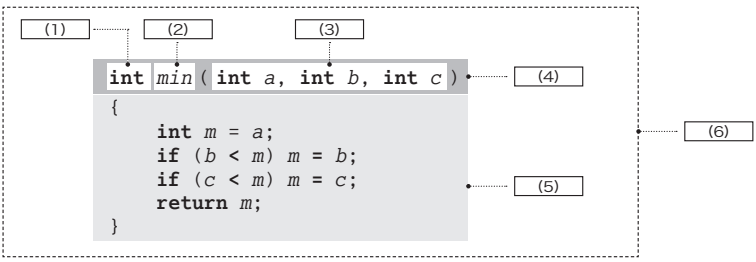
錬成問題

▪ 関数は、プログラムの部品である。各部の名称を記入せよ。



▪ 関数の中で宣言する局所変数の名前に関して、以下の規則がある。

○ 関数と同じ名前を与えることは (7) 。

○ 仮引数と同じ名前を与えることは (8) 。

▶ 共通の選択肢：(a)できる　(b)できない

▪ *min*(1, 3, 2) は、上に示した関数 *min* を呼び出す式である。

ここで使われている演算子 ( ) の名称は、 (9) 演算子である。また、関数に対して補助的な指示を与えるための 1 や 3 や 2 は、 (10) と呼ばれる。関数が呼び出されると、プログラムの流れは、その関数へと移る。

呼び出された関数 *min* の仮引数 *a*, *b*, *c* は、 (10) の値で初期化される。このような、受け渡しのメカニズムのことを (11) と呼ぶ。

関数 *min* は、受け取った三つの整数値の最小値を求めて、呼出し元に返す。呼出し元に返す値の型を示すのが、図中の (1) である。値を返さない関数の (1) は、 (12) と宣言する。なお、関数は二つ以上の値を一度に返すことは (13) 。

なお、関数によって返された値は、関数を呼び出した式を評価することによって得られる。*min*(1, 3, 2) を評価して得られる型は (14) で、値は (15) である。

▶ (13) の選択肢：(a)できる　(b)できない

▪ 変数の寿命すなわち生存期間のことを (16) と呼ぶ。その (16) には、 (17) (16) 、 (18) (16) などがある。宣言時に初期化子が与えられない場合、前者の変数は不定値で初期化され、後者の変数はゼロで初期化される。

▪ 右に示すのは、受け取った二つの **int** 型の値の和を返す関数である。

```
int sum_of(int a, int b)
{
    (19) a + b;
}
```

▪ 右に示すのは、『Hello!』と表示する関数である。なお、関数 *hello* のように、**inline** 付きで定義された関数は、 (20) 関数と呼ばれる。

```
inline (21) hello()
{
    cout << "Hello!\n";
}
```

▪ **return** 文は、コンマで区切ることによって、複数の式の値を返却することが (22) 。もし **main** 関数に **return** 文がまったくない場合に返却される値は、**int** 型の (23) である。

▶ (22) の選択肢：(a)できる　(b)できない

▪ プログラムの流れを一気に移動させる **break** 文・**continue** 文・**return** 文・**goto** 文の総称は (24) である。

▪ 同一の有効範囲内で同一の名前をもった関数を定義することを (25) と呼ぶ。(25) を行うためには、各関数の (26) が異なっている必要がある。

(26) に含まれるものには○を、含まれないものには×を示せ。

- ○関数名 … (27)
- ○仮引数の名前 … (28)
- ○返却値型 … (29)
- ○仮引数の型と個数の組合せ … (30)

▪ 以下に示すのは、受け取った二つの **int** 型の値の差を返す関数である。

```
int diff_of(int a, int b)
{
    if ( (31) )
        (32) ;
    (33) ;
}
```

```
int diff_of(int a, int b)
{
    return (31) ? a - b : (34) ;
}
```

▪ 右に示すのは、受け取った二つの **int** 型の値の平均を **double** 型の実数値で返す関数である。

```
double ave_of(int a, int b)
{
    return (35) < (36) >(a + b) / 2;
}
```

▪ 右に示すのは、受け取った二つの **int** 型引数の値が等しければ **true** を、そうでなければ **false** を返す返す関数である。

```
(37) eq(int a, int b)
{
    return (38) ;
}
```

▪ 以下に示すのは、x の n 乗を返却する関数である。

```
double power(double x, int n)
{
    double tmp = (39) ;
    while ( (40) > 0)
        tmp *= x;
    return tmp;
}
```

```
double power(double x, int n)
{
    double tmp = (39) ;
    for (int i = (41) ; i < n; i++)
        tmp *= x;
    return tmp;
}
```

▪ 右に示すプログラム部分において、『y は x を (42) する』と表現され、y は x の (43) となる。

▶ (43) の選択肢：(a)エイリアン
(b)エイリアス　(c)ライブラリアン

```
int  x = 1;
int& y = x;
x = 2;
cout << "x : " << x << '\n';
cout << "y : " << y << '\n';
```

```
x : (44)
y :
```

▪ 各変数の記憶域期間を示せ。
  ▫ 関数の外で定義された変数 … (45)
  ▫ 関数の中で **static** 付きで定義された変数 … (46)
  ▫ 関数の中で **static** 無しで定義された変数 … (47)
  ▶ 共通の選択肢：(a)自動記憶域期間　(b)静的記憶域期間

▪ **main** 関数で **return** 文を実行すると、プログラムは (48) する。処理に成功した場合は (49) を返却し、そうでなければ (49) 以外の値を返却することになっている。
  ▶ (48) の選択肢：(a)終了　(b)一時中断　(c)再起動　(d)暴走

▪ 関数の外で定義された変数には (50) 有効範囲が与えられ、関数の中で定義された変数には (51) 有効範囲が与えられる。

異なる有効範囲をもつ同一名の変数が存在する場合、より (52) のものが見えて、その逆のものが隠される。有効範囲解決演算子 **::** を利用すると、隠されている (53) 有効範囲の変数にアクセスできる。
  ▶ (52) の選択肢：(a)内側　(b)外側

▪ 定義の先頭に (54) が与えられた関数は、インライン関数となる。インライン関数は、(55) 規模な関数の実現に向いている。
  ▶ (55) の選択肢：(a)大　(b)小

▪ 右に示すのは、文字 *c* を *n* 個連続して表示する関数である。この関数を *put_nchar*(5.7, '+') と呼び出すと、文字 '+' が (56) 個表示される。

```
void put_nchar(int n, char c)
{
    while ( (57) > 0)
        cout << c;
}
```

▪ 右に示すのは、前問の関数 *put_nchar* を利用して、文字 '*' を *n* 個連続して表示するインライン関数である。

```
(58) void put_stars(int n)
{
    (59)( (60) , (61) );
}
```

▪ 右に示すのは、**int** 型の仮引数 *a*, *b*, *c* の和を求めて返却する関数であり、第２実引数と第３実引数を省略して呼び出せる。たとえば、*sum*(3, 1, 5) は 9 を返却し、*sum*(5, 2) は 7 を返却し、*sum*(6) は 6 を返却する。なお、実引数を省略して呼び出せるようにするために与える値は、(62) と呼ばれる。

```
int sum(int a, int (63) , int (64) )
{
    return (65) ;
}
```

▪ 右に示すのは、*a* に *b* を加える（*a* の値を *b* だけ増やす）関数である。ここでの第１引数の受渡しは (66) と呼ばれ、第２引数の受渡しは (67) と呼ばれる。

```
void add(int& a, int b)
{
    a (68) b;
}
```

▪ 右に示すプログラムで利用されている :: は、(69) 演算子である。

もし初期化子が与えられていなければ、❶の a の初期値は (70) となり、❷の a の初期値は (71) となる。

▶ (70) と (71) の選択肢：(a) 0　(b) 1　(c) 不定値　(d) int 型の最小値

```
❶ int a = 1;
   int main()                    (72)
   {
❷      int a = 2;
       cout <<   a << '\n';
       cout << ::a << '\n';
   }
```

▪ 以下に示すプログラム部分の実行結果を示せ。

```
int& ref()                  int main()
{                           {                          ref() = (73)
    static int x;               ref() = 5;
    return x;                   cout << "ref() = " << ref() << '\n';
}                           }
```

▪ 右に示すのは、< (74) > ヘッダに置かれている、標準ライブラリ *rand* 関数と *srand* 関数の (75) である。なお、仮引数名 *seed* は、省略 (76)。

▶ (76) の選択肢：(a) できる　(b) できない

```
int rand();
void srand(unsigned seed);
```

▪ 右に示すのは、呼び出された回数を返却する関数である（たとえば、最初に呼び出されたときは 1 を返却し、2 番目に呼び出されたときは 2 を返却する）。

```
int counter()
{
    (77) int c = 0;
    return (78);
}
```

▪ 右に示すのは、int 型の整数値を交換する関数である（引数は参照渡しによって受け取るものとする）。

```
void swap((79) x, (80) y)
{
    (81) t = x;
    x = y;
    y = t;
}
```

▪ 以下に示すプログラムのすべての誤りを指摘して正せ。… (82)

```
#include <iostream>

void main()
{
    int n;
    do {
        cout << "挨拶の回数は：";
        cin >> n;
    } while (n < 1);

    for (int i = 1; i <= n; i++)
        hello();
}

inline hello()
{
    cout << "こんにちは。\n";
}
```