C 言語 017 関数ポインタ

【関数ポインタ】

関数も変数と同じようにメモリに記録されるので「アドレス」を持っており、関数のアドレスを記録するポインタ変数を「関数ポインタ」と称し、関数ポインタを使用して関数が呼び出せます。

＜関数ポインタ変数の宣言＞

```
戻り値のデータ型 (*変数名)(引数のデータ型…);
```

＜アドレスの代入＞

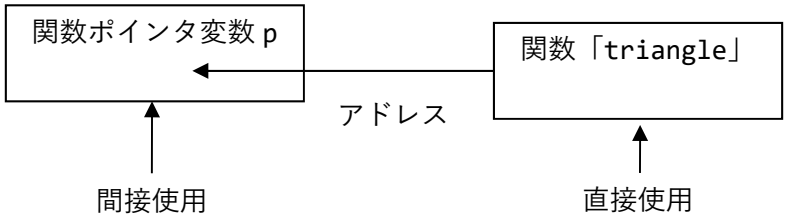
```
関数ポインタ変数 = 関数名;
```

＜関数ポインタを使用して関数を呼び出す＞

```
(*関数ポインタ変数)(実引数…);
```

＜例文 1＞

```
/* 三角形の面積を求める関数 */
double triangle(double base, double high) {
    return base * high / 2;
}

/* メイン関数 */
int main(void) {
    double (*p)(double, double);    /* 関数ポインタの宣言。戻り値、引数を合わせること */
    double area;                    /* 戻り値 */
    p = triangle;                   /* 関数「triangle」のエントリポインタを代入 */
    area = (*p)(10,20);             /* 関数ポインタを使用して関数「triangle」を呼び出す */
    printf("三角形の面積は%lf¥n", area);
    return 0;
}
```



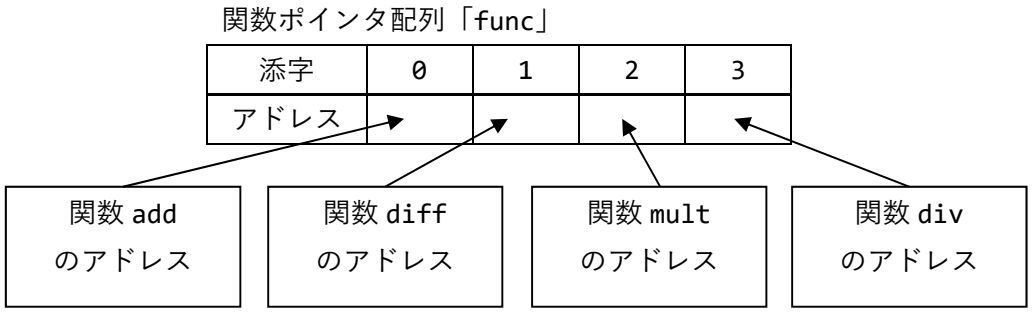
関数ポインタを使用すれば、関数を「直接使用」ではなく「間接使用」することができます。

＜例文 2＞

```
double add(double num_a, double num_b) {     /* 足し算を計算する関数 */
    return num_a + num_b;
}

double diff(double num_a, double num_b) {    /* 引き算を計算する関数 */
    return num_a - num_b;
}

double mult(double num_a, double num_b) {    /* 掛け算を計算する関数 */
    return num_a * num_b;
}

double div(double num_a, double num_b) {     /* 割り算を計算する関数 */
    return num_a / num_b;
}

/* メイン関数 */
int main(void) {

    /* 関数ポインタを配列化し、各関数のアドレスを代入 */
    double (*func[])(double, double) = {add, diff, mult, div};

    double  num_a;     /* 数値 A */
    double  num_b;     /* 数値 B */
    int     kind;      /* 計算の種類 */
    double  answer;    /* 答え */

    /* 数値を入力 */
    printf("数値 A を入力して下さい。¥n");
    scanf("%lf", &num_a);
    printf("数値 B を入力して下さい。¥n");
    scanf("%lf", &num_b);

    /* 計算の種類を選択 */
    printf("0=足し算、1=引き算、2=掛け算、3=割り算¥n");
    scanf("%d", &kind);
```

（続く）

```
    answer = (*func[kind])(num_a, num_b);    /* 関数ポインタの使用 */
    printf("答えは%.2f です。¥n", answer);

    return 0;
}
```



【関数ポインタを引数にする】

＜例文＞

```
double add(double num_a, double num_b) {     /* 足し算を計算する関数 */
    return num_a + num_b;
}

double diff(double num_a, double num_b) {    /* 引き算を計算する関数 */
    return num_a - num_b;
}

/* 第 1 引数を関数ポインタにしている */
double func( double (*p)(double, double), double num_a, double num_b ) {
    double answer = (*p)(num_a, num_b);
    return answer;
}

int main(void) {
    double answer;
    answer = func(add, 10,5);     /* 第 1 引数が add 関数のアドレス */
    printf("%f¥n", answer);
    answer = func(diff, 10,5);    /* 第 1 引数が diff 関数のアドレス */
    printf("%f¥n", answer);
    return 0;
}
```

【コールバック関数】

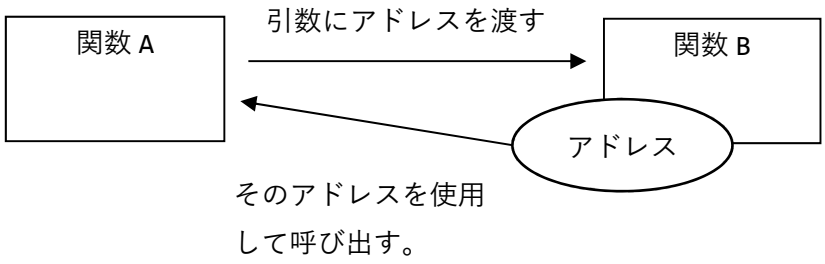
コールバックとは、一旦相手に電話をかけて、再度相手側からかけ直してもらうことです。

電話ではこのようになります。

「A さんは B さんに電話をかける」→「A さんは B さんに電話番号を教える」→「A さんは電話を切る」→「B さんが A さんに電話をかける」

コールバック関数の場合は下記のようになります。

「関数 A は関数 B を呼び出す」→「関数 A は関数 B にアドレスを渡す」→「関数 A を終了する」→「関数 B が関数 A を呼び出す」



コールバックを使用すれば関数同士が相互に呼び出せます。