため，26 単位で更新されることになる．後の科目区分も同様の計算を行うことになるが，いずれも他の科目区分での単位増が残り単位を上回り，不足単位が負の値を示す．そのため，p_min の候補は，現在値を超えることはなく，更新されないこととなる（表 4）．

4 考察

4.1 アルゴリズムの有効性

本システムの開発にあたって，2 つの動的パラメータを導入し，その算出を再帰的アルゴリズムによって実装できた．図 9 に示したように，2 つの動的パラメータによって 3 段階の判定基準が明確で単純になった．その動的パラメータを更新するために再帰的なアルゴリズムが有効であった．この結果，汎用性をもたせ，どのような科目体系でも構築できることになった．

図 1 に示した例は，表 4 に示したカリキュラムに語学の科目を加えたもので，語学は最大認定単位 10 単位となっており，これ以上単位を取得しても，上位の合計には貢献できず，すでに 10 単位取得しているため，緑判定（○）となっている．このように合計不一致型の体系であっても，どの科目区分でさらに単位取得が必要であるのかが簡単にわかるようになった．

ただし，他の科目区分に単位を流用するプログラムが十分に完成しておらず，今後さらに検討する必要がある．また，現在のプログラムでは，科目区分ごとに，他の科目区分で何単位取得する可能性があるのかを計算しており，一度算出した値を再度計算し直す形になっており，無駄な計算を無くすような工夫も必要である．

4.2 卒業単位チェックシステムの有用性

本システムの実現によって，学生は科目区分ごとに取得単位数を入力するだけで，3 つのレベルで単位の取得状況を確認できるようになった．学生はどの科目区分が不足しているのかがわかり，単位計算を誤って卒業できなくなったりしてしまうことがほとんど無くなる．さらにこのシステムの利点は，表示結果をもとに履修プランを立て，取得予定単位数を入力すれば，計画通りの単位取得で卒業要件を実際に満たすのかどうか確認できることである．卒業できるのかどうか不安を感じる学生の不安を解消することができる．

また，このシステムは汎用性をもたせたシステムであるため，他大学での利用も可能であり，教務担当者が科目表を作成すればどの大学のカリキュラムにも対応することができる．卒業要件だけではなく，卒業論文（卒業研究）着手要件，進級要件などの設定も簡単に行うことができる．

履修プランや卒業要件のチェックは，学生自身が履修便覧等を精読して，履修要件を十分に理解すれば生じることではないことであるが，人間である以上，間違いが生じるのはやむを得ない．人間が起こす間違い（ヒューマンエラー）に対して，松尾（2003）は，間違いであることに気づかせるための外的手がかりのしくみを設けることが必要だと主張している．自分だけで考えていると大丈夫だと思い込んでしまい，間違いに気づかないことが多い．松尾は外的手がかりとして，対象，表示，ドキュメント，人を提案している．卒業単位のような情報に関するエラーの場合において，松尾の外的手がかりの提案に適用すると，人，ドキュメントの外的手がかりのしくみを設けることが考えられる．

人は，友人，教員，事務職員に尋ねるなど，自分以外の他者に確認をしてもらうことを指す．しかし，現実は人という外的手がかりに頼ることは難しい．友人と確認しあったとしても，その確認で正しいのかどうか学生の不安がなくなるわけではない．教員に尋ねることは敷居が高いし，教員の中には卒業要件を十分に把握していない教員もいる．事務職員の場合，一人ひとりの学生に対応するのは大変であるし，事務職員も人間である以上，間違うこともあり，間違ったことを教えてしまった場合，責任問題となってしまう．

ドキュメントは，履修便覧や確認表などで確認することであるが，それを読むコストがかかったりわかりにくかったりすることがある．誰もが履修便覧で確認できれば，卒業単位の計算ミスは起きないわけであり，このようなドキュメントを使っても，単位計算を誤って卒業できなくなるケースが出てくるのである．したがって，このようなドキュメントは有効ではない．．松尾はドキュメントの中には，紙だけではなく電子的なものも含めて考えている．本研究で開発したチェックシステムは電子的なドキュメントに相当する．しかも，単なるドキュメントというよりも電子アシスタントというべきものである．このような電子アシスタントであれば間違いはなくなる．

ただし，松尾 (2003) が指摘しているように，外的手がかりは主観的確信が高い場合，たとえしくみとして設けてあってもそれを利用されないことが考えられる．まり，学生自身の主観的確信が高いとこのようなシステムは利用されにくい．学生が大丈夫だと思い込んでこのシステムを使わなければ意味がない．また，単位数の入力ミスに対しても脆弱である．汎用性を持たせたため，学生に配布される成績一覧表と本システムの表示画面の対応がわかりにくくなり，単位数の入力ミスが発生する可能性もある．学務情報システムが完備して，自分の取得単位を確認する際に同時にこのようなチェックシステムが使えることが理想的であるかもしれない．しかし，導入コスト等の問題を考えれば，本システムの有用性はかなり高いものであると考えられる．

5 謝辞

東亜大学の具志堅伸隆先生には，本システムを使っていただき，バグ等の指摘をいただきました．記して感謝いたします．

参考文献

[1] 岩下亜希子・片桐理紗,・天田貴大・南弘樹・中居祐一 (2006). インターネットを利用した教員免許取得単位のチェックシステム 朝日大学学長研究奨励費研究結果論文集, 2, 26-31.

[2] 梶田将司･角所考･中澤馬志･竹村治雄･美濃導彦・間瀬健二 (2007). 高等教育機関における次世代コース管理システムの構築に向けて 日本教育工学会論文誌, 31, 297-305.

[3] 栗原和夫 (2007). Web アプリによる卒業要件単位計算 情報学研究（朝日大学経営学部電子計算機室年報），16, 21-28.

[4] 松尾太加志 (2003). 外的手掛かりによるヒューマンエラー防止のための動機づけモデル，ヒューマンインタフェース学会誌, 5, 75-84.

[5] 吉牟田裕 (2004). Web による履修登録システムの構築と運用－ウェブデザインからDBMS、運用体制まで－，富山国際大学地域学部紀要，4, 111-116.

[6] ヴィルト，N.(浦昭二・國府方久史　訳) (1990). データ構造とアルゴリズム 近代科学社

A　卒業単位チェックシステムの使い方

help.htm 実行時の画面

① 修得単位の入力

自分の成績表から各科目群の単位を入力する.

科目群	算入科目	必要単位	認定単位	修得単位	
1) □□科目		20			←修得した単位を入れる
2) ○○科目		10			←修得した単位を入れる

② 単位を計算させる

単位の計算 ←ここをクリック

以下のように単位のチェック結果が色分けして表示される.
必要単位と認定単位を比較すると, 単位の過不足がわかる.

科目群		算入科目	必要単位	認定単位	修得単位	
1) 基礎科目			20	○ 20	20	←単位充足
2) 専門合計	3) 専門基礎科目		8	○ (8)	10	←10単位修得だが, 認定最大は8単位まで. 単位は充足
3~6	4) 専門応用科目		8	△ 12	12	}←必要単位は満たしているが, 専門合計を満たすためには, さらに取得も要検討
必要 40	5) 専門A科目		6	△ 8	8	
認定 × 32	6) 専門B科目		6	× 4	4	}←単位不足
7) 卒業合計		1+2	60	× 52		

↑「2)専門合計」は単位不足.

1)基礎科目は, 必要単位を充足(○)している.
2)専門合計は, 3)~6)の科目の合計であるが(3~6), 必要単位40を満たしていない(×).
4)専門応用科目と5)専門A科目は, 必要単位を満たしているが, 3~6の合計の2)専門合計を満たすには, いずれかの科目を増やす選択もあるので, 完全に充足(○)ではなく, △で表示される. 3)~6)の必要単位を合計(8 + 8 + 6 + 6 = 28)しても, 2)専門合計の必要単位40にはならないため, 3)~6)の科目は必要単位を超えて修得する必要がある(単位の体系によっては, 単位を増やしても意味ない場合であっても, △表示になることがある).
ただし, 3)専門基礎科目は, 10単位修得しているが, 最大認定単位が8単位であるため, 8単位しか認定されず(8), これ以上修得しても2)専門合計には貢献しないため, 充足(○)と表示されている.
6)専門B科目は, 必要単位を満たしていない(×).
7)卒業合計は, 1)基礎科目と2)専門合計の合計(1+2)であるが, 必要単位を満たしていない(×).

③ 単位を増やしてシミュレーションを

修得単位の数を増やして「単位の計算」をやってみる.
そうすると, どの科目群の単位を増やせば, 卒業要件を満たすかがわかる.
たとえば, 専門A科目を8→12とし, 専門B科目を4→8としてみる.

科目群		算入科目	必要単位	認定単位	修得単位
1) 基礎科目			20	○ 20	20
2) 専門合計	3) 専門基礎科目		8	○ (8)	10
3~6	4) 専門応用科目		8	○ 12	12
必要 40	5) 専門A科目		6	○ 12	12
認定 ○ 40	6) 専門B科目		6	○ 8	8
7) 卒業合計		1+2	60	○ 60	

すべてが充足(○)と判定されたため, 要件を満たしたことになる. ひとつでも×があると要件は満たしていない.

● 注意
このシステムは単位数だけのチェックを行うシステムです.
科目の修得の仕方に細かい規定がある場合, **単位数だけでは最終的な卒業判定ができません.**
大学・学部・学科によって異なります. 自分で確認してください.

トップページに戻る　前のページに戻る

卒業単位チェックシステム β版Ver.2.21

B　科目表作成プログラムの使い方

1. CSVファイルの作成

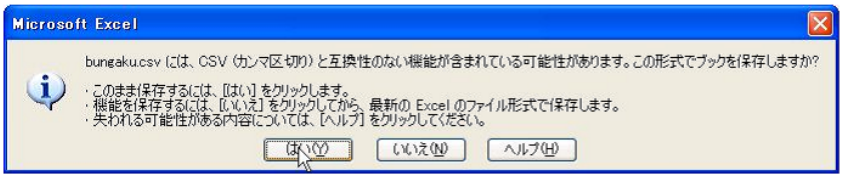
Excelを使った場合で説明します.

1行	入力の必要なし. 各列に入力すべき項目の名称が書いてありますが, 無くてもかまいません.
2行 B列	**表題** 学部学科名等を入力するのがいいでしょう. Webでのタイトルになります.
3行 B列	**コメント** 表題のすぐ下に表示されます.
4行以降	以下のように科目情報を入力します.
A列	**科目番号**(1～99:半角). 順序の制約はありません. 同じ番号は使えません. 表示したときに, 表の上から順番になるようにするとよいでしょう. dummy, spaceの場合, 番号は表示されませんので, 80番代とか90番代を使うとよいでしょう.
B列	**科目区分名** 表示される文字数は全角文字(半角文字). G列で指定するタイプがspaceの場合, 空欄で可. nameの場合, 科目番号が表示されず, ここで指定した文字列だけが表示される.
C列	**コメント** 全角文字以内(半角文字以内). 画面上で科目区分名にマウスポインタをあわせると, ポップアップで表示されます.
D列	**階層番号**(1～49). 科目表の列番号を指定します. 新しく行を変えるときは1を指定します. 直前の行の科目区分の下に並べる場合は, 直前の行で指定した階層番号と同じ番号を指定してください. 同じ行で横に並べる場合は, 直前の行の階層番号よりひとつ大きい値を指定してください.
E列	**最小取得単位**
F列	**最大認定単位** 単位を何単位とっても認定上限がある場合, あるいは, 開講科目数がはっきりわかっていて, 取得上限が明確な場合, 指定してください. できる限りに指定したほうが細かい判定が可能になります. 0は上限無しを意味する. 空欄でも上限無しとみなす.
G列	**タイプ**　以下のように指定します. **input**　取得単位を入力する科目区分であることを示す. **sum**　他の科目区分を合計する区分であることを示す. **space**　科目区分名のところが空白になる. レイアウト上使用するとよい. **name**　科目区分名のところにコメントなどを表示するようにレイアウトしたいときに使う. spaceでは, 空欄になるが, nameでは科目区分名で指定した文字列が表示される.
H列	**合計科目番号** sumの場合, 合計する他の科目区分の番号を指定する. 数字の後にmを付加すると, 合計ではなく, 移動(move)を意味する.
I列	**算入上限** 合計に算入する際, 上限を指定する. 0は上限無しを意味する. 合計する科目区分が複数ある場合, 上記の要領で, 2列ずつに以下の列に入力していく
J,K列:... L,M列:	

1行の入力文字制限が640文字

2. CSVファイルの保存

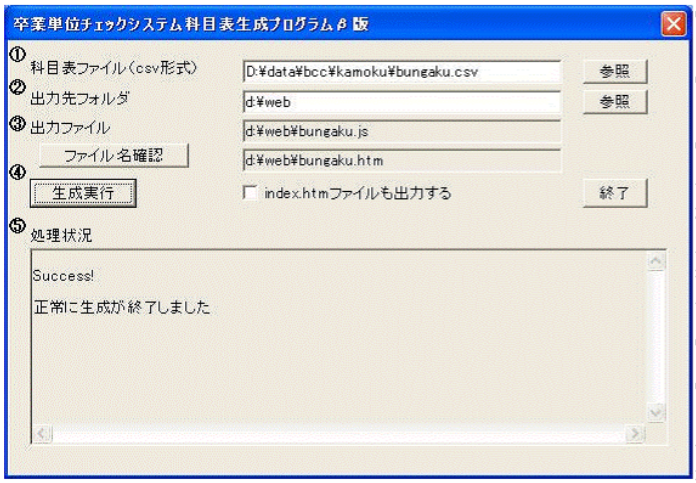
メニュー「ファイル」－「名前をつけて保存」から, ファイルの種類に「CSV(カンマ区切り)」を選択.
　CSVファイルを「bungaku」とした場合(以下, 「bungaku」として例で説明します.
ここで決めたファイル名が, 生成されるファイル(htmlファイルとjsファイル)の名称に使われるため, 他のファイル名と重複しないように設定してください. Webで利用することを考えて, 英数半角文字にしておいてください.



上記のメッセージが表示されますが, そのまま「はい(Y)」をクリックしてください. それで保存されています.

3. 科目表生成プログラムの起動

科目表生成プログラム(kamoku.exe)の起動



①科目表ファイル(csvファイル)の指定
ファイル名を直接入力するか,「参照」をクリックしてファイルを指定
②出力フォルダの指定
フォルダ名を直接入力するか,「参照」をクリックしてフォルダを指定
③出力ファイルの確認
「ファイル名確認」をクリックして,指定したフォルダやファイル名に間違いがないかどうか確認
「index.htmファイルも出力する」トップページとなるファイルを作成したい場合,チェックを入れてください.とりあえず動かせるよう簡単なものを作成します.実際に利用する場合は作成し直すのがよいと思います.
④生成実行
「生成実行」をクリック
⑤処理状況の確認
csv形式の科目表ファイルの内容に論理的におかしなところがあるとエラーが出てきます.エラーがあった場合,「処理状況」内に表示されますので,科目表ファイルを修正して,改めて生成を実行してください.

4. プログラムファイルのコピー

以下のファイルを出力先フォルダにコピーしてください.
maink.js
help.htm
help.js
information.js
kai.css

5. テスト実行

出力先のフォルダのbungaku.htmファイルをクリックして起動してください.いろいろなケースを考えて,卒業要件通りにチェックできているかどうかを確かめてみてください.とくに,特異なケースのチェックが必要となります.

6. Webサーバへのアップロード

①必要に応じて,index.htmファイルを作り直す.
複数の学部学科の単位チェックシステムを構築する場合,プログラムファイルは共通なので,CSVファイルの名称を変え,同じフォルダに出力すれば大丈夫です.
以下の例は,「bungaku」のほか,「keizai」,「hougaku」も作った例です.

後は，index.htmを少し作り変えればいいのです．
以下のような記述に作り変えてください．

```
---------index.htm-----------------------------------------
    <a href="bungaku.htm">文学部</a><br>
    <a href="keizai.htm">経済学部</a><br>
    <a href="hougaku.htm">法学部</a><br>
-------------------------------------------------------------
```

②出力先のフォルダのファイルをすべてWebサーバにアップロードして使ってください．

index.htm
bungaku.htm
bungaku.js
hougaku.htm
hougaku.js
keizai.htm
keizai.js
maink.js
help.htm
help.js
information.js
kai.css

C 研究成果の発表

1. 「汎用的な卒業単位チェックシステム」（教育システム情報学会第 34 回全国大会　2009 年 8 月 20 日）

2. 「階層的科目体系での卒業単位チェックプログラムにおける再帰的アルゴリズム」（日本情報ディレクトリ学会第 13 回全国大会　2009 年 9 月 12 日）

3. 「階層的科目体系における卒業単位チェックの判定アルゴリズム－動的パラメータの導入と再帰的アルゴリズムによる実現－」（日本情報ディレクトリ学会誌第 8 巻　2010 年 3 月）