ソフトウェア期末レポート

Report

提出日：　２００９年2月１２日

所属：　情報工学部一年次

学籍番号：　０９５７４５Ｄ

氏名：　西島本　純

問題１

設問１
from to len の値が負でないという条件は、プログラムに明示されている。

したがって a には配列 M の添字の値が上限を超えないことを判定する式が入ればよい。空欄 b、c は、具体的なケースをあげて検討する。

from から(from+len-1)までを to から(to+len-1)まで複写する。一方配列の最後は size-1 である。したがって、(from+len-1)および(to+len-1)がいずれも、配列の最後より前でないと行けないので○○＜＝size をつくる。従って、and で二つの条件を結合する。

”to=from+1”に設定すると、”M[from]→M[to]=M[from+1]”となり、”W”が”M[to]=M[from+1]”に複写される。次に、"M[from+1]"を”M[to+1]”に複写するのであるが、”M[from+1]”が”W”となっているため、”M[to+1]=M[from+2]”が”W”となる。このように、次々と”W”が複写元に複写されてしまうため、すべてが”W”となってします。したがって、”to”の値を”from+1”としたとき、このような状況が発生する。

空欄 b の状況を一般的に考えればよい。問題文に説明されているように、複写先として格納された要素が、あとで複写元の要素として読み出される場合である。これは、次のように複写元と複写先が重なる場合である。from と to が同じ場合は、何も変化がない。以上から、”from<to<from+len”であれば、このような状況が発生する。これは、”from<to and to<from+len”である。

設問２
設問１の空欄 c の検討で、複写先の配列と複写元の配列が全く重なれば問題はない。２つの条件である”to<=from”と”from+len<=to”のいずれかが成立すればよいので、解答の or をつかった"from+len<=to or to<=from"が正解となる。

”from+len>size”が循環していることを示している。したがって、空欄 e は from より右側に to が存在しない場合を考えればよい。まず”from+len-size<=to”である。また、from と to が重なる状態で、複写先の to はこれより左側、すなわち”to<=from”である。この２つの条件は and なので合致する"from+len-size<=to and to<=and to<=from"となる。

問題２

設問１
空欄 ab は QuickSort を再起的に呼び出すのに、配列の範囲を指定するためのパラメタの指定である。空欄 cd は pivot が見つかった場合で Pivot の位置を Ret に格納する処理である。範囲を指定して QuickSort を再起的に呼び出しているが、分割した左半分から整列している。これは空欄 a を含む QuickSort の呼び出しの範囲の先頭の要素番号が Min となっていることからわかる。QuickSort の先頭で Findpivot を呼び出したときに、J＝１が戻される。

これは、問題文表４の１回目では

3	5	8	4	0	6	9	1	2	7

”A[Min+1]から A[Max]まで、A[Min]と値が異なる要素を順次探し出し、最初に見つかった要素の値と A[Min]の値の打ち大きい方を基準値(Pivot)として選ぶ”という記述からわかる。次に、J＞－１が成立するので、Arrange を呼び出す。左側が Pivot より小さい値、右側が Pivot 以上の値をもつ要素が並ぶ。そこで、戻り値 Ret に代入している”L”を検討する。Pivot より小さい値は５つなので、左側が小さい要素だとわかる。A[Min]…A[i-1]の値が Pivot 未満となり、A[i]、…A[Max]の Pivot 以上となるように並べ替え、i の値を返すから、右側の範囲の開始位置が戻り値となる。

位置０	１	２	３	４	５	６	７	８	９
		小さい			5		以上		

Arrange から戻ると、”L←K－１”で、L には左側の最後の要素位置”４”が格納されるので、QuickSort(A[],Min,L) として左側に整列する。左側の整列が完了すると右側の整列であるが、最初の要素の値は K に格納されている。すなわち、QuickSort(A[],K,Max)として右側を整列する。

findpivot では、最初に”pivot← A[Max]”によって、左側の要素を Pivot に格納している。次に”K←Min＋１”とした後、A[Min＋１]から A[Max]にむかって pivot と等しくない値を探している。pivot は大きい方を選択するため、空欄 c は"Ret←K"が入り、空欄 d は Pivot が移動していないので、"Ret←Min"がはいる。

設問２

２回目で左側のすべての要素がそろったことになるので、次に A[Min]＝A[０]と A[４]を比較して、右側は変化しないので、

0	2	1	4	3	6	9	8	5	7

となる。

Arrange で Ret に３を返すので、 QuickSort に戻ると K＝３となっている。そこで、次の処理によって L に K－１の値、２となり３回目の QuickSort が呼ばれる。この時は、Min=0,Max=2 となっている。

問題３

fscanf()関数の変換指定子は％ld なので、４バイト整数値の入力である。プログラムにおいて４バイト整数（long int）型の変数は psiz だけなので、ここには psiz のポインタを入れることになる。また、fscanf()関数の制約によりアドレスなので&psiz となる。

while ループ終了後には、入出力ファイルのクローズ処理があるだけである。ということは、while ループを終了する条件がファイルの終わりであるということがすぐわかる。しかし、入力関数は空欄 b の直後の fgets だけである。解凍群を見ても psiz に関連するこ

とだけである。そこで、必要な回数の処理が終わった場合であると確信する。したがって、ここにはループカウンタなどが入ることがわかる。一方、psiz の初期値は実行形式ファイルのサイズであり、while ループ内では 1 回ごとにデクリメントされているので、継続条件として psiz＞0 が入ると見当がつくが、空欄が 3 カ所あるので確認が必要である。２つ目の空欄 b では、テキストファイルの 1 行分（６０文字）かテキストファイルのすべての文字が終了するまでの処理をすればよいので、psiz>0 でよい。また、３つ目の空欄 b では、テキストファイルの文字を 1 文字ずつシフトしながら空欄 d で OR 演算を行うので、やはり psiz＞0 の間、文字を操作すればよい。

配列 tbuf[]内の文字位置を示すための変数が入る。tbuf[]は、テキスト形式ファイルか 1 行分のデータを読み取るために使用されている。一方、解答群を見ると、p しか使われていない。そこで、p は 1 行入力するごとに初期値 0 とし、終了するときに入力行の文字数となればよい。初期化の p=0;はあるので、p をインクリメントする処理が必要である。したがって、p++が入る。

バイナリデータを組み立てる処理である。変数 pch は、ファイルに出力されていることから、バイナリデータであることがわかる。一方、tch はテキストデータのオフセット値である。すると、前述したような方法でバイナリデータを求めているということがわかる。前述したとおり、ここには、OR 演算が入る。したがって、｜＝が入る。

なお、tch>>(8-s)は右シフト、s の初期値は４で、２ずつインクリメントされる。２文字目は４ビット右シフト、３文字目は２ビット右シフト、４文字目は 0 ビット右シフトとなる。

問題４

設問 1

最初に GerLine で 1 行分を編集し、 1 行分を page[][]に格納した後、、”Gp>Gyo”ならば、出力している。そして、Gp をインクリメントが直前にあり、Gyo は主プログラムから渡されている定数である。Gp をインクリメントするごとに 1 ページ分まとまったら出力しているとわかる。

したがって、空欄 a では、 1 行分を page[][]に出力するのに、Text[]の開始位置が終了 1 が必要である。page[Gp][Mp]←Text[Sp]は、Text[]の 1 文字を page[][]に移している。その後、Sp と Mp はインクリメントして Gp は変化させていない。なので、 1 行分、すなわち Gp 行目の 1 行を編集するために Text[]の添字を、Sp から Ep まで変化させればよい。Text[Sp]は 1 行分の開始位置、Text[Ep]は 1 行目の終了位置にである。ループさせる条件は Sp＝＞Ep となる。

GetLine では、編集後の 1 行の行末の Text[]における位置を、Ep に格納する。そして Ep をインクリメントしているので終了位置を探していると予想する。ループの条件 Ep<N and LOOP を見て、その終了条件であるピリオドと最大文字数を判定しているのだと考える。つまり Ep＜（Sp＋Moji ー 1）を and で結合すればよい。

設問２

２段に出力している変数はDpである。Dpは１のとき処理を行い、２のときPageOutを呼び出していて、交互に切り替えて制御している。２段になったときの文字数を格納する必要がある。２段になった時の条件は、”１段目と２段目の間は２文字あける”のだから、mojiから２を引いて２で割った値が、格段の文字数となる。Dan←（moji-2）/2ならばかならず割り切れる。

空欄edは２つ入ることをまず考慮する。その他の処理は１行の文字数が違うだけで変更前と変わらない。Mpを設定する処理が必要であるのでYohakuを考慮して空欄dにはMp←Yohaku+1が入る。空欄eは２段目の出力時に、移動先の最初の位置をMpにセットする必要がある。Yohaku+1団目の文字数+空白２文字、ずらさなければならないのでMp＝Yohaku+Dan+3が入る。

PageOutなのでいずれも行数が入る。Dp＝１は１段目のみ出力する。これは変更前と同じなので、Gp−１が入る。しかしGpは実際の行数よりも１多いことに気をつける。空欄gについては、GpとGyoの関係に気をつける。Gyoは１ページの最大行数である。すると、２段目になっているということは、１段目のGyoまで使っているということである。

問題５

プログラムを解読して、①右が道ならばそこに進む②壁であれば左９０度向く直す、を繰り返していることに気づけば解くことができる。

設問１

ウエは開始してすぐ左折するような選択肢である。右の判定しているのですぐに消すことができる。アイの変化は５番目の⑤か④にある。③の地点到達した時点で右を判定し、右に進むので正解はアであるとわかる。

設問２

プログラムの特性から左回りすることはすぐにわかる。ここで、始点から１週回ってきたあと①の地点に達したときの動きに注目する。２１と２２行目のif文のなかに始点のコードとしてENTRANCEが入っていないので、壁のように扱います。その点に注意してウを選択する。

設問３

解答群を見ると２９の直後はすべてに入っているので、２０と２３と２８の違いを検証する。始点を印字できないイウは不適当である。そこからアが正解とわかる。

設問４

関数lcheck()は変更前とは逆に左を判定して左折する。つまりdirを進行方向に対して左折方向にセットする必要があるので、行番号２３と２８をdir=(dir+3)%4に書き換えればよい。