筆答専門試験科目（午前）　　３０ 大修

情報工学系

時間　９：３０～１２：３０

注　意　事　項

1. 各解答用紙に受験番号を記入せよ.
2. 次の５題のうち，１番～３番の３題は必ず解答し，４番と５番からは１題を選び解答すること.
3. 解答は１題ごとに別々の解答用紙に，問題番号を明記した上で記入せよ．必要であれば，解答用紙の裏面に記入して良いが，解答用紙の表面にその旨を明記すること.
4. １枚の解答用紙に２題以上の解答を記入した場合はそれらの解答を無効とすることがある.
5. １題の解答を２枚以上の解答用紙に記入した場合はその解答を無効とすることがある.
6. 電子式卓上計算機等の使用は認めない.

1.

1) 3つのベクトル $\mathbf{x}_1 = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ について次の問いに答えよ.

a) $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ は線形独立かどうかを根拠とともに述べよ.

b) Gram-Schmidt の正規直交化法を $\mathbf{x}_1$ から順に適用して，これら3つのベクトルが張る空間の正規直交基底を求めよ.

2) 行列 $A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$ について次の問いに答えよ.

a) $A^\top A$ を求めよ．ただし，$\top$ は転置を表す.

b) $A^\top A$ のランクを求めよ.

c) $A^\top A$ の固有値を求めよ.

d) $A^\top A$ の正規化された固有ベクトルを求めよ.

3) 3変数関数 $f(x, y, z) = x^2 + 2y^2 + 3z^2 + 2xy + 2xz + 3$ について次の問いに答えよ.

a) f の偏導関数 f'_x, f'_y, f'_z を求めよ.

b) f の全ての臨界点の座標 (x, y, z) を求めよ．ただし，臨界点では $f'_x = f'_y = f'_z = 0$ となる.

2.

1) 図 2.1 に示す有限状態オートマトンと等価な有限状態オートマトンのうち，状態数が最小のものを図示せよ．ただし，最小性の証明は必要ない．ここで，a, b は記号，“○” は非受理状態でそのうちの “①” が開始状態，“◎” は受理 (最終) 状態を示す．

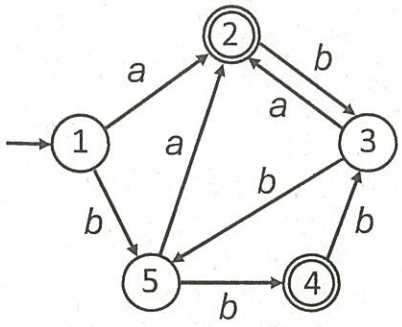

図 2.1: 有限状態オートマトン

2) 言語 $\{ ww \mid w \in \{ a, b \}^* \}$ は文脈自由言語か．文脈自由言語である場合はそれを生成する文法を示せ．文脈自由言語でない場合はそのことを証明せよ．ここで，a，b は記号である．

3) 次の言語クラスが，補集合と積集合 (共通集合) の各々について，閉じているか否かをその証明とともに示せ．

   a) 正規言語

   b) 文脈自由言語

3.

1) **（短絡評価，優先順位，結合性）**次の問いに答えよ．

a) C 言語の演算子である論理積「&&」と論理和「||」の評価方法は**短絡評価**なので，オペランドの評価順序に注意が必要である．すなわち，２つの式 e_1，e_2 に対して，

i) e_1 && e_2 の評価では，まず e_1 を評価し，e_1 の評価結果が真（0 でない整数）の場合のみ，e_2 を評価する．e_1 の評価結果が偽（0）の場合は，e_2 は評価されない．

ii) e_1 || e_2 の評価では，まず e_1 を評価し，e_1 の評価結果が偽（0）の場合のみ，e_2 を評価する．e_1 の評価結果が真（0 でない整数）の場合は，e_2 は評価されない．

短絡評価に注意して，図 3.1 のプログラムの出力結果を答えよ．なお printf の返り値は（ヌル文字を除いた）出力文字数である．

```
#include <stdio.h>
int main () {
    if (0 && printf ("A")) { printf ("B"); }
    if (1 && printf ("C")) { printf ("D"); }
    if (0 || printf ("E")) { printf ("F"); }
    if (1 || printf ("G")) { printf ("H"); }
}
```

図 3.1: 短絡評価に注意が必要なプログラム

b) 異なる演算子からなる式では，**優先順位**に従って，どの演算子を先に適用するかが決まる．例えば，C 言語では，引き算「-」よりも割り算「/」の優先順位が高いので，`1 - 2 / 3` は `1 - (2 / 3)` と同じ意味になる．

C 言語では，直接メンバー選択「.」と間接メンバー選択「->」は同じ優先順位である．また，アドレス演算「&」と間接演算「*」は同じ優先順位である．直接メンバー選択「.」と間接メンバー選択「->」はアドレス演算「&」と間接演算「*」よりも優先順位が高い．

次の式にカッコをつけて優先順位を明確にせよ．

i) `*x->y`

ii) `&x.y`

なお，間接メンバー選択を使った式 e->id は $(*e).id$ と同等である（ただし，e は式，id は識別子）．

c) 同じ優先順位を持つ演算子からなる式では，**結合性**に従って，どの演算子を先に適用するかが決まる．例えば，C 言語では，引き算「-」は左側の演算子を優先（左結合）するので，`1 - 2 - 3` は `(1 - 2) - 3` と同じ意味になる．また代入「=」は右結合なので，`x = y = z` は `x = (y = z)` と同じ意味になる．

C 言語では，直接メンバー選択「.」と間接メンバー選択「->」は左結合，また，アドレス演算「&」と間接演算「*」は右結合である．

次の式にカッコをつけて結合性を明確にせよ．

i) `x->y->z`

ii) `*&x`

(次ページにつづく)

(前ページのつづき)

2) **(線形リスト・挿入)** 次の問いに答えよ.

a) **(素朴なプログラム)** 図 3.2 は C 言語による線形リストの素朴な挿入プログラムである. 図 3.2 中の関数 insert1 は, 昇順にデータがソートされるように, 線形リスト head 中に新たなデータ data を挿入する. この動作となるように, 図 3.2 中の空欄 [A] ～ [D] に最も適切な式を, 図 3.5 の選択肢群から選んで, その番号を解答せよ. 同じ選択肢を何度選んでもよい. 例えば, 図 3.2 の main 関数の実行終了直前の線形リストは図 3.3 の通りになる. なお, NULL は「どこも指していないことを示す, 特別なポインタ値 (ヌルポインタ)」であり, 値は 0 である.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct CELL {
   int data; struct CELL *next;
} CELL;
CELL * CELL_alloc (int data) {
   CELL *p = malloc (sizeof (CELL));
   p->data = data; p->next = NULL; return p;
}
CELL * insert1 (CELL *head, int data) {
   CELL *new = CELL_alloc (data);
   CELL *p = head;
   if ( [ A ] || [ B ] ) { // 先頭に挿入する条件
      new->next = p;        // 先頭に挿入
      return new;
   } else {
      while ( [ C ] && [ D ] ) { // 挿入箇所を探す
         p = p->next;
      }
      new->next = p->next; p->next = new;
      return head;
   }
}
int main () {
   CELL *head = NULL;
   head = insert1 (head, 10);
   head = insert1 (head, 30);
   head = insert1 (head, 20);
}
```

図 3.2: 素朴な挿入プログラム

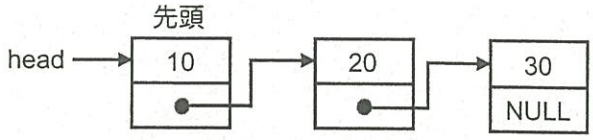

図 3.3: main 関数終了直前の線形リスト

```
void insert2 (CELL **head_p, int data) {
    CELL *new = CELL_alloc (data);
    CELL **p = head_p;
    while ( [ E ] && [ F ] ) {
        p = [ G ] ; // 次の next メンバーを指す
    }
    new->next = *p;
    *p = new;
}
int main () {
    CELL *head = NULL;
    insert2 (&head, 10);
    insert2 (&head, 30);
    insert2 (&head, 20);
}
```

図 3.4: ポインタのポインタを用いた挿入プログラム

b) **(ポインタのポインタを使うプログラム)** 図 3.2 のプログラムでは, 空リストや先頭に挿入する場合, 場合分けが必要になり, プログラムの見通しが悪い. この場合分けを不要にするには, 先頭にダミーセルを置く方法や, ポインタのポインタを使う方法がある. ここでは図 3.4 に示す通りポインタのポインタを使う方法を使う. ただし, 構造体 CELL と関数 CELL_alloc は図 3.2 と同じとする. 図 3.4 中の関数 insert2 が, 図 3.2 中の関数 insert1 と同じ挿入結果を得られるように, 図 3.4 中の空欄 [E] ～ [G] に最も適切な式を, 図 3.5 の選択肢群から選んで, その番号を解答せよ. 同じ選択肢を何度選んでもよい.

(1) p == NULL (2) p != NULL (3) p->next == NULL (4) p->next != NULL (5) (*p) == NULL
(6) (*p) != NULL (7) data < p->data (8) p->data < data (9) data < p->next->data
(10) p->next->data < data (11) data < (*p)->data (12) (*p)->data < data (13) p->next
(14) (*p)->next (15) *p->next (16) &*p->next (17) &(*p)->next (18) (&*p)->next

図 3.5: 選択肢群

(次ページにつづく)

(前ページのつづき).

3) **(線形リスト・逆順並び替え)** 次の問いに答えよ．ただし，構造体 CELL は図 3.2 と同じとする．

a) **(素朴な再帰)** 図 3.6 の reverse1 は，再帰を用いた，線形リストを逆順に並び替えるプログラムである．図 3.3 の線形リスト head を引数として reverse1 を呼び出した時，reverse1 からリターンする直前の線形リストの接続状況を図 3.3 のような形式で図示せよ．また，図にはその時点で変数 head と p が線形リストのどの要素を指しているかも明示すること．reverse1 は再帰呼び出しのため複数回リターンすることに注意せよ（リターンする回数分の図が必要．実行時系列順に解答すること）．

```
CELL * reverse1 (CELL *head) {
    CELL *p = NULL;
    if (head == NULL || head->next == NULL) {
        return head;
    }
    p = reverse1 (head->next);
    head->next->next = head;
    head->next = NULL;
    return p;
}
```

図 3.6: 素朴な再帰で逆順にするプログラム

```
CELL * reverse2 (CELL *head1, CELL *head2) {
    if (head1 == NULL) {
        return head2;
    }
    CELL *new_head1 = [ H ] ;
    [ I ]
    CELL *new_head2 = [ J ] ;
    return reverse2 (new_head1, new_head2);
}
```

図 3.7: 途中状態を引数で渡す，再帰逆順プログラム

```
CELL * reverse3 (CELL *head) {
    CELL *p1 = NULL, *p2 = head, *p3;
    while (p2 != NULL) {
        [ K ]
        p2->next = p1; // 逆向きにリンク
        [ L ]
        [ M ]
    }
    return p1;
}
```

図 3.8: ループで逆順にするプログラム

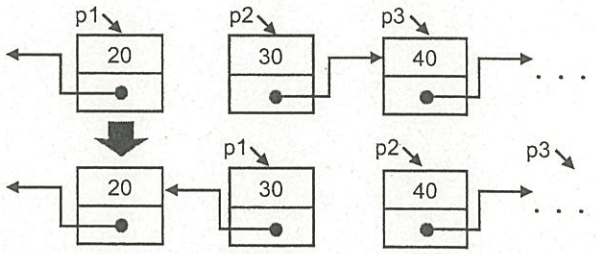

図 3.9: 1 回のループで，ポインタを 1 つ張替え，変数 p1, p2, p3 を 1 つ右に移動させる

b) **(途中状態を引数で渡す再帰)** 図 3.7 の reverse2 も与えられた線形リストを逆順に並び替えるプログラムである．reverse2 (head, NULL); という形式で呼び出して用いる．再帰呼び出しの際に，reverse2 の第 1 引数は「元のリストをある位置で分割した後半のリスト」，第 2 引数は「元のリストを同じ位置で分割した前半を逆順に並び替えたリスト」となる．新たに reverse2 を再帰呼び出しするたびに，第 1 引数から先頭要素を削除し，その先頭要素を第 2 引数の先頭に加える． [H] と [J] に式を， [I] に代入文を入れて，プログラムを完成させよ．

c) **(ループ)** 図 3.8 の reverse3 も与えられた線形リストを逆順に並び替えるプログラムである．ループを使って線形リストを先頭から走査し，ポインタを逆順にしていく．図 3.9 に示すように，変数 p1, p2, p3 は連続する 3 つの要素を指し，1 回のループで 1 つのポインタを逆向きに張替える．次に p1, p2, p3 が指す要素を右に 1 つずらす． [K] , [L] , [M] に，代入文をそれぞれ 1 つずつ入れて，プログラムを完成させよ．

4. 以下，変数（例えば，X, X_i）は全て $\{0, 1\}$ の 2 値を取るものとする．また，$(X_{n-1}, X_{n-2}, ..., X_1, X_0)$ は，X_{n-1} を MSB (Most Significant Bit)，X_0 を LSB (Least Significant Bit) とする n 桁の二進数を表し，負数には 2 の補数表現を用いる．なお，算出途中の関数の出力値に対して新たな変数を割当ててもよい（例えば，$VfXY = f(X, Y)$ のように，変数 $VfXY$ を関数 $f(X, Y)$ の出力の値に割当ててよい）．問に対する解が複数ある場合にはその内の一つを示せばよい．

1) 3 引数 X, Y, Z の多数決を取る関数，すなわち X, Y, Z 中で値が 1 である変数の個数が 2 個以上の場合にのみ 1，それ以外の場合には 0 を返す関数 $\mathrm{maj}(X, Y, Z)$ を積和標準形で表せ．

2) $\mathrm{nor}(X, Y)$ を 2 引数 X と Y の両方が 0 の時のみ 1 を返す関数とし，NX, NY, NZ をそれぞれ X, Y, Z の否定の値を持つ変数とする．このとき，NX, NY, NZ を引数として，X, Y, Z の値が 1 である変数が 2 個以上の場合にのみ 0，それ以外の場合には 1 を返す関数 $\mathrm{nmaj}(NX, NY, NZ)$ を 2 引数 nor 関数のみを用いて表せ．その際，用いる関数の個数を最少となるようにせよ．

3) X, Y, Z のうちの奇数個の変数の値が 1 の時に 1，それ以外の場合には 0 を返す 3 引数関数 $\mathrm{xor}(X, Y, Z)$ を 2 引数 nor 関数のみを用いて表せ．その際，用いる関数の個数を最少となるようにせよ．

4) 以下，二進数 (X_2, X_1, X_0) の 1 の補数を (CX_2, CX_1, CX_0)，同様に二進数 (Y_2, Y_1, Y_0) の 1 の補数を (CY_2, CY_1, CY_0) とする．二進数 (X_2, X_1, X_0) と二進数 (Y_2, Y_1, Y_0) を加算した結果である二進数を (A_3, A_2, A_1, A_0) とする．この時，A_3, A_2, A_1, A_0 のそれぞれを，$CX_2, CX_1, CX_0, CY_2, CY_1, CY_0$ と 3 引数の xor 関数，3 引数の nmaj 関数，2 引数の nor 関数のみを用いて表せ．その際，用いる関数の個数を最少となるようにせよ．

5) 二進数 (X_2, X_1, X_0) から二進数 (Y_2, Y_1, Y_0) を減算した結果である二進数を (S_3, S_2, S_1, S_0) とする．この時，S_3, S_2, S_1, S_0 のそれぞれを，$X_2, X_1, X_0, Y_2, Y_1, Y_0, CX_2, CX_1, CX_0, CY_2, CY_1, CY_0$，3 引数の xor 関数，3 引数の nmaj 関数，2 引数の nor 関数のみを用いて表せ．その際，用いる関数の個数を最少となるようにせよ．

6) 二進数 $(X_{n-1}, X_{n-2}, ..., X_1, X_0)$ と二進数 $(Y_{n-1}, Y_{n-2}, ..., Y_1, Y_0)$ を乗算した結果である二進数を $(M_{2n-1}, M_{2n-2}, ..., M_1, M_0)$ とする．算出のため，それぞれ $2n$ 個の D フリップフロップ $R_{2n-1}, ..., R_0$ と，$P_{2n-1}, ..., P_0$ を用意する．それぞれ k 番目の状態を $Q(R_{2n-1})^{(k)}, ..., Q(R_0)^{(k)}$ および $Q(P_{2n-1})^{(k)}, ..., Q(P_0)^{(k)}$ で表す．初期状態（$k = 0$）で，$i = 2n-1$ から $i = n$ までの R_i に $X_{n-1}, X_{n-2}, ..., X_1, X_0$ を，$i = n-1$ から $i = 0$ までの P_i に $Y_{n-1}, Y_{n-2}, ..., Y_1, Y_0$ を保持させる．それら以外のフリップフロップには 0 を保持させる．すなわち，$0 \le j \le n-1$ なる j に対して，$Q(R_{n+j})^{(0)} = X_j$，$Q(R_j)^{(0)} = 0$，$Q(P_{n+j})^{(0)} = 0$，$Q(P_j)^{(0)} = Y_j$ となるように初期設定する．さらに，フリップフロップ P_{-1} を用意し，初期状態で 0，すなわち，$Q(P_{-1})^{(0)} = 0$ とする．

   a) $0 \le k \le n-1$ において $0 \le i \le 2n-1$ に対し，
      - $Q(P_0)^{(k)} = 0$，$Q(P_{-1})^{(k)} = 0$ の場合に，$Q(P_i)^{(k)} \to Q(P_{i-1})^{(k+1)}$
      - $Q(P_0)^{(k)} = 1$，$Q(P_{-1})^{(k)} = 0$ の場合に，$(Q(P_i)^{(k)} + Q(R_i)^{(k)}) \to Q(P_{i-1})^{(k+1)}$
      - $Q(P_0)^{(k)} = 0$，$Q(P_{-1})^{(k)} = 1$ の場合に，$(Q(P_i)^{(k)} - Q(R_i)^{(k)}) \to Q(P_{i-1})^{(k+1)}$
      - $Q(P_0)^{(k)} = 1$，$Q(P_{-1})^{(k)} = 1$ の場合に，[A]

      と処理を進める．なお，加算と減算では桁上げを行う．その結果 $k = n-1$ の時の $Q(P_{2n-1})^{(k)}, ..., Q(P_0)^{(k)}$ が $M_{2n-1}, ..., M_0$ になるために，[A] ですべき動作を述べよ．

   b) 上記のように加算と減算を行うことで積が求められる理由を，数式を用いて具体的に説明せよ．

   c) $Q(P_{-1})^{(k)}$ の値を参照せず，
      - $Q(P_0)^{(k)} = 0$ の場合に，$Q(P_i)^{(k)} \to Q(P_{i-1})^{(k+1)}$
      - $Q(P_0)^{(k)} = 1$ の場合に，$(Q(P_i)^{(k)} + Q(R_i)^{(k)}) \to Q(P_{i-1})^{(k+1)}$

      とした場合と比べ，a) で述べた動作の長所と短所を述べよ．

5.

1) 周期 2π の区分的に滑らかな周期関数 $f(x)$ のフーリエ級数は，不連続点を除けば，

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx)$$

で与えられ，その係数 a_0, a_k, b_k $(k = 1, 2, \cdots)$ は，

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,dx,\ a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos kx\,dx,\ b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin kx\,dx$$

となる．以下の問いに答えよ．

a) 次の周期 2π の周期関数 $f_1(x)$ のフーリエ級数の係数 a_0, a_k, b_k $(k = 1, 2, \cdots)$ を求めよ．

$$f_1(x) \quad = \quad x^2 \quad (-\pi < x \leq \pi)$$

b) 上記 a) の結果を用いて，$\displaystyle\sum_{k=1}^{\infty}\frac{1}{k^2}$ と $\displaystyle\sum_{k=1}^{\infty}\frac{1}{k^4}$ の値を計算せよ．必要ならばパーシヴァル (Parseval) の等式

$$\frac{1}{\pi}\int_{-\pi}^{\pi}\{f(x)\}^2\,dx = \frac{a_0^2}{2} + \sum_{k=1}^{\infty}(a_k^2 + b_k^2)$$

を用いてよい．

c) 次の周期 2π の周期関数 $f_2(x)$ のフーリエ級数の係数 a_0, a_k, b_k $(k = 1, 2, \cdots)$ を求めよ．

$$f_2(x) = \begin{cases} -\pi & (-\pi < x \leq 0) \\ x - \pi & (0 < x \leq \pi) \end{cases}$$

d) 上記 c) で与えられた周期関数 $f_2(x)$ の不連続点近傍における，$f_2(x)$ のフーリエ級数の振舞いを簡潔に説明せよ．

2) $-\infty < t < \infty$ で区分的に連続で絶対可積分な関数 $f(t)$ のフーリエ変換 $\mathscr{F}[f(t)](\omega)$ を

$$\mathscr{F}[f(t)](\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,dt$$

と定義する．ただし，i は虚数単位であり，ω は実数である．以下の問いに答えよ．

a) $f(t)$ と $g(t)$ を $-\infty < t < \infty$ で区分的に連続で絶対可積分な関数とする．このとき，以下の 2 つの式が成り立つことを示せ．ただし，a, b, ω_0 は実定数である．

i) $\mathscr{F}[af(t) + bg(t)](\omega) = a\mathscr{F}[f(t)](\omega) + b\mathscr{F}[g(t)](\omega)$

ii) $\mathscr{F}[f(t)e^{i\omega_0 t}](\omega) = \mathscr{F}[f(t)](\omega - \omega_0)$

b) $-\infty < t < \infty$ で区分的に連続で絶対可積分な関数 $f(t)$ のフーリエ変換 $\mathscr{F}[f(t)](\omega)$ を $F(\omega)$ とおくとき，以下の関数のフーリエ変換を $F(\omega)$ を用いて表せ．ただし，ω_0 は実定数である．

i) $f(t)\cos\omega_0 t$ ii) $f(t)\cos^2\omega_0 t$

（次ページにつづく）

（前ページのつづき）

c) $-\infty < t < \infty$ で区分的に連続で絶対可積分な信号 $f(t)$ のフーリエ変換 $\mathscr{F}[f(t)](\omega)$ を $F(\omega)$ とする．$|F(\omega)|$ のグラフが図 5.1 のように与えられたとする．$F(\omega)$ は $|\omega| > \omega_1$ において $|F(\omega)| = 0$ である．ただし，ω_1 は正の定数であり，$|F(0)| = F_0$ である．以下の問いに答えよ．

i) 信号 $g(t) = f(t)\cos\omega_0 t$ のフーリエ変換 $\mathscr{F}[g(t)](\omega)$ について，$|\mathscr{F}[g(t)](\omega)|$ のグラフを図示せよ．ただし，ω_0 は正の定数であり，$\omega_0 > 2\omega_1$ であるとする．

ii) 信号 $h(t) = g(t)\cos\omega_0 t$ のフーリエ変換 $\mathscr{F}[h(t)](\omega)$ について，$|\mathscr{F}[h(t)](\omega)|$ のグラフを図示せよ．また，信号 $g(t)$ から元の信号 $f(t)$ を復元する方法を簡潔に説明せよ．

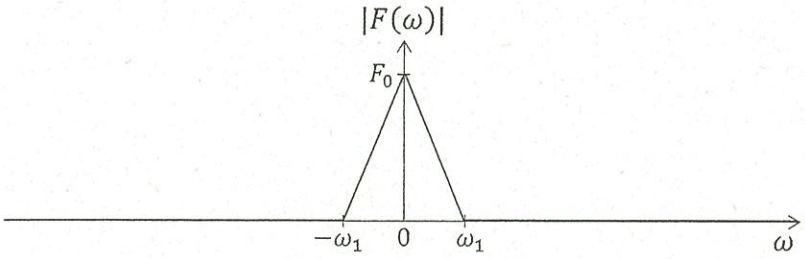

図 5.1 $|F(\omega)|$ のグラフ