C++ ポインタ

[C++][C++ によるオブジェクト指向プログラミング]

ポインタ型

・ポインタは、変数およびマシンアドレスを参照するのに使用
・配列および文字列の処理に緊密に結びついている
・C++ の配列は、連続したメモリに関連付けれられたインデックス付け可能な特殊なポインタ

```
int* p;
```

・上記は、p を int へのポインタ型として宣言している

例

```
#include <iostream>

using namespace std;

int main()
{
    int    i  = 5;
    int* p = &i;  // ポインタを i のアドレスに初期化

    cout << *p << endl;  // ポインタが指す変数の値を出力

    int j = *p + 1;      // ポインタが指す変数の値に +1 した値で j を初期化

    cout << j << endl;

    p = &j;              // j のアドレスをポインタに設定

    cout << *p << endl;  // p は j を指す
 }
```

結果

```
W:\oop02\Debug>oop02.exe
5
6
6
```

ポインタベースの参照渡し

1. 関数ヘッダでポインタパラメータを宣言
2. 関数本体でポインタを間接参照
3. 関数呼び出し時に、引数としてアドレスを渡す

例

```
#include <iostream>

using namespace std;

void sort();

// ポインタパラメータを宣言
void order(int*, int*);
```

```
int main()
{
    sort();
}

void sort()
{
    int nums[] = { 2, 34, 656, 767, 893, 123, 5, 7, 892, 4, 6, 81, 1134, 56, 3 };
    int len = sizeof(nums) / sizeof(nums[0]);

    for (int i=0; i<len; i++) {
        for (int j=i; j<len; j++) {
            // 引数としてアドレスを渡す
            order(&nums[i], &nums[j]);
        }
    }
    for (int i=0; i<len; i++) {
        cout << nums[i] << endl;
    }
}

void order(int* n, int* m)
{
    int tmp;
    // ポインタを間接参照する
    if (*n > *m) {
        tmp = *n;
        *n  = *m;
        *m  = tmp;
    }
}
```

結果

```
W:\oop02\Debug>oop02.exe
2
3
4
5
6
7
34
56
81
123
656
767
892
893
1134
```

汎用ポインタ

・void* gp のような汎用ポインタへは任意の型のアドレスを代入できる
・間接参照はできない

例

```
void* vp;       // 汎用ポインタ
int*  ip;       // int ポインタ
char* cp;       // char ポインタ

int  i = 10;
char c = 'a';

// それぞれの型のポインタに値を設定
ip = &i;
cp = &c;
cout << *ip << "," << *cp << endl;
```

```
vp = &i;                                    // int ポインタを 汎用ポインタに変換
int j = *reinterpret_cast<int*>(vp);        // 汎用ポインタを int ポインタに変換
cout << j << endl;

vp = &c;                                    // char ポインタを 汎用ポインタに変換
char d = *reinterpret_cast<char*>(vp);      // 汎用ポインタを char ポインタに変換
cout << d << endl;

// error C2100: 間接指定演算子 (*) の使い方が正しくありません。
// cout << *vp << endl;                     // 汎用ポインタの間接参照不可

// 'char *' から 'int *' に変換できません。
// ip = cp;                                 // 異なる型のポインタへの変換不可
```

結果

```
10,a
10
a
```

配列とポインタ

- 配列の名前それ自身ではアドレス (ポインタ値) である
- 配列は定数ポインタとみなすことのできる特定の固定アドレス
- 配列が宣言されるとコンパイラはすべての要素を記憶するのにたる記憶領域を割り当てる
- 配列のベースアドレスはメモリの先頭位置でインデックス 0 のアドレスと等しい

例

```
const int SIZE = 5;
int nums[SIZE] = {1,2,3,4,5};

// 連続したアドレスを確保
for (int i=0; i<SIZE; i++) {
    cout << &nums[i] << ",";
}
cout << endl;

// 配列のベースアドレスは先頭アドレス
cout << &nums << endl;

// nums + 1 と &nums「1] は等しい
int* ip = nums + 1;
cout << ip << "=" << *ip << endl;
int* ip2 = &nums[1];
cout << ip2 << "=" << *ip2 << endl;
```

結果

```
0022FDDC,0022FDE0,0022FDE4,0022FDE8,0022FDEC,
0022FDDC
0022FDE0=2
0022FDE0=2
```

参照

- 参照宣言は、オブジェクトの別名 (エイリアス) と宣言し、簡単な形式での参照私を可能にする
- 以下の例で、i および ri はお互いのエイリアスで同じオブジェクトを指す
- 宣言された時点でそれに関連付けられたアドレスと記憶領域を持ち、後からは変更できない

例

```
int   i     = 5;
int*  pi  = &i;        // i のポインタ
int&  ri  = i;        // i の参照

cout << i << "," << *pi << "," << ri << endl;
cout << "&i=" << &i << ", &pi=" << &pi << ", &ri=" << &ri << endl;

*pi = 9;
cout << i << "," << *pi << "," << ri << endl;

i = 10;
cout << i << "," << *pi << "," << ri << endl;

// 一旦宣言されると、別のオブジェクトの参照にはなれない
int j = 11;
ri = j;
cout << i << "," << *pi << "," << ri << endl;

// ポインタはアドレスを記憶した変数だが、参照はエイリアスであり同じオブジェクトを指す
cout << "&i=" << &i << ", &pi=" << &pi << ", &ri=" << &ri << ", &j=" << &j << endl;
```

結果

```
5,5,5
&i=002AFDFC, &pi=002AFDF0, &ri=002AFDFC
9,9,9
10,10,10
11,11,11
&i=002AFDFC, &pi=002AFDF0, &ri=002AFDFC, &j=002AFDD8
```

参照渡し

・上の例で行った「ポインタベースの参照渡し」を参照渡しに書き換える
・ポインタベースの参照渡しでは、ポインタに別のアドレスを代入できてしまうが、参照ではそのようなことはない
・ポインタベースの参照渡しでは、間接参照が必要だが、参照では不要

例

```
void ref_sort()
{
    int nums[] = { 2, 34, 656, 767, 893, 123, 5, 7, 892, 4, 6, 81, 1134, 56, 3 };
    int len = sizeof(nums) / sizeof(nums[0]);

    for (int i=0; i<len; i++) {
        for (int j=i; j<len; j++) {
            // 引数として値自身（参照）を渡す
            ref_order(nums[i], nums[j]);
        }
    }
    for (int i=0; i<len; i++) {
        cout << nums[i] << endl;
    }
}

void ref_order(int& n, int& m)
{
    int tmp;
    // ポインタを間接参照する必要はない
    if (n > m) {
        tmp = n;
        n   = m;
        m   = tmp;
    }
}
```

結果

```
2
3
4
5
6
7
34
56
81
123
656
767
892
893
1134
```