初心者向けの
プログラミング演習方式に関する研究

平成18年度

三重大学大学院工学研究科
博士前期課程 電気電子工学専攻

川元健司

修士論文

初心者向けの
プログラミング演習方式に関する研究

平成 18 年度修了

三重大学大学院工学研究科

博士前期課程　電気電子工学専攻

川元 健司

目次

第1章　はじめに

近年の情報化社会の進展に伴い，大学におけるプログラミング教育の重要性が高まっている．プログラミングに関する講義では，学生にプログラム作成の能力を身につけさせることが重要である．そのために，プログラム作成の演習が行われている．しかし，現状では，学生は課題で指定されたプログラムを作成できないことがある．プログラミングの初心者では，特に，これが顕著となる．

プログラミング初心者がプログラム作成できるようにするための演習形式にごく短い空欄にプログラムを記述させる穴埋め形式がある．しかし，穴埋め形式で演習を行っても学生はプログラムの処理の手順を考えずにプログラム作成することがあるため，穴埋め形式で演習を行っても自分の力でプログラム作成できるようにはならない．

本研究では，プログラミング初心者，特に，プログラミングの苦手な学生であっても，プログラムの形になったものを作成することのできる演習方式を提案する．ここで，プログラムの形になったものとしたのは，構文エラー，動作エラーなどのエラーを含んでいてもかまわないことを意味する．エラーを含んでいてもプログラムの形となったものが提出されれば，それに対して指導を行うことができるが，何も提出されていない状態で指導を行うことは難しい．

本研究では，次の二つの特徴を持つ新しいプログラム作成の演習方式を提案する．

(1) 簡単な短いプログラムであっても，プログラミングが苦手な学生は，何から書き始めたらよいのかが分からずプログラムの作成をすることができない．そこで，プログラム全体を一から記述させるのではなく，プログラムの雛形は予め与えておいて，学習させたい機能を使う部分を空欄として，その部分のみ記述させることにする．空欄に当たる部分の大きさは，学習させたい機能を使うひとまとまりの処理の数行とする．

(2) プログラムの雛形が予め与えられていても，プログラミングが苦手な学生の場合は，与えられている部分に書かれていることを理解できないことがある．また，空欄部分の処理をどのように考えて記述してよいのか

が分からないことがある．そこで，与えられている部分，空欄部分，それぞれに対して，プログラムを書くときに考えるべきことを，学生の要求に応じて段階的に提供することにする．

本論文の構成を以下に示す．第２章で，プログラミング演習の現状について述べる．第３章で，提案する初心者向けの演習方式について述べる．第４章で，提案する演習形式を用いて行った演習の結果について述べる．最後に，第５章でまとめる．

第2章　プログラミング演習の現状

2.1　プログラミングに関する講義

プログラミング教育では，講師がプログラミング言語の機能の説明を行った後に，学生がその機能を使ったプログラムを書けるようにするためにプログラム作成の演習が行われる．演習で使われるプログラム作成の課題には，次の二種類がある．ひとつは，プログラミング言語の機能を説明したときに用いた例題のプログラムの動作を多少変更したプログラム作成する課題である．この種類の課題の場合，学生は例題のプログラムを参考に，極端に言うと，そのプログラムをそのまま入力し，動作の変わった部分のみを自分で考えて書くことになる．もうひとつは，説明された機能を用いたプログラムを作成する課題であるが，直接参考となるプログラムが例題などで与えられていないものである．この種類の課題の場合，学生は自分の理解を基に新規にプログラム作成することになる．実際のシステム開発においては例題になるようなプログラムは存在せず開発者が一からプログラム作成することになる．そのため，将来，システム開発に携わる可能性のある学生には，新規の課題に対してプログラム作成することのできる能力が求められる．新規の課題に対してプログラム作成することができてはじめて，プログラミング能力を修得したということができる．

本研究ではまず，二種類の課題でプログラミング演習を実施したときのプログラムの提出状況を調査した．調査は2005年度後期プログラミング演習Iの受講者40人を対象に行った．課題の総数は4問である．表2.1に講義中にプログラムを提出できた学生の人数を示す．

表2.1 講義中のプログラム提出者数

課題	提出者数	課題の種類
問1	38人	類題
問2	36人	類題
問3	15人	新規の課題
問4	4人	新規の課題

講義中に演習の状況をその場で観察したところ，問 2 のプログラムを提出した時点で，残りの授業時間が十分あったにもかかわらず，新規の課題である問 3 のプログラムを提出できなかった学生の多くがプログラム作成できていなかった．

実際にプログラミングが苦手な学生がどの程度プログラム作成できないかを調べるために，プログラミングが苦手な学生 3 人に新規のプログラム作成の演習を行ってもらった．課題 7 問を出題した．課題は図 2.1 に示すような簡単な課題である．表 2.2 に学生のプログラム提出数を示す．表 2.2 に示すように，プログラミングが苦手な学生は簡単な課題でも新規の課題となるとプログラムを作成できていないことがわかった．

> 問5
> 読み込んだ数字を10 回表示するプログラムを，for 文を用いて作成してください．

図 2.1 調査のためのプログラム作成課題

表 2.2 プログラム提出者数

課題	問 1	問 2	問 3	問 4	問 5	問 6	問 7
提出者数	0	2	1	2	0	1	1

2.2 初心者のための演習形式

プログラムが作成できない者の負担を軽減し，プログラム作成できるようにするために，図 2.3 に示すようなごく短い空欄にプログラムを記述させる穴埋め形式で演習が行われることがある[1][2]．しかし，図 2.3 に示すような穴埋め形式では，学生はプログラム全体の流れを自ら考えないことがあり，空欄を埋めることができても類題でプログラムを作成したことと大きな違いはない．そのため，学生にプログラムの流れを考えさせて，プログラム作成させる必要がある．

課題プログラム

```
001 /*配列に10個の整数値を読み込んで最大値を表示する*/
002
003 #include <stdio.h>
004
005 #define SIZE 10
006
007 int main(void)
008 {
009         int i;
010         int max;
011         int [          ]
012
013         printf("10個の整数値を入力してください.¥n");
014         for(i=0;i<SIZE;i++){
015             printf("整数%d:",i+1);
016             scanf("%d",&n[i]);
017         }
018
019         max=[          ]
020         for(i=1;i<SIZE;i++){
021             if(max<n[i]) max=[          ]
022         }
023
024         printf("入力された整数値の中の最大値は%dです．¥n,max");
025
026         return(0);
027 }
```

図 2.3 従来の穴埋め形式

第3章　本研究で提案する初心者向けの演習方式

本研究ではプログラミングが苦手な学生が新規の課題でプログラム作成をできるように演習方式を提案する.

3.1 部分記述形式のプログラム作成

本研究では学生がプログラム作成できるように，課題で指定したプログラム全体ではなく，プログラム中で学習させたい機能を用いる部分数行のみを記述させる部分記述形式を提案する.

部分記述形式では，図 3.1 に示すように，プログラムの雛形は予め与えておき，学習させたい機能を使う部分のみを記述させることにする．空欄の大きさは，学習させたい機能を使うひとまとまりの処理に相当する数行とする．これにより，学生にプログラム全体の流れを考えさせることができる.

課題プログラム

```
001   /*
002 配列に10個の整数値を読み込んで最大値を表示する
003  */
004
005  #include <stdio.h>
006
007  #define SIZE 10 /* 配列の大きさ */
008
009  int main(void)
010  {
011       int n[SIZE] ; /* 値を格納するための配列 */
012       int i ; /* 配列の添え字のための変数 */
013       int max;     /* 最大値を格納するための変数 */
014
015       /* 配列に整数値を10個読み込む。 */
016       printf("整数値を10個入力してください。¥n");
017       for ( i=0 ; i<SIZE ; i++ ) {
018            printf("整数%d:", i+1);
019            scanf("%d", &n[i]);
020       }
021
022       /* 最大値を求める。 */
```

ここにプログラムを記述してください.

```
023
024       /* 最大値を表示する。 */
025       printf("最大値は%dです。¥n", max);
026
027       return (0);
```

図 3.1 提案する部分記述形式

3.2 部分記述形式の有効性を確認するための実験

本研究で提案する部分記述形式の有効性を確認するために，実際の講義で実験を行った．

実験では演習を2回行った．1回目の演習では，前半クラスの学生に初めからプログラムを記述させる新規の課題を課し，後半クラスの学生に部分記述形式の課題を課した．2回目の演習では，前半クラスと後半クラスを入れ替えた．なお実験前に行われた中間試験の成績のクラス平均では，クラス間に講義内容の理解に大きな差はなかった．

実験では以下の条件で演習を行い，アンケートを実施した．

科目： 2005年度後期プログラミング演習I
対象： 電気電子工学科 1年生　前半クラス40人
後半クラス48人

以下にアンケート結果を示す．

① 部分記述形式の課題において，プログラムが一部与えられていましたが，初めからプログラムを記述する形式の課題と比較して課題を考え易くなりましたか？
- 考え易くなった　66人
- 考えにくくなった　1人
- 変わらない　12人

② 部分記述形式の課題において，予め与えられていた部分のプログラムをすぐに理解することができましたか？
- すぐに理解できた　47人
- なかなか理解できなかった　33人
- 未回答　8人

アンケートの結果より，学生は部分記述形式で課題に取り組むほうが，課題を考え易いと感じていることがわかる．しかし，講義で部分記述形式でプログラム作成に取り組んだ学生のプログラム作成状況を観察していたところ，プログラム作成できるようになってはいなかった．そのため，プログラミン

グが苦手な学生がプログラムを作成できるようにするためにはどういった指導が良いか調査する必要があると考えた.

また，予め与えられていた部分のプログラムをなかなか理解できなかったという回答が四割近くになったことから，予め与えた部分に対する解説が必要であると考えた.

3.3 プログラミング初心者に対する指導

学生がプログラム作成できない時にどのような指導を行ったら良いかを実際に演習を行い調査した．以下の条件で，学生がプログラムの作成に行き詰ったときに，どこが分からないのかを聞きながら調査を進めた.

- 対象：プログラミングが苦手な学生
- 演習課題数：7 問
- 課題：図 2.1 に示すような簡単な課題

学生のプログラム作成状況を観察していたところ，以下のようにプログラムを作成できていない状況はさまざまであった.

- どういった変数が必要かわかっていない.
- 何から記述したらよいかわからず，止まっている.
- 機能（if 文，for 文など）の使い方を理解していない.
- アルゴリズムを考えられない.

上記の学生の状況に応じて以下の内容の指導を行うことでプログラム作成できるようになった.

- プログラムの全体像を教える.
- プログラムの処理手順を教える.
- プログラム作成に用いるべき機能（if 文，for 文など）を教える.
- 変数の値の変化を図で説明する.

また，大まかな説明をしても理解できなかった学生に対してはより詳細な説明を行うことでプログラム作成をできるようになった．これらの結果より，学生の状況に応じて，段階的に解説を与えることによりプログラム

を作成できるようになるということがわかった.

しかし，実際の講義で講師が，学生の状況に応じて段階的に解説を与えることは負担が大きく難しい．そこで本研究ではシステムを用いて，学生の要求に応じて段階的に解説を与えることでプログラム作成の支援を行う.

3.4 学生の要求に応じた解説の提供

前節で述べたように，予め与えた部分のプログラムを理解できないことや，プログラミングが苦手な学生は部分記述形式でプログラム作成を行うだけでは，プログラム作成できるようにはならないことがわかった．また，学生に段階的に解説を与えることでプログラム作成できるようになることがわかった．そこで，部分記述形式において，予め与えられている部分，空欄部分，それぞれに対して，プログラムを書くときに考えるべきことを，学生の要求に応じて段階的に提供することにする．以下に学生に与える情報を示す.

(1) プログラムの全体像を理解するための情報

- プログラムの作成のアイデア

  プログラムの全体像を理解するための情報として，課題で指定されたプログラムを書くときの基本的なアイデアを学生に示す[3][4].

- プログラムの作成の方針

  プログラムの全体像を理解するための情報として，基本的なアイデアに基づいてどのようなプログラムを作成するのかという方針を学生に示す.

(2) プログラムの予め与えてある部分を理解するための情報

- プログラムの流れの説明

  プログラムの予め与えてある部分を理解するための情報として，空欄の部分も含めたプログラム全体の流れを学生に示す．プログラム数行のまとまりに対して情報を与える.

- プログラムの各行の説明

  プログラムの予め与えてある部分を理解するための情報として，プログラムの流れの説明を読んでも十分に理解できなかった場合のために，プログラムの各行を理解するために必要な情報を学生に示す．

(3) プログラムの空欄の部分に書くべきことを考えるための情報

- 空欄の部分に対するヒント

  プログラムの空欄の部分に書くべきことを考えるための情報として，空欄の部分に対するヒントを学生に示す．一度に多くの情報を提示すると理解しにくくなるので，学生の要求に応じて，必要な部分の情報のリストを示す．学生はリストを閲覧し，リストから自分に必要な詳しい情報を得ることができる．

- 空欄に記述すべきプログラムの流れを示した図

  文章で与えられた「空欄の部分に対するヒント」を理解できない学生のために，プログラムの空欄の部分に書くべきことを図を用いて，プログラムの動作をアニメーションで学生に示す[5]．

3.5 提案する演習形式におけるプログラム作成の流れ

提案する演習方式におけるプログラムの作成の流れを示す．

Step 1 学生は課題で示された問題文と実行例を読んで，どうようなプログラムを作成すべきかの概要を考える．

Step 2 作成するプログラムの概要が分からなければ，学生は「プログラムの作成のアイデア」を表示させる．基本的なアイデアだけでは分からなければ，それをもう少し詳しく説明した「プログラムの作成の方針」を表示させる．

Step 3 作成するプログラムの概要を理解した上で，学生はプログラムの予め与えてある部分を読んで理解する．

Step 4 プログラムの予め与えてある部分を理解することができなかっ

た場合には，学生は「プログラムの流れの説明」を表示させる．必要があれば，さらに詳しく説明した「プログラムの各行の説明」を表示させる．

Step 5　プログラムの予め与えてある部分を理解した上で，学生は空欄の部分に書くべき内容を考える．

Step 6　空欄の部分に書くべき内容が分からなければ，学生は「空欄の部分に対するヒント」「空欄に記述すべきプログラムの流れを示した図」を表示させ空欄にどういったプログラムを書いたらよいか理解する．

Step 7　学生は空欄にプログラムを書く．

以上の流れでプログラム作成を行うことにより，プログラミング初心者が新規の課題に取り組む場合でも，プログラム作成ができるようになると考えられる．

3.6　提案する提案方式で課題を行った場合の説明

本節では具体的な例を用いて提案する演習方式を説明する．ここでは，配列を用いたプログラムの作成課題の例として図 3.2 に示す課題を行う場合の説明を行う．また，システムでは学生の要求に応じて解説を与えるために，解説のボタンを押すことで解説を画面に表示する．

課題例

大きさ 10 の配列に整数値を 10 個読み込んでその中の最大値を表示するプログラムを，for 文を用いて記述しなさい。

実行例

整数値を 10 個入力してください。
整数 1:*15*
整数 2:*0*
整数 4:*2*
整数 5:*99*
整数 6:*14*
整数 7:*-15*
整数 8:*95*
整数 9:*15*
整数 10:*-11*
最大値は 99 です。

図 3.2 課題例

3.6.1 プログラムの雛形

図 3.3 に学生に与えるプログラムの雛形，図 3.4 に解答例を示す．この課題で重要となる箇所のプログラムは最大値を求める処理の部分である．そのため，最大値を求める処理の部分（22 行目から 23 行目の間）を空欄にし学生に記述させる．

プログラム 002

```
001   /*
002配列に10個の整数値を読み込んでその中の最大値を表示するプログラム
003   */
004
005   #include <stdio.h>
006
007   #define SIZE 10       /* 配列の大きさ */
008
009   int main(void)
010   {
011        int n[SIZE] ;    /* 読み込んだ値を格納するための配列 */
012        int i ;          /* 配列の添え字のための変数 */
013        int max;         /* 最大値を格納するための変数 */
014
015        /* 配列に整数値を10個読み込む。 */
016        printf("整数値を10個入力してください。¥n");
017        for ( i=0 ; i<SIZE ; i++ ) {
018             printf("整数%d:", i+1);
019             scanf("%d", &n[i]);
020        }
021
022        /* 最大値を求める。 */
```

【配列と for 文を用いて以下の空欄を埋めてプログラムを完成させて下さい。】

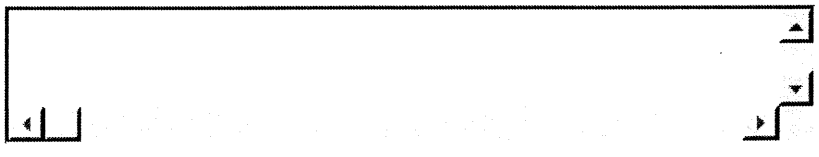


図 3.3 学生に与えるプログラムの雛形（1/2）

プログラム 002(つづき)

```
023
024         /* 最大値を表示する。 */
025         printf("最大値は%d です。¥n", max);
026
027         return (0);
028   }
```

図 3.3 学生に与えるプログラムの雛形 (2/2)

解答例

```
    max=n[0];
    for(i=1;i<SIZE;i++){
         if(max<n[i]) max=n[i];
    }
```

図 3.4 解答例

3-6.2　プログラム全体像を理解するための情報

図 3.5 にプログラムの作成のアイデア，図 3.6 にプログラム作成の方針を示す．プログラム作成のアイデアではどのようなプログラムを記述したらよいか大まかな手順を説明する．この課題では，図 3.5 に示すように，大きく 4 つの手順に分けることができる．プログラム作成の方針では，各手順の詳細について説明を行う．

プログラム作成のアイデア 表示する

(1) 必要な変数を宣言します．
(2) 配列に数値を格納します．
(3) 最大値を求めます。
(4) 最大値を表示します。

図 3.5 プログラムの作成のアイデア

プログラム作成の方針 表示する

(1) 最大値を格納する変数，大きさが 10 の配列，配列の添え字のための変数を用意します．
(2) for 文を用いて 10 個の整数値を配列に読み込みます。
(3) 最大値を求めます。
　(3.1)　読み込んだ値のうち 1 番目のものを仮の最大値とします。
　(3.2)　2 番目以降の値を順番に仮の最大値と比較します。仮の最大値よりも大きければ，それを新しい仮の最大値とします。
　(3.3)　2.2 の処理を 10 番目の値まで行えば，仮の最大値は 10 個の中で一番大きな値ですので，それが最大値です。
(4) 最大値を表示します。

図 3.6 プログラム作成の方針

3.6.3　プログラムの予め与えてある部分を理解するための情報

図 3.7 にプログラムの流れの説明の一部を示す．図 3.8 にプログラムの各行の説明の一部を示す．この課題で予め与えた部分のプログラムで配列，配列の添え字のための変数，求めた最大値を格納するための変数宣言が 11 行目から 13 行目で行われている．11 行目から 13 行目に対し，プログラムの流れの説明でまず必要最低限の説明（図 3.7）を与え，不足するならプログラムの各行の説明(図 3.8)では詳細な説明を与える．

プログラムの流れの説明

表示する

【005】 標準入出力関数が使えるようにヘッダファイルを読み込みます。

【011】 必要な配列を用意します。このプログラムで必要な配列は，読み込んだ 10 個の整数値を格納するための整数型の配列 1 つです。

【012～013】 必要な変数を用意します。このプログラムで必要な変数は，求める最大値を格納するための整数型の変数と，配列の添え字のための変数です．

【016～020】 配列 n に整数値を 10 個読み込みます。

【空欄】 最大値を求めます。

【025】 最大値を表示します。

【027】 プログラムを終了します。

図 3.7 プログラムの流れの説明

プログラムの各行の説明

【001～005】 表示する

【001～003】 プログラムの概要をコメントとして書きます。

【004】 プログラムを読みやすくするために，空白行を 1 行入れます。

【005】 出力のための printf 関数，入力のための scanf 関数などの標準入出力関数が使えるようにヘッダファイルを読み込みます。

【007～008】 表示する

【007】 配列の大きさをマクロとして定義します。配列の大きさを表す数値は何度もプログラム中に出てくるので，その数値に名前を付けるためです。名前を付けることにより，単に数値で表すよりも，プログラムが読みやすくなります。

【008】 プログラムを読みやすくするために，マクロの定義の後に空白行を 1 行入れます。

【009～0110】 表示する

【009～010】 main 関数の定義を始めます。

【011】 表示する

【011】 読み込んだ 10 個の整数値を格納するための配列として，大きさ 10 の配列 n を宣言します。配列の大きさの指定にはマクロ SIZE を用います。

【012～13】 表示する

【012】 配列の添え字のための変数として，整数型の変数 i を宣言します。

【013】 求める最大値を格納するための変数として，整数型の変数 max を宣言します。

【016～020】 表示する

【015】 処理のまとまりの最初にコメントを記します。

【016】 入力を促すメッセージを表示します。

図 3.8 プログラムの各行の説明

3.6.4　プログラムの空欄の部分に書くべきことを考えるための情報

図 3.9 に空欄の部分に対するヒントの一部，図 3.10 に空欄に記述すべきプログラムの流れを示した図を示す．この課題では，配列に値を格納するために，for 文を用いる必要がある．しかし，プログラムが苦手な学生は for 文の使い方がわからないことがある．そこで，空欄に対するヒントでは，for 文の使い方について具体的に解説（図 3.9）を与えた．また，空欄に記述すべきプログラムの流れを示した図（図 3.10）では，記述すべき箇所の変数の変化を図を用いて示した．

表示する

空欄に対するヒント

for 文の繰り返しを制御する変数の初期値を決めます。表示する

・2 番目以降の値を仮の最大値と比較したいので，初期値は 1 となります。
・配列の添え字が 0 から始まるのに注意して下さい。
・普通に何番目という場合と 1 つずれますので注意して下さい。

for 文の繰り返しの変数をいくつ更新するかを決めます。表示する

・2 番目から 10 番目まで順番に繰り返したいので，1 つずつ増加させます。
・2 つの値を比較するのには，比較演算子を用います。
・比較演算子のうち，「<」「>」「<=」「>=」のどれかを使います。このプログラムは，どれを用いても書くことができます。

図 3.9 空欄に対するヒント

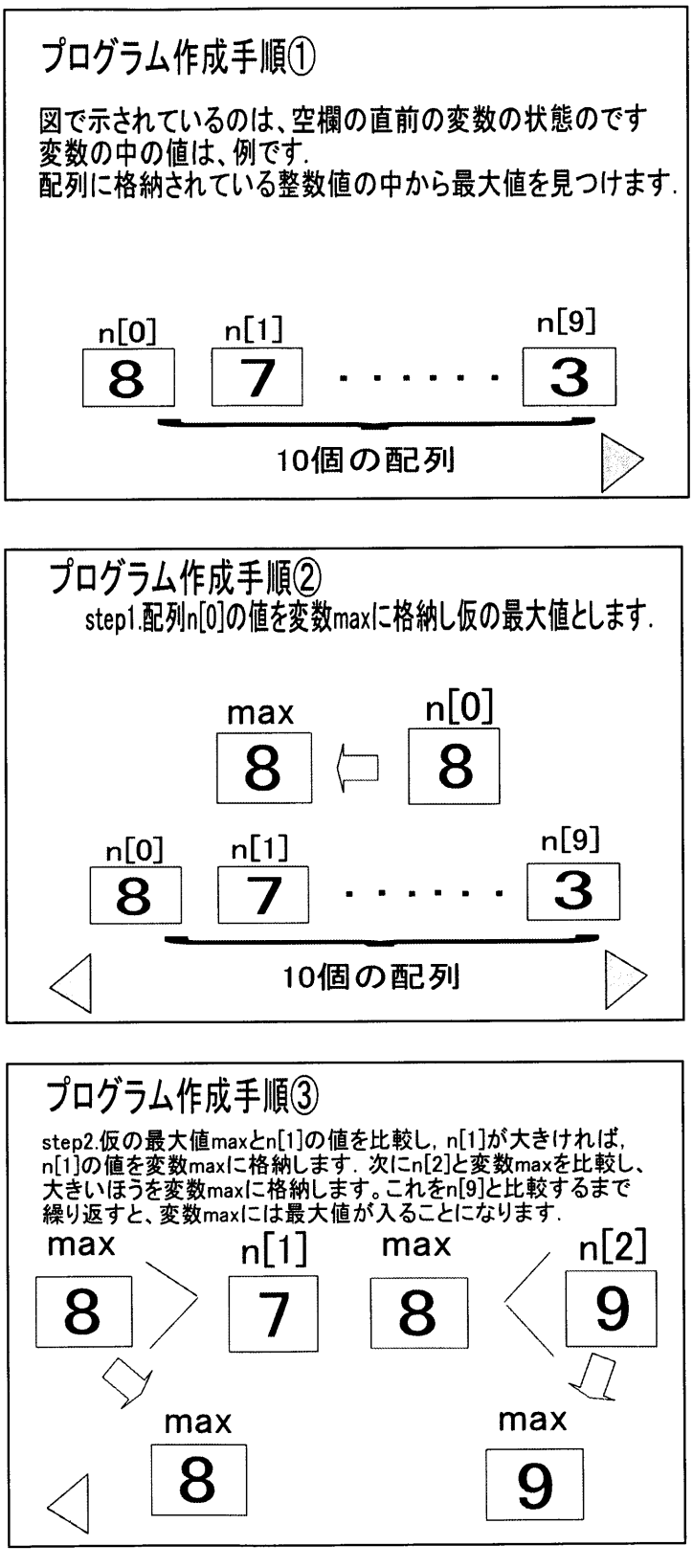


図 3.10 空欄に記述すべきプログラムの流れを示した図

第4章　実験と考察

本研究で提案した演習方式の有効性を確認するため，プログラミングが苦手な学生 6 名を対象に演習を行った．演習ではまず，学生に解説なしでプログラムを初めから記述させる従来の形式でプログラム作成に取り組んでもらった．従来の方式でプログラム作成できなければ，提案した演習方式で再度同じ課題に取り組んでもらった．表 4.1 に演習中に学生がプログラムを作成できた件数を示す．

- 対象 2006 年度プログラミング演習 I 受講者　　6 人
- 課題数 3 問
- 課題内容 C 言語の if 文，for 文，配列を用いたプログラムの作成
- 演習時間 90 分

表 4.1 実験で学生がプログラム作成できた件数

学生の力のみで作成できた件数	4件
提案した演習方式で作成できた件数	13件
提案方式でもプログラム作成できなかった件数	1件

表 4.1 に示したように従来形式でプログラムを作成できなかった学生の 14 人中 13 人が提案方式を用いることでプログラムを作成できるようになった．これにより，提案方式の有効性を確認することができた．表 4.2，表 4.3，表 4.4 に各解説に対するアンケート結果を示す．

表 4.2 アンケート例（プログラム全体像を理解するための情報）

・プログラム全体像を理解できなかった時，「プログラムの作成のアイデア」を見ることでわかりましたか？。

学生 A. プログラム作成のために一番初めにすべきことが理解できた.
学生 B. 自分の基礎力が足らず，全体像を理解できなかった.

・プログラム全体像を理解できなかった時，「プログラムの作成の方針」を見ることでわかりましたか？

学生 B. 全体像を理解することができた.

表 4.2 の結果より，学生 B は「プログラムの作成の方針」を読んだがプログラムの全体像を理解できなかったと回答したが，「プログラムの作成の方針」を閲覧することでで理解することができたと回答している．これより，プログラム全体像を理解するための情報を学生の要求に応じて段階的に解説を表示することが有効であることがわかった．

表 4.3 アンケート例（予め与えてある部分を理解するための情報）

・予め与えられたプログラムを理解できなかったとき，
「プログラムの流れの説明」を見ることでわかりましたか？

学生 C.　プログラムを書くのに非常に役立った.
学生 D. もっと詳しい説明がないと理解できなかった.

・ 「プログラムの各行の説明」を見ることで与えられた
プログラムをわかりましたか？

学生 D. 書くべきプログラムの流れと使うべき機能が わかった

表 4.3 の結果より，学生 D は「プログラムの流れの説明」を読んだが予め与えたプログラムを理解できなかったと回答したが，「プログラムの各行の説明」を閲覧することでで理解することができたと回答している．これより，予め与えてある部分を理解するための情報を学生の要求に応じて段階的に解説を表示することが有効であることがわかった．

表 4.4 アンケート例（空欄の部分に書くべきことを考えるための情報）

・空欄に記入すべきプログラムがわからなかった時,
「空欄に対するヒント」を見ることでわかりましたか？

学生 E.　プログラムを書くのに非常に役立った.
学生 F.　書いてあることをプログラムで表現できなかった.

・空欄に記入すべきプログラムがわからなかった時,
「空欄に記述すべきプログラムの流れを示した図」を
見ることでわかりましたか？

学生 E. プログラムがどのように動いているか理解することができた.

表 4.4 の結果より，学生 E は空欄に対するヒントと空欄に記述すべきプログラムの流れを示した図がプログラム作成に役立ったと解答している．このことから，空欄の部分に書くべきことを考えるための情報を与えることに効果があったことがわかる．

第５章　まとめ

本研究では，初心者，特にプログラミングが苦手な学生がプログラムの作成できるようにするために，部分記述形式の課題を用いて演習を行うことを提案した．提案する演習形式では，プログラムの予め与えてある部分を理解できるように，学生の要求に応じて，説明を提供する．また，学生の要求に応じて，部分記述形式の空欄の部分を書くときに考えるべきことを提供する．

提案した演習方式の有効性を確認するために，プログラミングが苦手な学生を対象に演習を行った．その結果，提案した演習方式でプログラム作成を行うことでプログラム作成を行えるようになった．

参考文献

[1] 河西朝雄,「C 言語プログラミング学習方程式」, 技術評論社, 2007

[2] 高橋 参吉, 久保田 和男, 松永 公廣, 橋本 はる美, 佐野 繭美,「Web 電子問題集を用いた C プログラミング演習」, 教育システム情報学会誌, 2000

[3] 高橋麻奈,「やさしいプログラミング」, ソフトバンククリエイティブ株式会社, 2006

[4] 杉浦英樹,「自分のペースでゆったり学ぶ C プログラミング超入門」, 技術評論社, 2006

[5] 小林麻衣子, 宮地弘子, 高橋誠,「C の絵本」, 翔泳社, 2006

業績一覧

[1] 川元健司，北英彦，高瀬治彦，林照峯，初心者向けプログラミング問題演習システム，コンピュータ利用教育協議会，2005 PC カンファレンス論文集，pp.337-338，2005

[2] 川元健司，北英彦，高瀬治彦，林照峯，初心者向けプログラミング問題演習システム，電気関係学会，平成 17 年度電気関係学会東海支部連合大会講演論文集，O-212，2005

[3] 北英彦，川元健司，北英彦，高瀬治彦，林照峯，初心者向けプログラミング演習システム，日本教育工学会，日本教育工学会第 21 回全国大会講演論文集，pp.437-438，2005

[4] 川元健司，北英彦，高瀬治彦，林照峯，初心者がプログラム作成に取り掛かりやすくするための試み，コンピュータ利用教育協議会，2006 PC Conference 論文集，pp.335-336，2006

[5] 川元健司，北英彦，高瀬治彦，林照峯，初心者がプログラム作成の課題に取り掛かり易くするための試み，計測制御学会，平成 18 年度三重地区計測制御研究講演会講演会論文集，P-18，2006

[6] Kawamoto Kenji, Hideyuki Kanza, Hidehiko Kita, Haruhiko Takase, Terumine Hayashi, Programming Exercise System with Automatic Test for Novice Programmers, The 14th International Conference on Computers in Education, Interactive Events2, Beijing, 2006

謝辞

本論文は，著者が三重大学大学院工学研究科博士前期課程に在籍中に行った研究をまとめたものである．本研究を進めるにあたり，懇切丁寧な御指導と御督励を賜った三重大学工学部電気電子工学科林照峯教授，北英彦助教授，高瀬治彦助手に感謝いたします．

また，貴重な時間をさいて本研究論文を査読して頂いた，三重大学工学部電気電子工学科森香津夫助教授に深く感謝いたします．

本研究で提案するプログラミング演習支援システムの評価のために，講義で実際に運用していただき，また，貴重なコメントをいただいた鶴岡信治教授，篠木剛助教授，森香津夫助教授に深く感謝いたします．

最後に，本研究論文をまとめるにあたり，助言，討論，試行実験など，お世話になった森田直樹氏，望月将行氏，計算機工学研究室の皆様方，他すべての人に感謝します．

付録

プログラミング演習の現状を調査するために用いた課題

問 1

下記のプログラムを書き換えて，先頭から順に 5,4,3,2,1 を代入せよ．

```
/*
配列の各要素に先頭から順に 1,2,3,4,5 を代入して表示(for 文)
*/
#include   <stdio.h>

int main(void)
{
        int i;
        int vc[5];

        for(i=0;i<5;i++)  {
        vc[i]=i+1;
        }
        for(i=0;i<5,i++){
        printf("vc[%d ]= %d¥n", i, vc[i]);
        return(0);
}
```

実行結果

```
vc[0]=1
vc[1]=2
vc[2]=3
vc[3]=4
vc[4]=5
```

問 2

下記のプログラムを書き換えて，配列 va の要素の並びを逆順にしたものを配列 vb にコピーせよ．

```
/*
配列の全要素を別の配列にコピー
*/
#include <stdio.h>
 int main(void)
{
        int   i;
        int va[5] ={15,20,30};  /*{15,20,30,0,0}で初期化*/
        int vb[5];
        for(i=0;i<5;i++){
                vb[i]=va[i];
        }

        puts("va   vb"):
        puts("----------");
        for(i=0;i<5;i++){
                printf("%3d%3d¥n", va[i],vb[i]);
        }
}
```

実行結果

```
va      vb
---------------
-
15      15
20      20
30      30
0       0
0       0
```

問 3

1 人分の小テストの成績を集計するプログラムを作成する.

****

- 小テストの回数は 10 回とする.
- 各回の小テストは 10 点満点とする.
- 小テストの点数は整数値とする.
- 小テストの点数の合計が 60 点以上を合格とする.

****

問 4

クラスの 1 回分の小テストの成績を処理するプログラムを作成する.

****

- 小テストは 10 点満点とする.
- 小テストの点数は整数値とする.
- 小テストの点数の合計が 6 点以上を合格とする.
- 受講者数は 10 人とする.

****

指導方法を調査するために用いた 7 つの課題

問題 1

整数値を読み込んで，読み込んだ数値が偶数ならそのまま表示し，奇数ならその数値を２倍した値を表示するプログラムを作成してください.

問題 2

A,B,2 つの整数値を読み込み、A が大きいなら、「A が大きいです」と表示し、B が大きいなら「B が大きいです」と表示、同じなら「同じです」と表示するプログラムを作成してください。

問題 3

二回さいころを振り，二回とも出た目が 5 以上なら「３点」，二回とも 5 未満なら「0 点」,どちらか一方が 5 以上なら「1 点」と表示するプログラムを作成してください．配列は使用しないでください.

問題 4

4 つの数字を読み込んで, もっとも小さい値表示するプログラムを作成してください。If を用いて作成してください.

問題 5

読み込んだ数字を，10 回表示するプログラムを作成してください。

問題 6

20 から 0 までカウントダウンを行うプログラムを作成してください。

動作結果

20 19 18 17・・・3 2 1 0

問題 7

下に示すように 20 から 30 までの和を計算し,表示するプログラムを作成してください。

動作結果

20 から 30 までの和は 275 です.

提案した演習方式の有効性を確認するために演習で用いた課題と解説

課題 001

整数値を 3 つ読み込んでその中の最大値を表示するプログラムを記述しなさい。

実行例

整数値を 3 つ入力してください。
整数 1:*47*
整数 2:*103*
整数 3:*8*
最大値は 103 です。

プログラムの作成のアイデア表示する

(1) 必要な変数を宣言します。
(2) 3 つの整数値を読み込んで, 変数に格納します。
(3) 最大値を求めます。
(4) 求めた最大値を表示します.

プログラムの作成の方針表示する

(1)　3 つの整数値を格納するための変数を 3 つ, 最大値を格納するための変数を 1 つ宣言します.
(2)　3 つの整数値を読み込みます。
(3)　最大値を求めます。
　(3.1)　読み込んだ整数値のうち最初の 2 つを比較します。大きい方を仮の最大値とします。
　(3.2)　3 つ目の整数値を仮の最大値と比較します。大きい方が 3 つの整数値の中で一番大きな値です。大きい方が求める最大値です。
(4)　最大値を表示します。

プログラム 001	プログラムの流れの説明	プログラムの各行の説明
001 /* 002 整数値を 3 つ読み込んでその中の最大値を表示するプログラム 003 */ 004 005 #include <stdio.h> 006 007 int main(void) 008 { 009 int n1,n2,n3; /* 読み込んだ値を格納するための変数 */ 010 int max; /* 最大値を格納するための変数 */ 011 012 /* 整数値を 3 つ読み込む。 */ 013 printf("整数値を 3 つ入力してください。¥n"); 014 printf("整数 1:"); scanf("%d", &n1); 015 printf("整数 2:"); scanf("%d", &n2); 016 printf("整数 3:"); scanf("%d", &n3); 017 018 /* 最大値を求める。 */ 【以下の空欄を埋めてプログラムを完成させて下さい。】 019 020 /* 最大値を表示する。 */ 021 printf("最大値は%d です。¥n", max); 022 023 return (0); 024 }	表示する 【001～005】 標準入出力関数が使えるようにヘッダファイルを読み込みます。 【007～010】 必要な変数を用意します。このプログラムで必要な変数は，読み込んだ 3 つの整数値を格納するための整数型の変数と，求める最大値を格納するための整数型の変数です。 【012～016】 整数値を 3 つ読み込みます。 【018＋空欄】 最大値を求めます。 【020～021】 最大値を表示します。 【023】 プログラムを終了します。	【001～005】 表示する 【001～003】 プログラムの概要をコメントとして書きます。 【004】 プログラムを読みやすくするために，空白行を 1 行入れます。 【005】 出力のための printf 関数，入力のための scanf 関数などの標準入出力関数が使えるようにヘッダファイルを読み込みます。 【007～010】 表示する 【007～008】 main 関数の定義を始めます。 【009】 読み込んだ 3 つの整数値を格納するための変数として，整数型の変数 n1,n2,n3 を宣言します。 【010】 求める最大値を格納するための変数として，整数型の変数 max を宣言します。 【012～016】 表示する 【012】 処理のまとまりの最初にコメントを記します。 【013】 入力を促すメッセージを表示します。 【014】 1 つ目の整数値の入力を促すためのメッセージを表示します。scanf 関数で変数 n1 に 1 つ目の整数値を読み込みます。 【015】 2 つ目の整数値の入力を促すためのメッセージを表示します。scanf 関数で変数 n2 に 2 つ目の整数値を読み込みます。 【016】 3 つ目の整数値の入力を促すためのメッセージを表示します。scanf 関数で変数 n3 に 3 つ目の整数値を読み込みます。 【018～019】 表示する 【018】 処理のまとまりの最初にコメントを記します。 【空欄】 空欄の部分に対するヒントを見てください。 【019】 プログラムを読みやすくするために，処理のまとまりの間に空白行を 1 行入れます。 【020～021】 表示する 【020】 処理のまとまりの最初にコメントを記します。 【021】 最大値を表示します。 【023～024】 表示する 【023】 プログラムを終了します。 【024】 main 関数の定義を終了します。

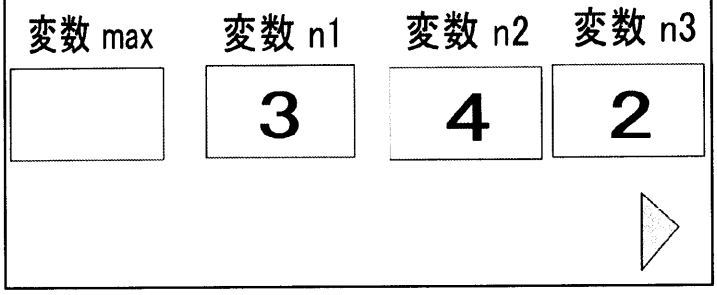
空欄の部分に対するヒント　表示する

● 読み込んだ整数値を 2 つずつ比較して最大値を求めます。	少し詳しい説明　表示する ● 3 つの値の中で一番大きなものを見つけたいのですから，3 つの値を比較する必要があります。 ● 3 つの値を同時に比較することはできませんので，2 つずつ比較することになります。
● 「仮の最大値」を格納するために変数 **max** を使います。	少し詳しい説明　表示する ● 求めた最大値は，変数宣言の部分で用意した変数 **max** に格納します。 ● 最大値を求める処理の途中での「仮の最大値」は，変数 **max** に格納することにします。
● まず，読み込んだ値のうち最初の 2 つを比較します。	少し詳しい説明　表示する ● プログラムの前半部で，1 つ目に読み込んだ値を変数 **n1** に，2 つ目に読み込んだ値を変数 **n2** に格納していますので，変数 **n1** と変数 **n2** の中の値を比較します。 ● 2 つの変数中の値を比較するのには，比較演算子を用います。 ● 比較演算子のうち，「**<**」「**>**」「**<=**」「**>=**」のどれかを使います。このプログラムは，どれを用いても書くことができます。 ● このプログラムでは，「**n1>n2**」「**n1<n2**」「**n1>=n2**」「**n1<=n2**」のどれかの形になります。
● 比較の結果によってプログラムの実行を分岐させます。	少し詳しい説明　表示する ● 比較した結果に従って，プログラムの実行を分岐させます。プログラムの実行を分岐させるには，**if** 文，**if-else** 文または **switch** 文を用います。 ● このプログラムでは，**if-else** 文を使う必要があります。 **if (**2 つの変数中の値を比較する条件式**) {** 条件が成り立ったときに実行する文（または文の並び） **} else {** 条件が成り立たなかったときに実行する文（または文の並び） **}** ● このプログラムでは，条件が成り立ったときと条件が成り立たなかったときの両方の場合で処理が必要ですので，**if** 文ではなく **if-else** 文になります。
● 比較した結果，大きい方を「仮の最大値」とします。	少し詳しい説明　表示する ● 大きい方の値を変数 **max** に代入します。 ● 条件が成り立ったときに実行する文は下記のようになります。 **max =** 大きい方の値が格納されている変数名 **;** ここで，大きい方の値が格納されている変数名は，条件式の書き方により，**n1** または **n2** のどちらかになります。 ● 条件が成り立たなかったときに実行する文は下記のようになります。 **max =** 大きい方の値が格納されている変数名 **;** ここで，大きい方の値が格納されている変数名は，条件式の書き方により，**n1** または **n2** のどちらかになります。 ● 空欄の部分のここまでの処理で，最初に読み込んだ 2 つ値のうち大きい方が「仮の最大値」として変数 **max** に格納されます。
● 3 つ目に読み込んだ値を「仮の最大値」と比較します。	少し詳しい説明　表示する ● プログラムの前半部で 3 つ目に読み込んだ値は変数 **n3** に，空欄の部分のここまでの処理で「仮の最大値」は変数 **max** に格納されています。 ● 変数 **n3** と変数 **max** の中の値を比較します。 ● 2 つの変数中の値を比較する方法については，上記の空欄の部分に対するヒントの「2 つの変数中の値を比較する」を参照して下さい。

● 比較した結果，大きい方が最大値です。 ● 空欄の部分で処理するのはここまでです。	少し詳しい説明 表示する ● 大きい方の値を変数 **max** に代入します。 ● 比較の結果によってプログラムの実行を分岐させる方法については，上記の空欄の部分に対するヒントの「比較の結果によってプログラムの実行を分岐する」を参照して下さい。 ● 条件が成り立ったときに実行する文は下記のようになります。 **max** = 大きい方の値が格納されている変数名 **;** ここで，大きい方の値が格納されている変数名は，条件式の書き方により，**n1** または **n2** のどちらかになります。 ● 条件が成り立たなかったときに実行する文は下記のようになります。 **max** = 大きい方の値が格納されている変数名 **;** ここで，大きい方の値が格納されている変数名は，条件式の書き方により，**n1** または **n2** のどちらかになります。 ● 空欄の部分のここまでの処理で，**3** つの値の中で一番大きなものが変数 **max** に格納されます。

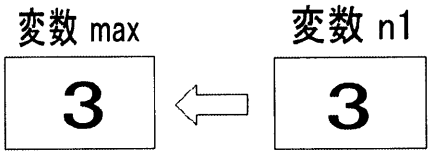
プログラム作成手順①

変数max,変数n1,変数n2,変数n3が宣言されています.
空欄の直前で, 変数n1に3,変数n2に4, 変数n3に2が格納されているとします.
変数maxには何が格納されているかわかりません.



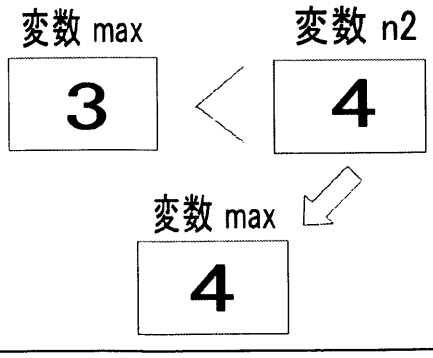
プログラム作成手順②

変数n1の値を仮の最大値にします. そこで,変数maxに変数n1の値を格納します.



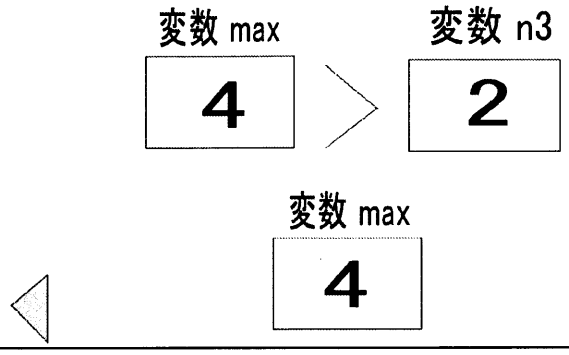
プログラム作成手順③

変数maxと変数n2の値を比較して, 大きい方の変数の値を変数maxに格納します.



プログラム作成手順④

変数maxと変数n3の値を比較して，大きい方の変数の値を変数maxに格納します．
変数maxの方が大きい場合は，そのままの状態で良いです．
以上の処理で変数maxには最大値が格納されます．



課題 002

大きさ 10 の配列に整数値を 10 個読み込んでその中の最大値を表示するプログラムを，for 文を用いて記述しなさい。

実行例

整数値を 10 個入力してください。
整数 1:*15*
整数 2:*0*
整数 4:*2*
整数 5:*99*
整数 6:*14*
整数 7:*-15*
整数 8:*95*
整数 9:*15*
整数 10:*-11*
最大値は 99 です。

プログラムの作成のアイデア表示する

(1) 必要な変数を宣言します。
(2) 配列に数値を格納します。
(3) 最大値を求めます。
(4) 最大値を表示します。

プログラムの作成の方針表示する

(1) 最大値を格納する変数，大きさが 10 の配列，配列の添え字のための変数を用意します。
(2) for 文を用いて 10 個の整数値を配列に読み込みます。
(3) 最大値を求めます。
    (3.1) 読み込んだ値のうち 1 番目のものを仮の最大値とします。
    (3.2) 2 番目以降の値を順番に仮の最大値と比較します。仮の最大値よりも大きければ，それを新しい仮の最大値とします。
    (3.3) 2.2 の処理を 10 番目の値まで行えば，仮の最大値は 10 個の中で一番大きな値ですので，それが最大値です。
(4) (4) 最大値を表示します。

プログラム 002

```
001   /*
002 配列に 10 個の整数値を読み込んでその中の最大値を表示するプログラム
003   */
004
005   #include <stdio.h>
006
007   #define SIZE 10      /* 配列の大きさ */
008
009   int main(void)
010   {
011       int n[SIZE] ;   /* 読み込んだ値を格納するための配列 */
012       int i ;         /* 配列の添え字のための変数 */
013       int max;        /* 最大値を格納するための変数 */
014
015       /* 配列に整数値を 10 個読み込む。 */
016       printf("整数値を 10 個入力してください。¥n");
017       for ( i=0 ; i<SIZE ; i++ ) {
018           printf("整数%d:", i+1);
019           scanf("%d", &n[i]);
020       }
021
022       /* 最大値を求める。 */
          【以下の空欄を埋めてプログラムを完成させて下さい。】
          【配列と for 文を用いて記述して下さい。】
023
024       /* 最大値を表示する。 */
025       printf("最大値は%d です。¥n", max);
026
```

プログラムの流れの説明

表示する

【001～005】 標準入出力関数が使えるようにヘッダファイルを読み込みます。

【007～008】
配列の大きさをマクロとして定義します。

【009～013】 必要な変数を用意します。このプログラムで必要な変数は，求める最大値を格納するための整数型の変数と配列の添え字のための変数,さらに読み込んだ 10 個の整数値を格納するための整数型の配列 1 つです.

【015～016】
処理のまとまりの最初にコメントを記します

プログラムの各行の説明

【001～006】 表示する
【001～003】 プログラムの概要をコメントとして書きます。
【004】 プログラムを読みやすくするために，空白行を 1 行入れます。
【005】 出力のための printf 関数，入力のための scanf 関数などの標準入出力関数が使えるようにヘッダファイルを読み込みます。

【007～008】 表示する
【007】 配列の大きさをマクロとして定義します。配列の大きさを表す数値は何度もプログラム中に出てくるので，その数値に名前を付けるためです。名前を付けることにより，単に数値で表すよりも，プログラムが読みやすくなります。
【008】 プログラムを読みやすくするために，マクロの定義の後に空白行を 1 行入れます。

【009～013】 表示する
【009～010】 main 関数の定義を始めます。
【011】 読み込んだ 10 個の整数値を格納するための配列として，大きさ 10 の配列 n を宣言します。配列の大きさの指定にはマクロ SIZE を用います。
【012】 配列の添え字のための変数として，整数型の変数 i を宣言します。
【013】 求める最大値を格納するための変数として，整数型の変数 max を宣言します。

【015～016】 表示する
【015】 処理のまとまりの最初にコメントを記します。
【016】 入力を促すメッセージを表示します。

【017～020】 表示する

【017～020】 for 文を用いて，10 個の整数値を大きさ 10 の配列 n に読み込みます。

【017】 for 文を 0～9 まで 10 回繰り返します。

- i=0　　配列の添え字を表す変数 i の初期値を 0 することを表します。
- i<SIZE　　変数 i が SIZE より小さいときに繰り返すことを表します。マクロ SIZE が表す数値は 10 ですので，変数 i が 10 より小さい，すなわち 9 まで繰り返すことを表します。
- i++　　繰り返すときに，変数iの値を 1 つ増加させることを表します。「i++」は「i=i+1」と同じ意味です。

【018】 i+1 番目の整数値を入力してもらうように促します。変数 i の値は 0～9 となるので「整数 1:」～「整数:10」のように表示するためには，「i」ではなく「i+1」にする必要があります。

【019】 i+1 番目の整数値を配列の添え字 i の要素に読み込みます。配列の添え字は 0 から始まるため，番号を 1 から始める場合に 1 つずれることに注意しましょう。

【020】 for 文の文末を示します。

【022～023】 表示する

【022】 処理のまとまりの最初にコメントを記します。

【空欄】 空欄の部分に対するヒントを見てください。

【024～025】 表示する

【024】 処理のまとまりの最初にコメントを記します。

【025】 最大値を表示します。

【027～028】 表示する

【027】 プログラムを終了します。

【028】 main 関数の定義を終了します。

空欄の部分に対するヒント 表示する

● for 文の繰り返しを制御する変数の初期値を決めます。	少し詳しい説明 表示する ● 2 番目以降の値を仮の最大値と比較したいので，初期値は 1 となります。 ● 配列の添え字が 0 から始まるのに注意して下さい。 ● 普通に何番目という場合と 1 つずれますので注意して下さい。
● for 文の繰り返しの変数をいくつ更新うるかを決めます。	少し詳しい説明 表示する 2 番目から 10 番目まで順番に繰り返したいので，1 つずつ増加させます。 ● 2 つの値を比較するのには，比較演算子を用います。 ● 比較演算子のうち，「<」「>」「<=」「>=」のどれかを使います。このプログラムは，どれを用いても書くことができます。 ● このプログラムでは，下記のどれかの形になります。 ➢ n1>n2 ➢ n1<n2 ➢ n1>=n2 n1<=n2

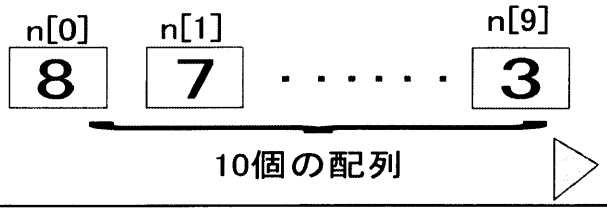
● 比較の結果によってプログラムの実行を分岐させる。	少し詳しい説明 表示する
	● 比較した結果に従って，プログラムの実行を分岐させます。プログラムの実行を分岐させるには，if 文，if-else 文または switch 文を用います。
	● このプログラムでは，if-else 文を使う必要があります。
	if (2 つの変数中の値を比較する条件式) {
	条件が成り立ったときに実行する文(または文の並び)
	} else {
	条件が成り立たなかったときに実行する文(または文の並び)
	}
	● このプログラムでは，条件が成り立ったときと条件が成り立たなかったときの両方の場合で処理が必要ですので，if 文ではなく if-else 文になります。
	● 比較した結果，大きい方を「仮の最大値」とします。大きい方の値を変数 max に代入します。

● 仮の最大値を格納する。	少し詳しい説明 表示する 条件が成り立ったときに実行する文は下記のようになります。 max = 大きい方の値が格納されている変数名 ; ここで，大きい方の値が格納されている変数名は，条件式の書き方により，n1 または n2 のどちらかになります。 ● 条件が成り立たなかったときに実行する文は下記のようになります。 max = 大きい方の値が格納されている変数名 ; ここで，大きい方の値が格納されている変数名は，条件式の書き方により，n1 または n2 のどちらかになります。 ● 空欄の部分のここまでの処理で，最初に読み込んだ 2 つ値のうち大きい方が「仮の最大値」として変数 max に格納されます。

● 3 つ目の値を仮の最大値と比較する。	少し詳しい説明 表示する ● 3 つ目に読み込んだ値を「仮の最大値」と比較します。 ● プログラムの前半部で 3 つ目に読み込んだ値は変数 n3 に，空欄の部分のここまでの処理で「仮の最大値」は変数 max に格納されています。 ● 変数 n3 と変数 max の中の値を比較します。 ● 2 つの変数中の値を比較する方法については，上記の空欄の部分に対するヒントの「2 つの変数中の値を比較する」を参照して下さい。
● 最大値を求める。	少し詳しい説明 表示する ● 比較の結果によってプログラムの実行を分岐させる方法については，上記の空欄の部分に対するヒントの「比較の結果によってプログラムの実行を分岐する」を参照して下さい。 ● 条件が成り立ったときに実行する文は下記のようになります。 `max = 大きい方の値が格納されている変数名 ;` ここで，大きい方の値が格納されている変数名は，条件式の書き方により，n1 または n2 のどちらかになります。 ● 条件が成り立たなかったときに実行する文は下記のようになります。 `max = 大きい方の値が格納されている変数名 ;` ここで，大きい方の値が格納されている変数名は，条件式の書き方により，n1 または n2 のどちらかになります。 ● 空欄の部分のここまでの処理で，3 つの値の中で一番大きなものが変数 max に格納されます。 ● すなわち，3 つの値の最大値が変数 max に格納されます。 ● 空欄の部分で処理するのはここまでです。

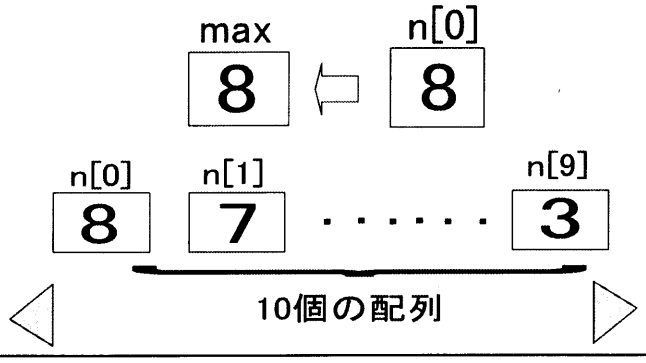
プログラム作成手順①

図で示されているのは、空欄の直前の変数の状態のです
変数の中の値は、例です.
配列に格納されている整数値の中から最大値を見つけます.



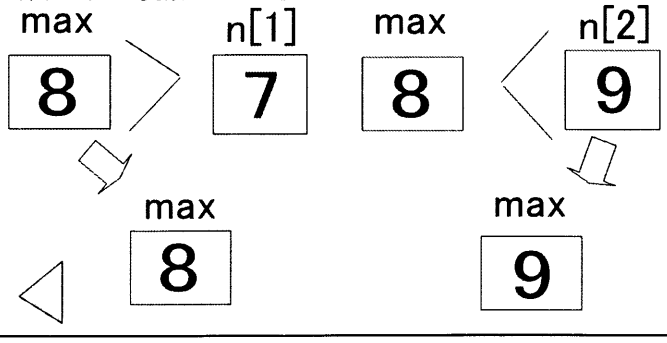
プログラム作成手順②

step1.配列n[0]の値を変数maxに格納し仮の最大値とします.



プログラム作成手順③

step2.仮の最大値maxとn[1]の値を比較し, n[1]が大きければ, n[1]の値を変数maxに格納します. 次にn[2]と変数maxを比較し、大きいほうを変数maxに格納します。これをn[9]と比較するまで繰り返すと、変数maxには最大値が入ることになります.



課題 003

for 文を用いて，読み込んだ整数値を 10 回表示するプログラムを記述してください。

実行例

数字を入力してください.
5
5
5
5
5
5
5
5
5
5

プログラムの作成のアイデア 表示する

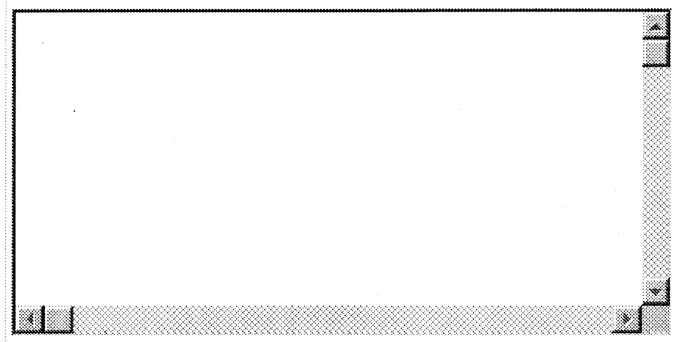
(1)必要な変数を用意します.
(2)for 文を用いて，同じ文を 10 回繰り返します。

プログラムの作成の方針 表示する

(1) 整数値を 1 つ読み込みます。
(2) その整数値を表示します。
(3) for 文を用いて，(2) を 10 回繰り返します。

プログラム 003

```
001  /*
002       入力された数字を繰り返し 10 回表示するプログラム
003  */
004
005  #include <stdio.h>
006
007  int main(void)
008  {
009       int n;         /* 読み込んだ値を格納するための変数 */
010       int i;         /* 繰り返しを制御するための変数 */
011
012       /* 整数値を 1 つ読み込む。 */
013       printf("整数値を 1 つ入力してください。¥n");
014       scanf("%d", &n);
015
016        /* 10 回繰り返し表示する。 */
          【以下の空欄を埋めてプログラムを完成させて下さい。】
          【for 文を用いて記述してください. 】
017
```



```
018       return (0);
019  }
020
```

プログラムの流れの説明

表示する

【005】 標準入出力関数が使えるようにヘッダファイルを読み込みます。

【007～010】 必要な変数を用意します。このプログラムで必要な変数は，読み込んだ整数値を格納するための整数型の変数 1 つと，繰り返しを制御するための変数するための整数型の変数 1 つです。

【012～015】 整数値を 1 つ読み込みます。

【016＋空欄】 for 文を用いて，同じ値を 10 回繰り返し表示します。

【018】 プログラムを終了します。

プログラムの各行の説明

【001～006】 表示する

【001～003】 プログラムの概要をコメントとして書きます。
【004】 プログラムを読みやすくするために，空白行を 1 行入れます。
【005】 出力のための printf 関数，入力のための scanf 関数などの標準入出力関数が使えるようにヘッダファイルを読み込みます。
【006】 プログラムを読みやすくするために，空白行を 1 行入れます。

【007～010】 表示する

【007～008】 main 関数の定義を始めます。
【009】 読み込んだ値を格納するための変数として，整数型の変数 n を宣言します。
【010】 繰り返しを制御するための変数として，整数型の変数 i を宣言します。

【012～015】 表示する

【012】 処理のまとまりの最初にコメントを記します。
【013】 入力を促すメッセージを表示します。
【014】 値の入力を促すためのメッセージを表示します。scanf で変数 n1 に 1 つ目の整数値を読み込みます。
【015】 プログラムを読みやすくするために，処理のまとまりの間に空白行を 1 行入れます。

【016～017】 表示する

【016】 処理のまとまりの最初にコメントを記します。
【空欄】 空欄の部分に対するヒントを見てください。
【017】 プログラムを読みやすくするために，処理のまとまりの間に空白行を 1 行入れます。

【018～020】 表示する

【018】 プログラムを終了します。
【019】 main 関数の定義を終了します。

空欄の部分に対するヒント 表示する

- for 文を用います。

少し詳しい説明 表示する （＊1）

- for 文は以下の形式で記述します.

```
for( 前処理; 制御式 ; 後始末 ){
    繰り返す文の並び
}
```

- for 文の前処理には，繰り返しを制御するための変数 i に初期値を設定するための代入文を記述します。
- for 文の条件式には，もう一度繰り返すかどうかを決めるための変数 i に関する条件を記述します。
- for 文の後始末には，繰り返しを制御するための変数 i を更新する代入文を記述します。
- for 文の本体には，繰り返す文の並びを記します。

- for 文の前処理には，繰り返しを制御するための変数 i に初期値を設定するための代入文を記述します。

少し詳しい説明 表示する （＊1の説明を読んでから表示させてください）

- for 文を用いて 10 回繰り返したいので，繰り返しを制御するための変数 i の値を，1 から 10，または，0 から 9 のように変化させる必要があります。
- 変数 i の値を 1 から 10 に変化させる場合には，変数 i に初期値 1 設定するための代入文を記述します。具体的には以下のように記述します。

```
for( i=1 ; 制御式 ; 後始末 ){
    繰り返す文の並び
}
```

- 変数 i の値を 0 から 9 に変化させる場合には，変数 i に初期値 0 設定するための代入文を記述します。具体的には以下のように記述します。

```
for( i=0 ; 制御式 ; 後始末 ){
    繰り返す文の並び
}
```

- for 文の条件式には，もう一度繰り返すかどうかを決めるための変数 i に関する条件を記述します。

少し詳しい説明 表示する （＊1の説明を読んでから表示させてください）

- for 文を用いて 10 回繰り返したいので，繰り返しを制御するための変数 i の値を，1 から 10，または，0 から 9 のように変化させる必要があります。
- 変数 i の値を 1 から 10 に変化させる場合には，変数 i の値が 10 以下のときに繰り返したいので，条件式は「i<=10」となります。

```
for( 前処理 ; i<=10 ; 後始末 ){
    繰り返す文の並び
}
```

- 変数 i の値を 0 から 9 に変化させる場合には，変数 i の値が 10 未満のときに繰り返したいので，条件式は「i<10」となります。

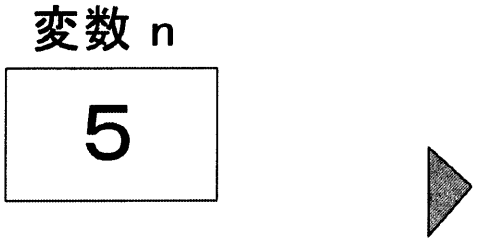
```
for( 前処理 ; i<10 ; 後始末 ){
    繰り返す文の並び
}
```

● for 文の後始末には，繰り返しを制御するための変数 i を更新する代入文を記述します。	少し詳しい説明 表示する （＊1の説明を読んでから表示させてください）
	● for 文を用いて 10 回繰り返したいので，繰り返しを制御するための変数 i の値を，1 から 10，または，0 から 9 のように変化させる必要があります。
	● 変数 i の値を 1 から 10 に変化させる場合には，変数 i の値を 1 つずつ増加させたいので，後始末は「i++」となります。
	for(前処理 ; 条件式 ; i++){
	繰り返す文の並び
	}
	● 変数 i の値を 0 から 9 に変化させる場合には，変数 i の値を 1 つずつ増加させたいので，後始末は「i++」となります。
	for(前処理 ; 条件式 ; i++){
	繰り返す文の並び
	}

● for 文の本体には，繰り返す文の並びを記します。	少し詳しい説明 表示する （＊1の説明を読んでから表示させてください）
	● この課題の場合には，読み込んだ整数値を表示させたいので，読み込んだ整数値を読み込んである変数 n の値を printf 関数を用いて表示させます。具体的には，以下のように記述します。
	for(前処理 ; 条件式 ; 後始末){
	printf(“%d¥n”, n) ;
	}

プログラム作成手順①

空欄の直前では，変数nに整数値が格納されています．
例えば，5が変数nに格納されているとします．



プログラム作成手順②

for文を用いて変数nに格納されている値5を繰り返し表示します．

for文とは、記述されたプログラムを繰り返し
実行する機能をもっています．
この課題では、printfで変数を表示するプログラムを
for文で繰り返し実行すれば、目的どおりの動作をおこないます．