1. Các cách thức đặt tên

1.1. Pascal Case

Tên được viết theo Pascal Case (hay còn gọi là Upper Camel Case) có các tính chất sau:

- **Các chữ cái đầu mỗi từ được viết hoa.**
- **Các chữ còn lại được viết thường.**

1.2. Camel Case

Tên được viết theo Camel Case (hay còn gọi là Lower Camel Case) có các tính chất sau:

- Chữ cái đầu tiên được viết thường.
- **Các chữ cái đầu mỗi từ được viết hoa.**
- Các chữ cái còn lại được viết thường.

```
UpperCamelCase      // Pascal Case
GetThisObject       // Pascal Case

lowerCamelCase      // Camel Case
getThatObject       // Camel Case
```

2. Các quy ước đặt tên trong chương trình

Quy ước 0: Tên các định danh (hằng, biến, hàm, bảng, trường, …) trong ứng dụng phải thể hiện được ý nghĩa của nó.

2.1. Quy ước đặt tên hằng

Quy ước 1.1: Tên hằng được viết hoa toàn bộ, các từ cách nhau bằng ký tự '_'.

```
const int   NumberOfElements 100        // Sai.
const int   NUMBEROFELEMENTS 100        // Sai.
const int   NUMBER_OF_ELEMENTS 100      // Dung.
```

2.2. Quy ước đặt tên biến

Quy ước 2.1: Tên biến kiểu dữ liệu định sẵn được viết theo Pascal Case và được đặt theo phong cách Hungarian (có phần tiếp đầu ngữ - prefix, thể hiện kiểu dữ liệu của biến).

```
int     ituso, imauso;      // Sai.
int     iTuso, iMauso;      // Sai.
int     iTuSo, iMauSo;      // Dung..
```

Bảng tiếp đầu ngữ ứng với các kiểu dữ liệu:

Kiểu dữ liệu số		
char – c short – s	char short	cKyTu; sSoNguyenNgan;

int – i long – l float – f double – d	int long float double int	iSoNguyen; lSoNguyenDai; fSoThuc; dSoThucDai; nSo;

Kiểu dữ liệu luận lý		
bool – b	bool	bLuanLy;

Kiểu dữ liệu mảng		
[] – arr	int HocSinh	arrSoNguyen[50]; arrDanhSach[50];

Kiểu dữ liệu chuỗi		
char *, char [] – str	char char	*strChuoi; strChuoi[50];

Kiểu dữ liệu con trỏ		
* - p	int HocSinh	*pConTro; *pDanhSach;

Quy ước 2.2: Tên biến kiểu dữ liệu tự định nghĩa (struct, class) được viết theo Camel Case và không có tiếp đầu ngữ.

```
HocSinh   hsHocSinh;   // Sai
HocSinh   hocSinh;     // Dung
```

Quy ước 2.3: Tên thuộc tính của lớp tuân thủ các quy ước 2.1, 2.2 và **có thêm tiếp đầu ngữ "m_"**;

```
class HocSinh
{
    private string      strHoTen;     // Sai
    private string      m_strHoTen;   // Dung

    private LopHoc      lopHoc;       // Sai
    private LopHoc      m_lopHoc;     // Dung
};
```

2.3. Quy ước đặt tên hàm

Quy ước 3.1: Tên hàm được viết theo Camel Case và phải là động từ thể hiện hành động cần thực hiện.

```
int checkforbadvalue(long lValue)       // Sai.
int CheckForBadValue(long lValue)       // Sai.
int checkForBadValue(long lValue)       // Dung.
```

```
int badValue(long lValue)          // Sai.
int checkForBadValue(long lValue)  // Dung.
```

2.4. Quy ước đặt tên kiểu dữ liệu tự định nghĩa

Quy ước 4.1: Tên kiểu dữ liệu tự định nghĩa được viết theo Pascal Case và phải là danh từ.

```
class phanso                       // Sai.
class PHANSO                       // Sai.
class Phanso                       // Sai.
class PhanSo                       // Dung.

class TinhDiemHocSinh              // Sai.
class HocSinh                      // Dung.
```

2.5. Quy ước viết câu lệnh

Quy ước 5.1: **Mỗi câu lệnh được viết riêng trên một dòng.**

```
// Sai.
x = 3; y = 5;

// Dung.
x = 3;
y = 5;

// Sai.
if (a > b) cout << "a lon hon b";
else cout << "a nho hon b";

// Dung.
if (a > b)
    cout << "a lon hon b";
else
    out << "a nho hon b";

// Sai.
for (int i = 0; i < n; i++) x = x + 5;

// Dung.
for (int i = 0; i < n; i++)
    x = x + 5;
```

Quy ước 5.2: **Các dấu '{', '}' được viết riêng trên một dòng.**

```
// Sai.
void Swap(int &a, int &b) {
        int c = a;
        a = b;
        b = c;
}

void Swap(int &a, int &b)
{       int c = a;
        a = b;
        b = c;
}
```

```
// Dung.
void Swap(int &a, int &b)
{
        int c = a;
        a = b;
        b = c;
}
```

Quy ước 5.3: **Các câu lệnh if, while, for được viết riêng trên một đoạn.**

```
// Sai.
if (a > b)
        cout << "a lon hon b";
for (int i = 0; i < n; i++)
        x = x + 5;
k = k * x;
```

```
// Dung.
if (a > b)
        cout << "a lon hon b";

for (int i = 0; i < n; i++)
        x = x + 5;

k = k * x;
```

Quy ước 5.4: Các câu lệnh cùng thực hiện một công việc được viết riêng trên một đoạn.

```
// Sai.
int c = a;
a = b;
b = c;
k = k * a;
x = b + c;
```

```
// Dung.
int c = a;
a = b;
b = c;

k = k * a;
x = b + c;
```

2.6. Quy ước về cách khoảng

Quy ước 6.1: Các câu lệnh nằm giữa dấu '{', '}' được viết cách vào một khoảng tab.

```
// Sai.
void Swap(int &a, int &b)
{
int c = a;
a = b;
b = c;
}
```

```
// Dung.
void Swap(int &a, int &b)
{
        int c = a;
        a = b;
        b = c;
}
```

Quy ước 6.2: Các câu lệnh ngay sau if, else, while, for được viết cách vào một khoảng tab.

```
// Sai.
if (a > b)
cout << "a lon hon b";
else
cout << "a nho hon b";

for (int i = 0; i < n; i++)
x = x + 5;
```

```
// Dung.
if (a > b)
        cout << "a lon hon b";
else
        cout << "a nho hon b";

for (int i = 0; i < n; i++)
        x = x + 5;
```

Quy ước 6.3: Xung quanh các toán tử 2 ngôi viết cách một khoảng trắng.

```
x=x+5*a-c;          // Sai.
x = x + 5 * a - c;  // Dung.

if (a>=b)   // Sai.
if (a >= b) // Dung.
```

Quy ước 6.4: Sau các dấu ‘,’, ‘;’ viết cách một khoảng trắng.

```
void CalculateValues(int a,int b,int c);    // Sai.
void CalculateValues(int a, int b, int c);  // Dung.

for (int i = 0;i < n;i++)     // Sai.
for (int i = 0; i < n; i++)   // Dung.
```

2.7. Quy ước viết chú thích

Quy ước 7.1: Chú thích phải rõ ràng, dễ hiểu và diễn giải được ý nghĩa của đoạn lệnh.

Quy ước 7.2: Đầu mỗi struct, class, hàm phải có chú thích diễn giải ý nghĩa của nó.

```
// Vi du chu thich so sai.
// Merge sort, gan : n * log(2)n, can mang phu b
void msort(int a[], int n, int l, int r, int b[])
{
    int       m, i, j, k;

    m = (l + r) / 2;

    if (l < m)
            msort(a, n, l, m, b);

    if (m + 1 < r)
```

```
                msort(a, n, m + 1, r, b);

        for (i = l; i <= m; i++)
                b[i] = a[i];

        for (i = m + 1; i <= r; i++)
                b[i] = a[m + 1 + r - i];

        for (i = l, j = l, k = r; i <= r; i++)
                if (b[j] < b[k])
                        a[i] = b[j++];
                else
                        a[i] = b[k--];
}

// Vi du chu thich ro rang, day du.
// Merge sort, gan : n * log(2)n, can mang phu b
void msort(int a[], int n, int l, int r, int b[])
{
        int     m, i, j, k;

        // Lay vi tri giua cua a
        m = (l + r) / 2;

        // Thuc hien merge sort tren a tu vi tri l den m
        if (l < m)
                msort(a, n, l, m, b);

        // Thuc hien merge sort tren a tu vi tri m + 1 den r neu m < r
        if (m + 1 < r)
                msort(a, n, m + 1, r, b);

        // Do cac phan tu cua a tu vi tri l den m co thu tu vao b
        for (i = l; i <= m; i++)
                b[i] = a[i];

        // Do cac phan tu cua a tu vi tri m + 1 den r co thu tu vao b theo thu tu
        // nguoc
        for (i = m + 1; i <= r; i++)
                b[i] = a[m + 1 + r - i];

        // Tron b tu vi tri l den m co thu tu va b tu vi tri r den m + 1 co thu tu vao a
        for (i = l, j = l, k = r; i <= r; i++)
                if (b[j] < b[k])
                        a[i] = b[j++];
                else
```

```
                a[i] = b[k--];
}
```

Quy ước 7.3: Dùng dấu chú thích từng dòng thay vì dấu chú thích đầu và cuối đoạn.

```
// Sai.
/* void Swap(int &a, int &b)
{
        int c = a;
        a = b;
        b = c;
} */
```

```
// Dung.
//void Swap(int &a, int &b)
//{
//      int c = a;
//      a = b;
//      b = c;
//}
```