基本情報技術者　公開セミナー「午後試験対策学習法」

1. 基本情報技術者試験の性格と概要

試験の概要

情報処理技術者試験は、「情報処理の促進に関する法律」に基づき経済産業省が、情報処理技術者としての「知識・技能」の水準がある程度以上であることを認定している国家試験です。

情報システムを構築・運用する「技術者」から情報システムを利用する「エンドユーザ（利用者)」まで、IT に関係するすべての人に活用いただける試験として実施しています。特定の製品やソフトウェアに関する試験ではなく、情報技術の背景として知るべき原理や基礎となる技能について、幅広い知識を総合的に評価しています。

試験の目的

○情報処理技術者に目標を示し、刺激を与えることによって、その技術の向上に資すること。
○情報処理技術者として備えるべき能力についての水準を示すことにより、学校教育、職業教育、企業内教育等における教育の水準の確保に資すること。
○情報技術を利用する企業、官庁などが情報処理技術者の採用を行う際に役立つよう客観的な評価の尺度を提供し、これを通じて情報処理技術者の社会的地位の確立を図ること。

注）上記は http://www.jitec.ipa.go.jp からの引用です。

2. 評価と資格取得のメリット

対外的な評価

“システムの仕事に役立つ資格”を調査した「2011 年版いる資格，いらない資格」(日経コンピュータ掲載)によると，基本情報技術者は IT ベンダーの技術者が取得すべき資格の第 2 位に，また，IT ベンダーの営業担当者･ユーザ企業のシステム担当者が取得すべき資格の第 1 位にランキングされています。IT ベンダーの人事担当者の半数は，技術者はもちろんのこと，営業担当者にも基本情報技術者を取得させたいと回答しており，システムを発注する側のユーザ企業においても，取得の必要性が高い資格であることがうかがえます。

資格取得のメリット

(1) 情報処理技術者としてのスキルを有していることが公的に証明される。
(2) ＩＴ業界で活躍する方が，現在どのレベルの専門知識を有するかが立証される。
(3) 上級資格受験のための基礎を習得できる。
(4) 就職活動を行う上でのアピール材料となる。
(5) 資格手当や一時金など報奨金制度，昇級条件，（学生さんなら）履修の単位や特待生など待遇面で優遇

3. 基本情報技術者試験の出題と形式　～何が問われるか～

(1) 午前試験

試験時間：２時間 30 分

出題形式：マークシートで四択 80 問，全問必須(1.25 点×80＝100 点)

合格基準：100 点満点の 60% 以上の正解で合格

出題比率

テクノロジ		マネジメント		ストラテジ
50	:	10	:	20

(2) 午後試験

試験時間：２時間 30 分

出題形式：マークシートで多肢選択式(複数個の選択肢から 1 個以上を選択)

13 問中 7 問を解答(次表を参照)

合格基準：100 点満点の 60% 以上の正解で合格

出題形式：長文問題形式

出題内容

問番号	テーマ		配点	解答数・出題数
1～4	ハードウェア	5分野から4問を出題	各12点	5問選択/7問出題（12×5＝60点）
	ソフトウェア			
	データベース			
	ネットワーク			
	情報セキュリティ			
5	ソフトウェア設計			
6	マネジメント			
7	ストラテジ			
8	データ構造とアルゴリズム		20点	必須（20×1＝20点）
9	C		各20点	1問選択/5問出題（20×1＝20点）
10	COBOL			
11	Java			
12	アセンブラ			
13	表計算			

4. 平成 22 年度秋の午後試験の出題内容と出題傾向

(1) 出題内容

問番号	分野	テーマ	難易度
1	ハードウェア	温度モニタ	標準
2	データベース	コールセンターの対応記録管理	
3	ネットワーク	CRC(巡回冗長検査)	
4	情報セキュリティ	認証システム	
5	ソフトウェア設計	部品の棚卸金額計算	
6	IT サービスマネジメント	IT サービスマネジメントにおける個人情報保護	
7	システム戦略	子会社の業績評価	
8	データ構造とアルゴリズム	符号付き 2 進整数の乗算	難
9	C	バスの到着待ち時間	標準
10	COBOL	有料自動車道路の IC 別売上と利用台数の集計	
11	Java	電子会議システム	
12	アセンブラ	ビット列の逆転	
13	表計算	シャンプーの価格弾力分析	

(2) 出題傾向

22 年秋の午後試験では，前半の選択問題のブロックと，後半のプログラム言語のブロックで，各問題で難易度に差がでないように工夫されていました。

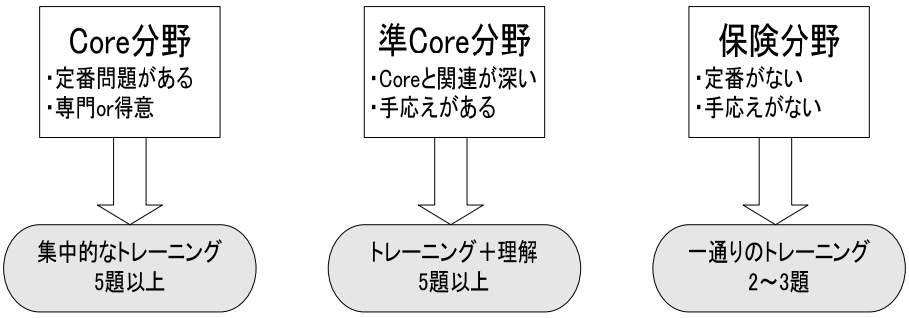
必須問題のアルゴリズムについては，前回(22 年春)よりも難易度が高くなっていました。後半のプログラム言語は，ややボリュームが多いものの，作業量の均一化が図られており，言語間の難易度の差は縮まっています。選択ブロックでの得点が鍵になったでしょう。

午後全体でみると，ボリュームはやや多めでしたが，「標準的な難易度」であったといえます。

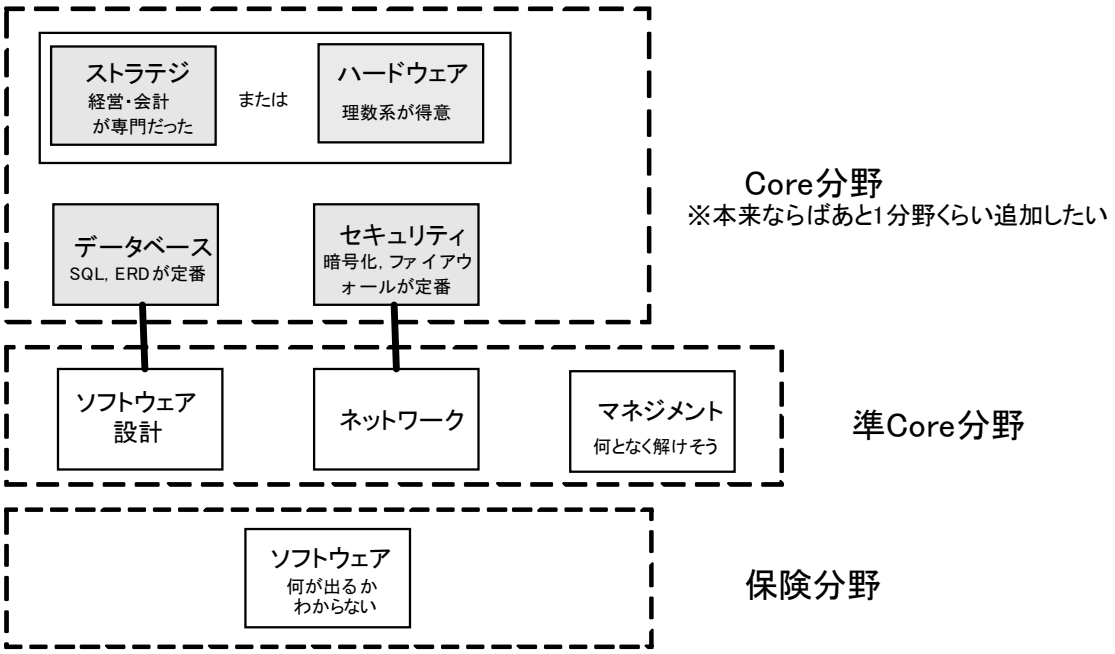
5. 午後対策　分野別学習法

(1) 午後選択問題対策

・トレーニング中心。間違えてしまった設問は解説で理解する。
　場合によってはテキストに戻って復習し直す。
・原則として苦手な分野は選ばない。 (午前と同様)分野にメリハリを付ける。
・Core 分野，準 Core 分野，保険分野に分ける
　　→ Core 分野は強力な得点源にする。
　　→ 準 Core 分野は Core と同じくらいに育てる
　　→ 保険分野はそのもの保険，余り力を入れない

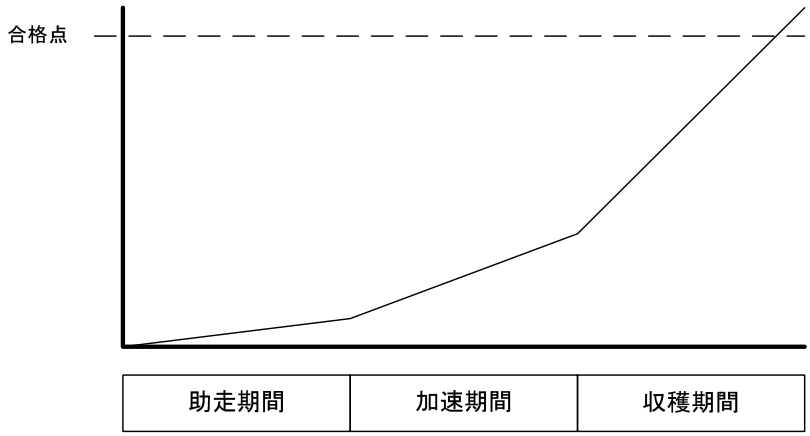


【選択分野の例】



(2) 午後　アルゴリズム系の学習期間

・問題を解くための「スキル」が必要な分野(アルゴリズム，プログラム言語)
・長期的な学習でスキルを高める
　→ スキルを合格レベル(50～80%)にあげるには，3 ヶ月～(場合によっては)6 ヶ月近く必要



★助走期間(0.5～2 ヶ月)

・とにかく，プログラムの全体像や流れをつかむようにする。
　この期間に集中して文法を覚えようとしても，難しく感じる部分が多くなってしまう。
・わからない部分を気にするよりも，わかるところをつまみ食いする。
・アルゴリズムのテキストを眺めてみる。アルゴリズムの読解を「試みる」。

★加速期間(1～2 ヶ月)

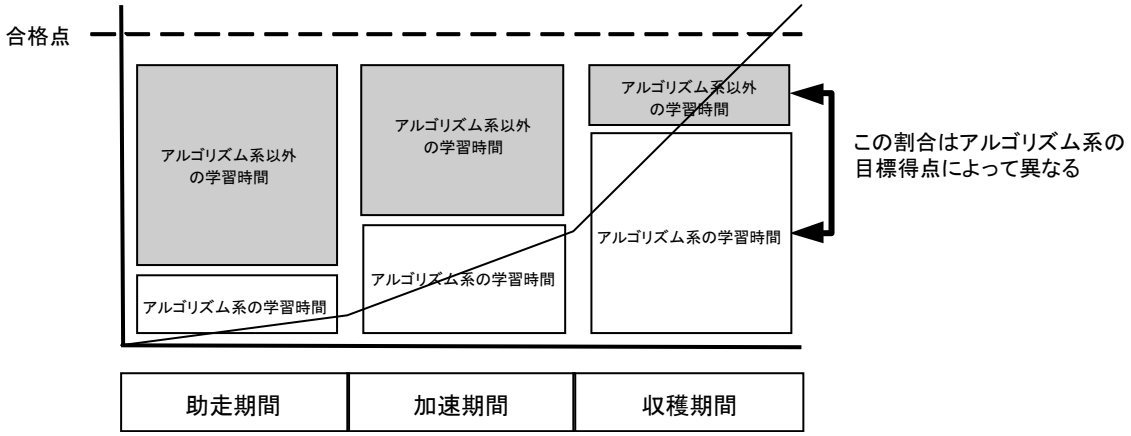
・テキストをしっかり読んで，疑問点をつぶしてゆく。
　→ 文法や記述法などを確実に覚えてゆく。
・わからなければ，誰か(講師，先輩，友達，etc)に聞く。
　→ それでもわからなければ先に進む(先に進むことで，後から解決することもある)。

★収穫期間(1～2 ヶ月)

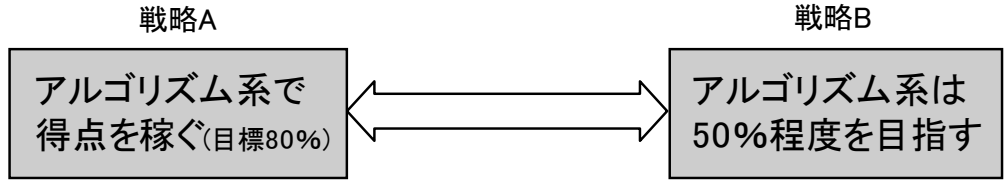
・過去問題 or 問題集などに収録された問題をとにかく解く。
・わからないところは，解説などを読んで理解する。
　→ とにかく「理解する」まではプログラムに取り組む。
・初めのうちは実際の解答時間(1 題あたり 20～25 分)にはこだわらない。
・終盤では解答時間を意識したトレーニングを行う。
・この期間はできるだけプログラミングに集中する。
　→ 他分野の学習は気晴らし or 実力維持トレーニング程度。

6. 午後対策の学習スケジュール

午後対策の学習スケジュールは，アルゴリズム系の対策が鍵！



(計画しておきたいこと)



・得点面での不合格リスクは低い
・時間的な不合格リスクが高い

・得点面での不合格リスクは高い
・時間的な不合格リスクは低い

・午前対策は「今すぐ」始める → アルゴリズム系の助走期間が最大のチャンス
・午前対策を一通り終えてから午後選択問題トレーニングに移った方が，成功率は高い
・アルゴリズム系を「50%でよい」とするならば，アルゴリズム系の学習時間を減らし，その分，選択分野の学習時間に充てる。

7. 午後の解法テクニックを伝授

～ アルゴリズム問題の攻略法 ～

「アルゴリズム(擬似言語)」は, 基本情報技術者試験の午後問 8 に出題される必須問題です。難易度, 配点ともに高く，午後試験をクリアする上での「難関」といってもよいでしょう。アルゴリズム問題を苦手とする多くの方が「どのように解けばよいのかわからない」と感じておられるようです。本セミナーでは, TAC 講師による擬似言語を解くときのアプローチやポイントの一部を公開したものです。もちろん，これらがすべてではありませんが，擬似言語が苦手という方は，このアプローチ方法をぜひ実践してみるとよいでしょう。

(1)　擬似言語のアプローチ

擬似言語を解くとき, 講師などの経験者は, 一体どのようなアプローチをとっているのでしょうか。実は，経験者がとるアプローチは 1 つです。それは「分割して考える」ことなのです。

プログラム全体を一度に考えようとすると，プログラム読解が難しく感じるでしょう。プログラムをいくつかのブロックに分割してみると，それぞれのブロック単位では，比較的構造が単純であることに気がつきます。プログラムを分割することにより，読解を容易にできるのです。これを積み上げていくことで，プログラム全体を把握でき，問題を解くのに要する時間を短縮できる場合が多いのです。

(2)　ブロック分割

プログラムは分割して考える，とはいえ，でたらめに分割しては効果がありません。意味のあるブロックに分割することが重要なのです。慣れないうちは，プログラムの制御構造に従って分割すればよいでしょう。(午後対策講義・演習では，より具体的なテクニックをお教えします)

(3)　変数の役割を明らかにする

プログラムで用いる変数の役割を明らかにしましょう。ブロック内だけで明らかにならない場合は，プログラム全体に視点を広げましょう。プログラムからその変数を用いている命令を抜き出せば，だいたいの想像がつくはずです。

特に変数の多いプログラムでは，多少時間がかかっても，変数の役割をまとめておくことが必要です。また，添字などは図示するとよいでしょう。

(4)　トレースは最小限に抑える

アルゴリズムを理解するためには「トレース」がかかせません。丁寧にトレースすれば，プログラムの動作が理解できるのです。ただし，本試験に限っては，トレースは
「必要最小限」に抑えなければなりません。なぜなら，トレースは時間のかかる作業だからです。ここでは，思い切って,

“練習ではベタトレース推奨，ただし，本試験ではトレースは必要最小限”

と言いたいと思います。こうすることで，本試験でのトレースによる時間のロスを防ぎ，効率的に解くことができるようになります。(具体的なテクニックについては，講義・演習でお教えいたします)

(5) 基本部品を蓄積する

プログラムの定型的な処理パターン（基本部品）を覚えておきましょう。基本部品を蓄積しておくことで，同じパターンを使ったアルゴリズム問題が出題された場合に時間を大幅に節約することができます。基本アルゴリズムやデータ構造についてはしっかり理解しておきましょう。

8. TAC の午後対策　アルゴリズム学習の流れ

● 講義

アルゴリズムの知識習得には時間がかかります。なるべく，早い時期から体系的に学習していくことが大切です。

・アルゴリズム・データ構造の知識を習得。

・基本問題の演習（問題を解くことで知識のモレ・弱点を確認）を行う。

・弱点箇所，未履修のテーマを必ず再確認する → 苦手分野が少なくなる

「インプット → アウトプット → 再インプット」

の学習サイクルを繰り返す。

● 演習

午後対策の専用教材を使って，答案練習を行います。

・広い範囲の過去・模擬問題の演習

・擬似言語・プログラム言語など午後の応用演習　｝ＴＡＣ式 解法テクニック

・選択問題対策

● 公開模試で腕試し

試験直前期の実力診断・弱点分野の把握に最適です。

※　基本情報技術者の午後試験では「プログラミング言語」が出題されます。
　　ご自身の学習目的にあったプログラミング言語を選択しましょう。

【プログラミング言語の種類・特徴】

名　称	特　徴	学習時間	過去の難易度
C	システム記述やアプリケーションの作成に用いられる開発者向きの汎用言語です。	やや多め	普通～やや難
COBOL	事務処理向きで，英文形式で命令を記述するため，理解しやすい言語ですが，現在ではやや歴史を感じさせる面もあります。	普通	普通
Java	現在広く利用されているオブジェクト指向型として人気の高い言語です。多機能ですが，学習内容は多くなります。	多め	普通～やや難
CASLⅡ	試験用のアセンブラ言語です。 命令数が少なく，理解し易いですが，実務向きではありません。初学者（開発系）向き。	やや少なめ	やや易～普通
表計算	試験唯一のユーザ向き言語です。身近で理解し易く，実務にも活かせます。初学者(ユーザ系)向き。	少なめ	やや易～普通

Memo

問4 次のプログラムの説明，擬似言語の記述形式の説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

逆ポーランド表記法で表された数式を，スタックを用いて計算する副プログラム Calc である。

(1) 逆ポーランド表記法で表された数式は，文字型の1次元配列 Ex の各要素 Ex[0]，…，Ex[Lp]（Lp ≧ 3）に1文字ずつ格納されている。

(2) 数式は，正負の整数及び一つ以上の四則演算記号からなる。ただし，正の整数の場合はプラス記号を付けない。

(3) 先頭の整数を除き，整数の前には一つの空白文字を置く。

(4) 計算は実数で行う。計算の途中で次の状態になった場合は，副プログラム Abort() を呼び出してプログラムを終了する。

① ゼロ除算が行われた。

② スタックの範囲外を参照した。

(5) Calc は，実数をスタックに積む副プログラム Push と，スタックから実数を取り出す副プログラム Pop を使用する。各副プログラムの引数の仕様を表1～3に示す。また，数字1文字を実数に変換する関数 ToReal も使用する。

(6) 実数型の1次元配列 Stack，Stack の最大要素番号を示す定数 MAX，スタックの操作位置を示す変数 Sp が，大域変数として定義されている。Sp の初期値は0である。

(7) 逆ポーランド表記法で表された数式は，正しいものとする。

例： 中置表記法による数式　　逆ポーランド表記法による入力

(1 + 2) * (3 − 4)

	0	1	2	3	4	5	6	7	8	9	
Ex	1	△	2	+	△	3	△	4	−	*	Lp = 9

12 ／(− 345)

	0	1	2	3	4	5	6	7	
Ex	1	2	△	−	3	4	5	／	Lp = 7

注　△は空白文字を示す

表1　Calc の引数の仕様

変数名	入力／出力	意　味
Ex[]	入力	逆ポーランド表記法の数式が格納された文字型の1次元配列
Lp	入力	文字型の1次元配列 Ex の最後の要素番号（Lp ≧ 3）
Ret	出力	計算結果

表2　Push の引数の仕様

変数名	入力／出力	意　味
T	入力	スタックに積む実数

表3　Pop の引数の仕様

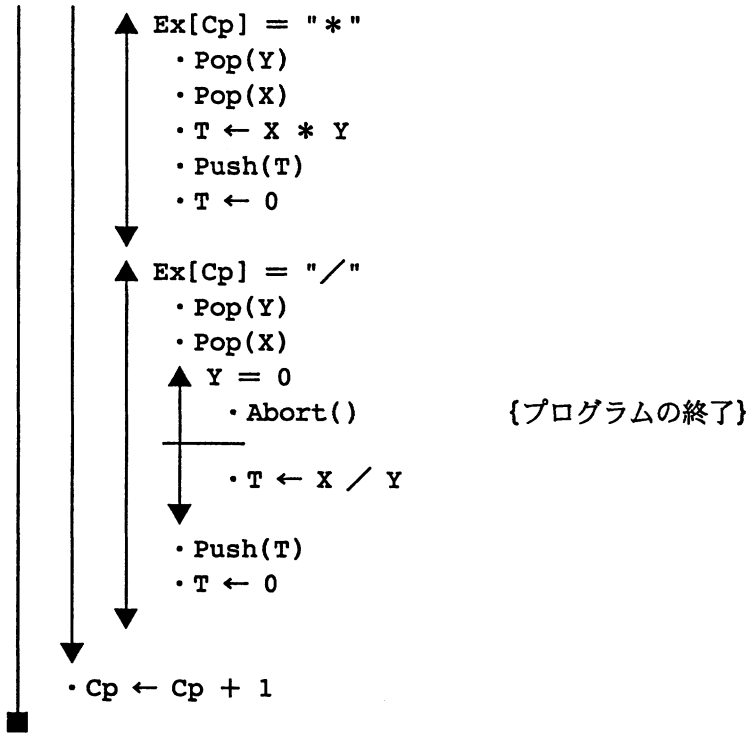
変数名	入力／出力	意　味
T	出力	スタックから取り出す実数

〔擬似言語の記述形式の説明〕

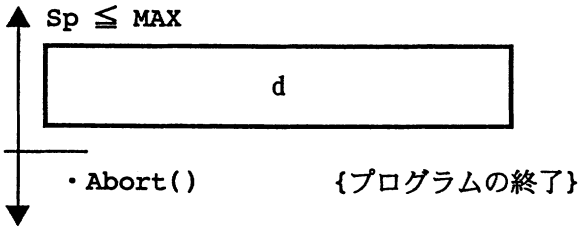
記述形式	説　明
○	手続，変数などの名前，型などを宣言する。
• 変数 ← 式	変数に式の値を代入する。
{文}	文に注釈を記述する。
▲ 条件式 • 処理1 — • 処理2 ▼	選択処理を示す。 条件式が真のときは処理1を実行し， 偽のときは処理2を実行する。
■ 条件式 • 処理 ■	前判定繰返し処理を示す。 条件式が真の間，処理を実行する。

〔プログラム〕

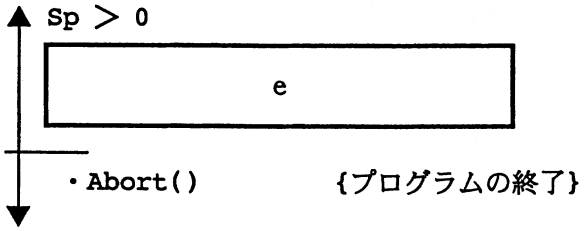
```
○副プログラム名： Calc(Ex[], Lp, Ret)
○文字型： Ex[]
○整数型： Lp, Cp
○実数型： Ret, X, Y, T
○論理型： NumF, NegF
・Cp ← 0
・T ← 0
・NumF ← false
・NegF ← false
■ Cp ≦ Lp
  ▲ Ex[Cp] ＝ 数字
    ・T ← T * 10 + ToReal(Ex[Cp])
      {ToReal() は数字１文字を実数型に変換する関数}
    ・NumF ← true
  ――――
    ▲ [ a ]
      NegF ＝ true
        ・T ← −T
        ・NegF ← false
      ・Push(T)
      ・[ b ]
      ・NumF ← false
    ▲ Ex[Cp] ＝ "+"
      ・Pop(Y)
      ・Pop(X)
      ・T ← X + Y
      ・Push(T)
      ・T ← 0
    ▲ Ex[Cp] ＝ "−"
      ▲ Cp ≠ Lp  AND  Ex[Cp+1] ＝ 数字
        ・[ c ]
      ――――
        ・Pop(Y)
        ・Pop(X)
        ・T ← X − Y
        ・Push(T)
        ・T ← 0
```



○副プログラム名： Push(T)
○実数型： T
○外部参照： Stack[], Sp



○副プログラム名： Pop(T)
○実数型： T
○外部参照： Stack[], Sp



設問　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　Ex[Cp] = " "　　イ　NumF = false　　ウ　NumF = true

エ　T = 0　　オ　T ≠ 0

b, c に関する解答群

ア　NegF ← false　　イ　NegF ← true　　ウ　NumF ← false

エ　NumF ← true　　オ　T ← 0

d, e に関する解答群

ア
・Sp ← Sp + 1
・Stack[Sp] ← T

イ
・Sp ← Sp + 1
・T ← Stack[Sp]

ウ
・Sp ← Sp − 1
・Stack[Sp] ← T

エ
・Sp ← Sp − 1
・T ← Stack[Sp]

オ
・Stack[Sp] ← T
・Sp ← Sp + 1

カ
・Stack[Sp] ← T
・Sp ← Sp − 1

キ
・T ← Stack[Sp]
・Sp ← Sp + 1

ク
・T ← Stack[Sp]
・Sp ← Sp − 1