

## Báo cáo training Lập trình hướng đối tượng

Họ và tên	:	Nguyễn Trung Hiếu
Ngày tháng	:	10/04/2023
Người hướng dẫn	:	Vũ Xuân Tuyền

# Mục lục:

<b>1. Lập trình hướng đối tượng</b>	3
1.1 Thế nào là lập trình hướng đối tượng	3
1.2 Tính đóng gói	3
1.3 Các loại biến Public – Private – Protected	4
1.4 Mảng (Array) trong C	5
1.5 Ô nhớ	6
1.6 Con trỏ trong C	7
1.7 Cấp phát động	8
1.8 Hàm tạo và hàm hủy	9
1.9 Hàm bạn và lớp bạn	10
1.10 Nạp chồng hàm và nạp chồng toán tử	11
1.11 Tính kế thừa	13
<b>2. Chương trình cờ Caro:</b>	14
2.1 Ý tưởng xây dựng chương trình	14
2.2 Yêu cầu khi viết chương trình	14
2.3 Các lớp và hàm dự kiến sẽ có trong chương trình	14
2.4 Flow chương trình	15
2.5 Code	15
2.6 Kết quả chạy chương trình	21
2.7 Đánh giá và rút ra nhận xét	22
2.8 Tài liệu tham khảo	22

## 1. Lập trình hướng đối tượng

### 1.1 Thế nào là lập trình hướng đối tượng

OOP là một phương pháp lập trình khác bên cạnh những phương pháp lập trình cũ, OOP có nhiều ưu điểm nổi trội về quản lý dữ liệu cũng như khả năng tương tác giữa các hàm. Bên cạnh đó OOP cũng đề cao tính bảo mật dữ liệu trong chương trình.

OOP bao gồm 2 thông tin là Thuộc tính (attribute) và Phương thức (method):

1. Thuộc tính là những đặc điểm của object đó ví dụ như với một chiếc xe thì là hãng xe, loại xe, số cây đã đi, ...
2. Phương thức là các tương tác của các thuộc tính với các vật, các thuộc tính xung quanh

Các ưu điểm của OOP:

1. Code re-usability: Khả năng tái sử dụng code
2. Data redundancy: Dư thừa dữ liệu - Cùng một loại dữ liệu được xuất hiện nhiều nơi trong database (Chưa hiểu vì sao lại là ưu điểm)
3. Code maintenance: Duy trì code - Hiểu nhầm na là khả năng sử dụng cùng 1 đoạn code vào nhiều việc khác nhau
4. Security: Tính bảo mật
5. Design benefit: Lợi ích thiết kế
6. Better productivity: Tăng năng suất công việc
7. Easy troubleshooting: Dễ dàng tìm lỗi
8. Problems solving: Khả năng giải quyết vấn đề

### 1.2 Tính đóng gói

Các dữ liệu và phương thức có liên quan với nhau được đóng gói thành các lớp để tiện cho việc quản lý và sử dụng. Nghĩa là mỗi lớp được xây dựng để thực hiện một nhóm chức năng đặc trưng của riêng lớp đó

Ngoài ra, đóng gói còn để che giấu một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không thể nhìn thấy (private

**1.3 Các loại biến Public – Private – Protected**

Public:

- Có thể được sử dụng ở tất cả mọi nơi trong chương trình
- Được sử dụng khi mà muốn sử dụng biến đó cho toàn bộ chương trình

Protected:

- Chỉ có thể sử dụng ở các lớp con có kế thừa thuộc tính của lớp cha mẹ
- Được sử dụng khi biến đó liên quan đến nhiều class có liên quan đến nhau, có tính kế thừa nhau

Private:

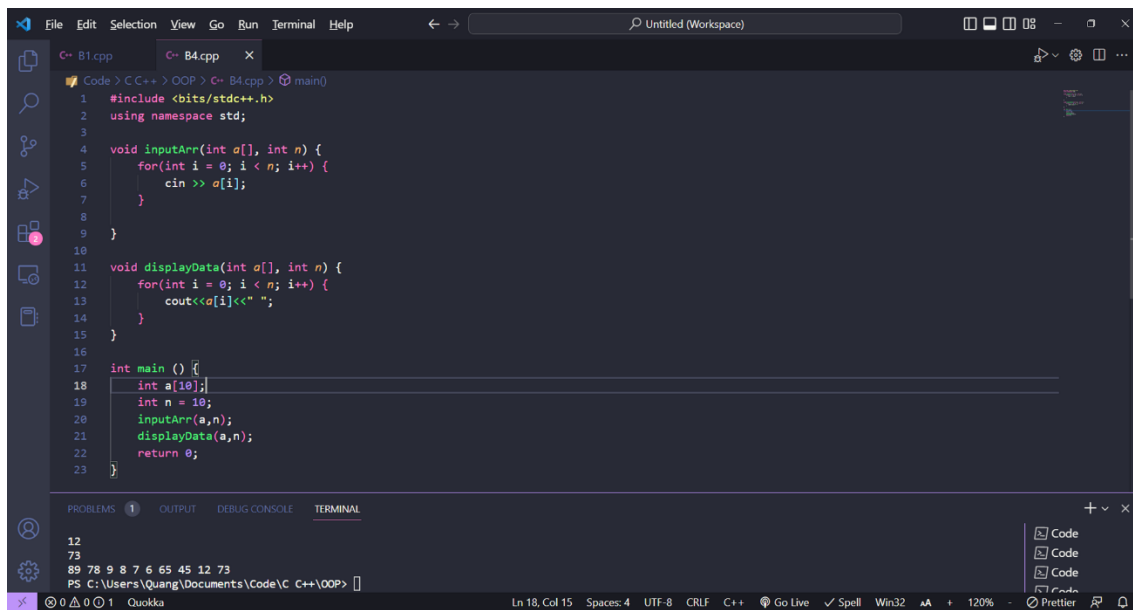
- Chỉ có thể sử dụng trong chính lớp cha mẹ
- Được sử dụng khi chỉ sử dụng biến đó trong lớp đó và không muốn nó có thể được truy cập từ bất cứ nơi nào trong chương trình nhằm nâng cao tính bảo mật

Khả năng truy xuất	Biến Private	Biến Protected	Biến Public
Khả năng truy xuất trong kế thừa với Public			
Truy xuất từ chính lớp đó	Có	Có	Có
Truy xuất từ lớp kế thừa	Không	Có	Có
Truy xuất từ lớp kế thừa đời 2	Không	Có	Có
Khả năng truy xuất trong kế thừa với Protected			
Truy xuất từ chính lớp đó	Có	Có	Có
Truy xuất từ lớp kế thừa	Không	Có	Có (Dưới dạng các biến Protected)
Truy xuất từ lớp kế thừa đời 2	Không	Có	Có
Khả năng truy xuất trong kế thừa với Private			
Truy xuất từ chính lớp đó	Có	Có	Có
Truy xuất từ lớp kế thừa	Không	Có (Dưới dạng các biến Private)	Có (Dưới dạng các biến Private)
Truy xuất từ lớp kế thừa đời 2	Không	Không	Không

## 1.4 Mảng (Array) trong C

Array là một cấu trúc dữ liệu trong các ngôn ngữ lập trình gồm hữu hạn các phần tử có cùng kiểu dữ liệu. Mỗi phần tử của mảng có thể được truy xuất dựa vào chỉ số của nó trong mảng. Chỉ số của các phần tử trong mảng bắt đầu từ số 0.

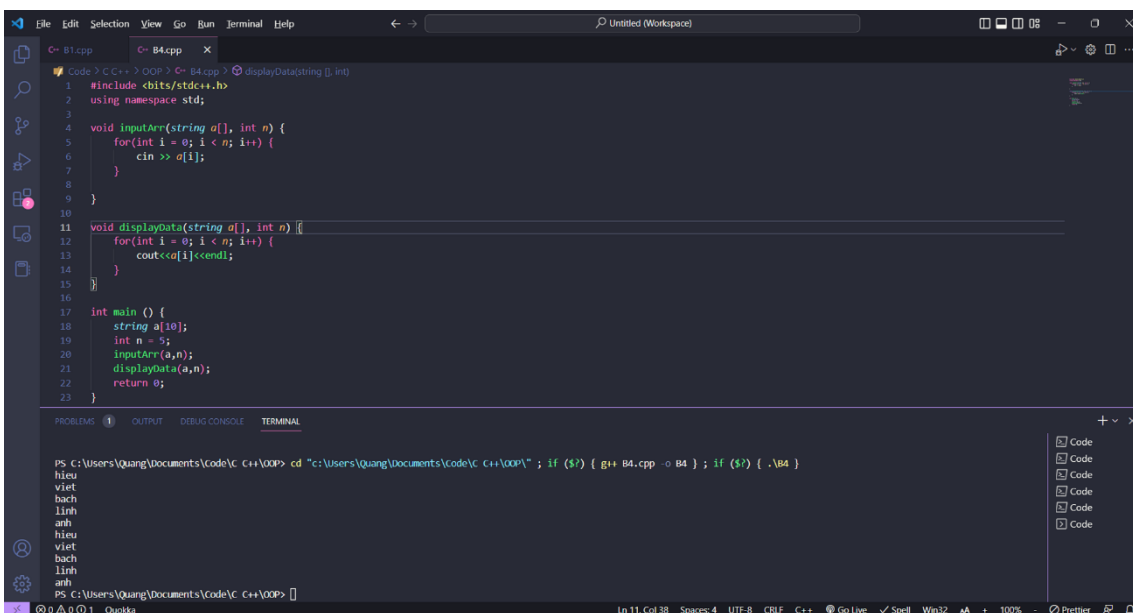
Ví dụ về mảng số nguyên (Array int):



```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 void inputArr(int a[], int n) {
5     for(int i = 0; i < n; i++) {
6         cin >> a[i];
7     }
8 }
9
10
11 void displayData(int a[], int n) {
12     for(int i = 0; i < n; i++) {
13         cout<<a[i]<<" ";
14     }
15 }
16
17 int main () {
18     int a[10];
19     int n = 10;
20     inputArr(a,n);
21     displayData(a,n);
22     return 0;
23 }
```

Terminal output: 89 78 9 8 7 6 65 45 12 73

Ví dụ về mảng chuỗi (Array string):



```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 void inputArr(string a[], int n) {
5     for(int i = 0; i < n; i++) {
6         cin >> a[i];
7     }
8 }
9
10
11 void displayData(string a[], int n) {
12     for(int i = 0; i < n; i++) {
13         cout<<a[i]<<endl;
14     }
15 }
16
17 int main () {
18     string a[10];
19     int n = 5;
20     inputArr(a,n);
21     displayData(a,n);
22     return 0;
23 }
```

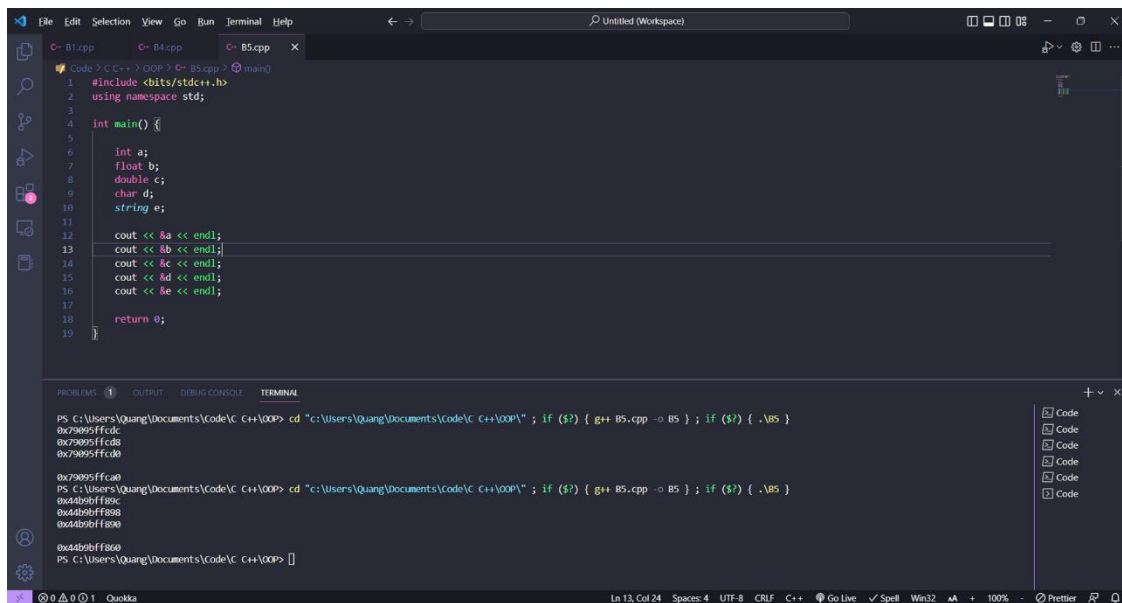
Terminal output: hieu, viet, bach, anh, hieu

## 1.5 Ô nhớ

Ô nhớ là đơn vị nhỏ nhất có thể đánh địa chỉ để quản lý bộ nhớ máy tính. Đơn vị này gồm 8 bits liên tiếp (1 bytes). Mỗi ô nhớ đều được đánh địa chỉ để CPU có thể truy xuất vào bộ nhớ.

Số lượng địa chỉ có thể đánh cho ô nhớ phụ thuộc vào độ rộng thanh ghi trong CPU. Ví dụ: CPU 32 bit thì có  $2^{32}$  địa chỉ có thể đánh cho các ô nhớ trong RAM, 0x7c9c7ffb5c, 0x7c9c7ffb58

Địa chỉ ô nhớ dùng hệ hex (hệ 16) khi ta khai báo biến thì máy sẽ cấp phát cho ta một vài ô nhớ liên nhau (có thể gọi là vùng nhớ) có kích thước tương ứng với kiểu dữ liệu ta khai báo cho biến.



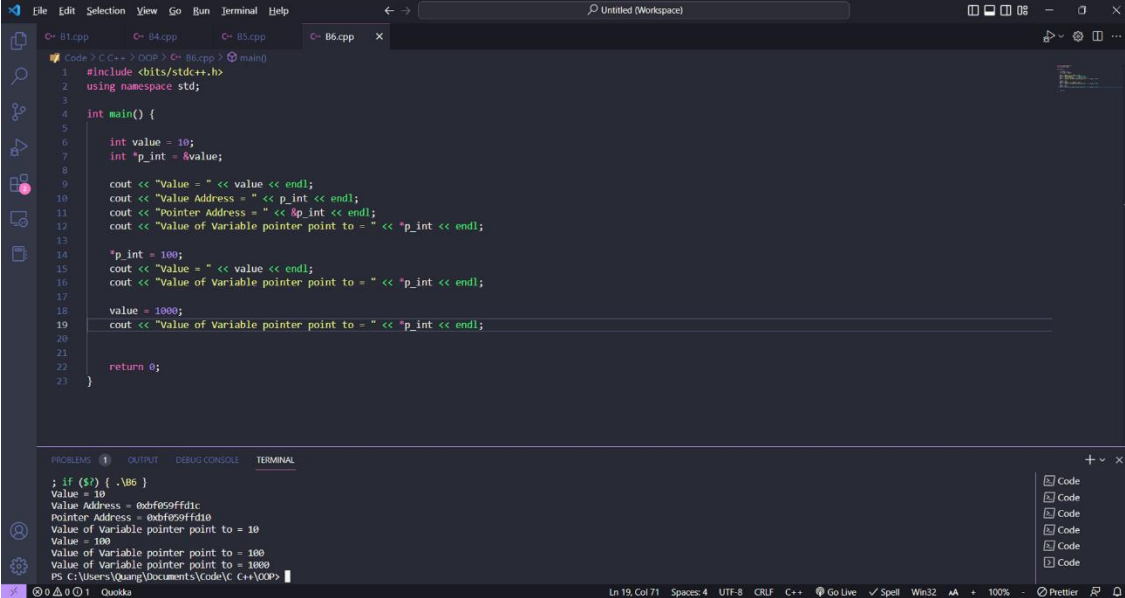
```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main() {
5
6     int a;
7     float b;
8     double c;
9     char d;
10    string e;
11
12    cout << &a << endl;
13    cout << &b << endl;
14    cout << &c << endl;
15    cout << &d << endl;
16    cout << &e << endl;
17
18    return 0;
19 }
```

```
PS C:\Users\Quang\Documents\Code\C++\00P> cd "C:\Users\Quang\Documents\Code\C++\00P\" ; if ($?) { g++ B5.cpp -o B5 } ; if ($?) { .\B5 }
0x79095ffcd8
0x79095ffcd8
0x79095ffcd8
0x79095ffca0
PS C:\Users\Quang\Documents\Code\C++\00P> cd "C:\Users\Quang\Documents\Code\C++\00P\" ; if ($?) { g++ B5.cpp -o B5 } ; if ($?) { .\B5 }
0x4b9bfff80c
0x4b9bfff80c
0x4b9bfff800
0x4b9bfff800
PS C:\Users\Quang\Documents\Code\C++\00P> 
```

Có thể thấy với mỗi lần chạy code thì ta có một địa chỉ ô nhớ khác nhau. Lí do là bởi mỗi khi chạy chương trình máy tính sẽ kiểm tra xem còn vùng nhớ nào trống rồi mới cấp phát cho ta và tại các thời điểm khác nhau thì địa chỉ các vùng nhớ trống là khác nhau. Và các địa chỉ ô nhớ đều bắt đầu bằng 0x bởi đó là quy ước của ngôn ngữ C/C++.

## 1.6 Con trỏ trong C

Con trỏ trong C có thể hiểu đơn giản cũng là 1 biến có thể khai báo, lưu trữ dữ liệu, khởi tạo và có địa chỉ của riêng nó. Tuy nhiên khác với các biến thông thường khác thì con trỏ là biến trỏ tới địa chỉ của 1 biến khác. Điều này có nghĩa là giá trị của của biến con trỏ sẽ là 1 địa chỉ.



```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main() {
5
6     int value = 10;
7     int *p_int = &value;
8
9     cout << "Value = " << value << endl;
10    cout << "Value Address = " << p_int << endl;
11    cout << "Pointer Address = " << &p_int << endl;
12    cout << "Value of Variable pointer point to = " << *p_int << endl;
13
14    *p_int = 100;
15    cout << "Value = " << value << endl;
16    cout << "Value of Variable pointer point to = " << *p_int << endl;
17
18    value = 1000;
19    cout << "Value of Variable pointer point to = " << *p_int << endl;
20
21
22    return 0;
23 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
if ($?) { .\06 }
Value = 10
Value Address = 0xbff059ffdc
Pointer Address = 0xbff059ffd0
Value of Variable pointer point to = 10
Value = 100
Value of Variable pointer point to = 100
Value of Variable pointer point to = 1000
Ps C:\Users\Quang\Documents\Code\C C++\00P
```

Vì mảng 1 chiều là tập hợp các phần tử có cùng 1 kiểu dữ liệu nên khi sử dụng 1 con trỏ có cùng kiểu giữ liệu ta có thể quản lý được toàn bộ các phần tử có trong mảng đó.

## 1.7 Cấp phát động

Cấp phát động được sử dụng khi khai báo 1 biến hoặc 1 mảng để được cung cấp vùng nhớ sử dụng cho biến hoặc mảng đó. Kiểu cấp phát này được sử dụng khi ta chưa biết chắc chắn kích thước của biến hoặc mảng.

Cú pháp: `new <kiểu dữ liệu>;`

Khi khai báo cấp phát động sẽ trả về một con trỏ trỏ tới địa chỉ vùng nhớ được cấp phát. Ta sẽ phải khai báo 1 con trỏ lưu địa chỉ được cấp đó để thuận tiện sử dụng. VD:  
`int *ptr = new int;`

Ta sử dụng cấp phát động khi nào

- Ví dụ khi ta tạo một mảng để lưu trữ thông tin các nhân viên trong 1 công ty. Nếu sử dụng cấp phát tĩnh ta sẽ phải đoán xem công ty có bao nhiêu người và khai báo kích thước của mảng ứng với số nhân viên trong công ty. Tuy nhiên nếu khai báo như vậy sẽ có 2 trường hợp:
  - o Kích thước ta khai báo nhỏ hơn so với số nhân viên trong công ty thì sẽ rất khó để xử lí.
  - o Kích thước ta khai báo lớn hơn so với số nhân viên trong công ty dẫn đến lãng phí bộ nhớ đã được cấp phát.

So sánh cấp phát tĩnh và cấp phát động:

Cấp phát tĩnh	Cấp phát động
Phải xác định kích thước biến hoặc mảng ngay trong quá trình biên dịch	Kích thước biến hoặc mảng được xác định trong quá trình chạy chương trình
Các giá trị của biến được lưu trữ trên stack	Các giá trị được lưu trữ trên Heap
Các vùng nhớ sẽ được cung cấp khi chạy chương trình và thu hồi khi chương trình kết thúc	Khi không cần sử dụng vùng nhớ phải chủ động thu hồi tránh lãng phí tài nguyên. (hàm delete)

Chú ý khi dùng cấp phát động

- Tránh sử dụng nhiều con trỏ cũng trỏ vào 1 vùng nhớ dẫn tới bị loạn
- Rò rỉ bộ nhớ: Điều này xảy ra khi ta cấp phát động nhưng quên không lưu địa chỉ được cấp phát hay khi ta sử dụng xong nhưng không trả lại vùng nhớ cho hệ điều hành dẫn tới cấp phát nhiều lần gây tiêu tốn tài nguyên phần cứng



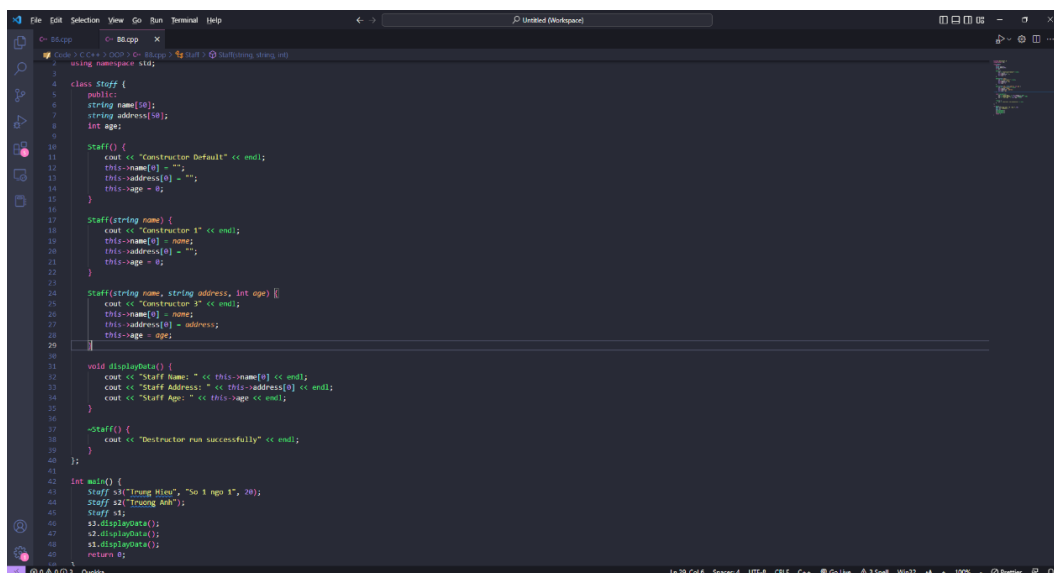
## 1.8 Hàm tạo và hàm hủy

Hàm tạo: Là hàm đặc biệt trong lớp dùng để khởi tạo cho các giá trị của đối tượng.

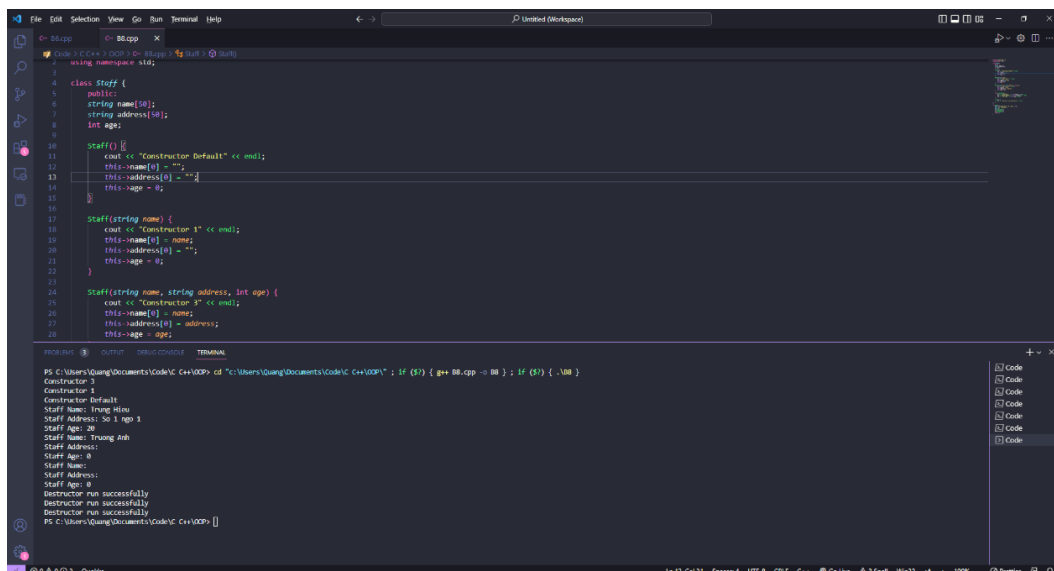
- Hàm tạo có tên trùng với tên lớp, nếu không có tham số truyền vào thì là hàm tạo mặc định ngược lại là hàm tạo thông thường. Trong một lớp có thể có nhiều hàm tạo và các hàm tạo phải được khai báo với phạm vi public
- Hàm tạo sao chép là một hàm được chép lại tất cả các thuộc tính của một đối tượng trong cùng lớp đã được tạo ra trước nó

Hàm hủy: Cũng là một hàm đặc biệt trong lớp giống hàm tạo nhưng dùng để xóa đối tượng trong lớp

- Hàm hủy có tên trùng với tên lớp nhưng có dấu ~ đằng trước. Trong 1 lớp chỉ có duy nhất 1 hàm hủy. Hàm hủy không có tham số truyền vào cũng không có giá trị trả về.



```
1 using namespace std;
2
3 class Staff {
4 public:
5     string name[50];
6     string address[50];
7     int age;
8
9     Staff() {
10         cout << "Constructor Default" << endl;
11         this->name[0] = "";
12         this->address[0] = "";
13         this->age = 0;
14     }
15
16     Staff(string name) {
17         cout << "Constructor 1" << endl;
18         this->name[0] = name;
19         this->address[0] = "";
20         this->age = 0;
21     }
22
23     Staff(string name, string address, int age) {
24         cout << "Constructor 2" << endl;
25         this->name[0] = name;
26         this->address[0] = address;
27         this->age = age;
28     }
29
30     void displayData() {
31         cout << "Staff Name: " << this->name[0] << endl;
32         cout << "Staff Address: " << this->address[0] << endl;
33         cout << "Staff Age: " << this->age << endl;
34     }
35
36     ~Staff() {
37         cout << "Destructor run successfully" << endl;
38     }
39 };
40
41 int main() {
42     Staff s1("Trung Hiếu", "Số 1 Ngõ 1", 20);
43     Staff s2("Trương Anh");
44     Staff s3;
45     s3.displayData();
46     s2.displayData();
47     s1.displayData();
48     return 0;
49 }
```



```
PS C:\Users\Quang\Documents\Code\C++\V03P> cd "C:\Users\Quang\Documents\Code\C++\V03P\"; if ($?) { g++ 08.cpp -o 08 }; if ($?) { .\08 }
Constructor 2
Constructor 1
Constructor Default
Staff Name: Trung Hiếu
Staff Address: Số 1 Ngõ 1
Staff Age: 20
Staff Name: Trương Anh
Staff Address:
Staff Age: 0
Staff Name:
Staff Address:
Staff Age: 0
Destructor run successfully
Destructor run successfully
Destructor run successfully
PS C:\Users\Quang\Documents\Code\C++\V03P>
```

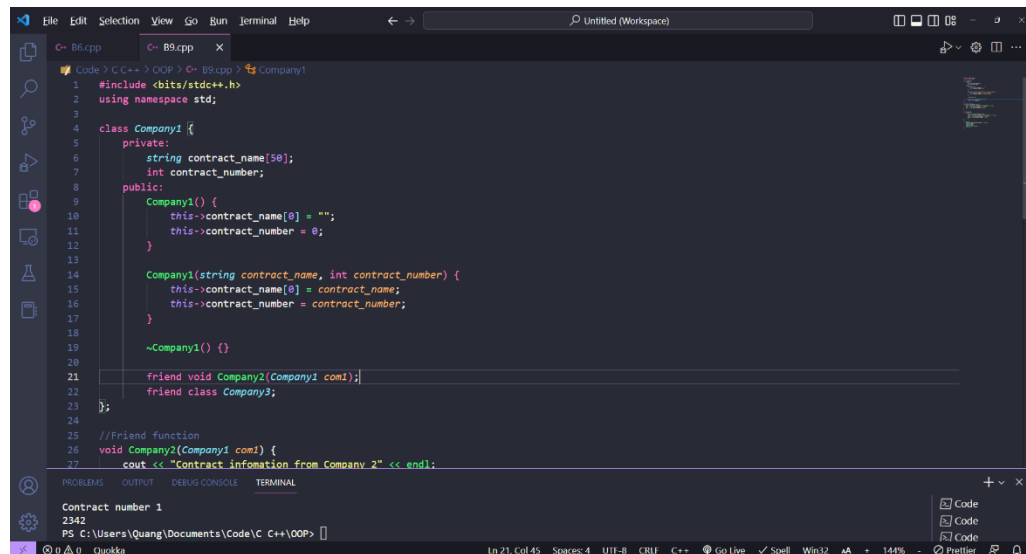
## 1.9 Hàm bạn và lớp bạn

Hàm bạn:

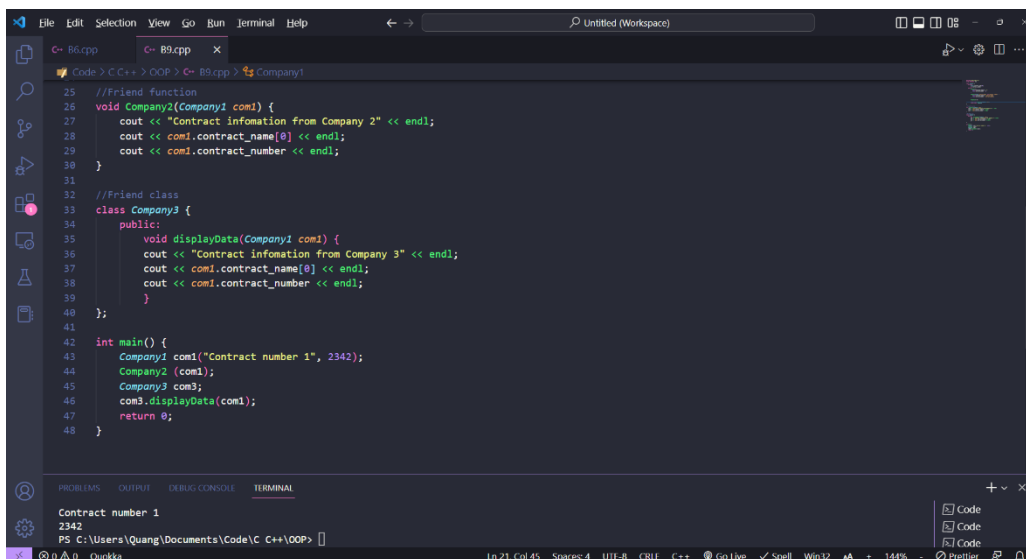
- Hàm bạn là một hàm được cấp phép để có thể truy cập vào vùng dữ liệu private của lớp. Hàm này có thể nằm ở trong hoặc ngoài lớp. Một lớp có thể có nhiều hàm bạn
- Cú pháp: friend <kiểu hàm> <tên hàm> (dữ liệu);

Lớp bạn:

- Tương tự như hàm bạn thì lớp bạn cũng được cấp phép để truy cập vào dữ liệu private của lớp. Có thể có nhiều lớp bạn tuy nhiên nếu B được khai báo là bạn A thì B có thể truy cập vào private của A nhưng điều ngược lại thì không đúng.
- Cú pháp: friend class <tên lớp bạn>;



```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class Company1 {
5     private:
6         string contract_name[50];
7         int contract_number;
8     public:
9         Company1() {
10             this->contract_name[0] = "";
11             this->contract_number = 0;
12         }
13
14         Company1(string contract_name, int contract_number) {
15             this->contract_name[0] = contract_name;
16             this->contract_number = contract_number;
17         }
18
19         ~Company1() {}
20
21         friend void Company2(Company1 com1);
22         friend class Company3;
23 };
24
25 //Friend function
26 void Company2(Company1 com1) {
27     cout << "Contract information from Company 2" << endl;
```



```
28     cout << "Contract information from Company 2" << endl;
29     cout << com1.contract_name[0] << endl;
30     cout << com1.contract_number << endl;
31 }
32
33 //Friend class
34 class Company3 {
35     public:
36         void displayData(Company1 com1) {
37             cout << "Contract information from Company 3" << endl;
38             cout << com1.contract_name[0] << endl;
39             cout << com1.contract_number << endl;
40         }
41 };
42
43 int main() {
44     Company1 com1("Contract number 1", 2342);
45     Company2 (com1);
46     Company3 com3;
47     com3.displayData(com1);
48     return 0;
49 }
```

```

25 //Friend function
26 void Company2(Company1 com1) {
27     cout << "Contract information from Company 2" << endl;
28     cout << com1.contract_name[0] << endl;
29     cout << com1.contract_number << endl;
30 }
31
32 //Friend class
33 class Company3 {
34 public:
35     //...
36 }
37
38 int main() {
39     Company1 com1("Contract 1", 12345);
40     Company2 com2(com1);
41     Company3 com3;
42     com3.display();
43     return 0;
44 }

```

Terminal Output:

```

PS C:\Users\Quang\Documents\Code\C++\OOP> cd "C:\Users\Quang\Documents\Code\C++\OOP\" ; if ($?) { g++ B9.cpp -o B9 } ; if ($?) { .\B9 }
Contract information from Company 2
Contract number 1
2342
Contract information from Company 3
Contract number 1
2342
PS C:\Users\Quang\Documents\Code\C++\OOP>

```

## 1.10 Nạp chồng hàm và nạp chồng toán tử

Nạp chồng hàm:

- Được sử dụng đối với các hàm có cùng một mục đích nhưng khác nhau về kiểu dữ liệu tham số đầu vào
- Các kiểu hàm không thể nạp chồng:
  - o Hàm chỉ khác nhau kiểu trả về
  - o Tham số hàm là kiểu typedef
  - o Tham số hàm là con trỏ và mảng (vì con trỏ và mảng là tương đương nhau)
  - o Dữ liệu đầu vào là const (chỉ được có const nếu dữ liệu đầu vào là con trỏ hoặc tham chiếu)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 //Function overloading
5 int add(int a, int b) {
6     return a+b;
7 }
8
9 double add(double a, double b) {
10    return a+b;
11 }
12
13 int main() {
14     cout << add(5, 8) << endl;
15     cout << add(5.5, 8.5) << endl;
16     return 0;
17 }

```

Terminal Output:

```

PS C:\Users\Quang\Documents\Code\C++\OOP> cd "C:\Users\Quang\Documents\Code\C++\OOP\" ; if ($?) { g++ B10.cpp -o B10 } ; if ($?) { .\B10 }
13
14
PS C:\Users\Quang\Documents\Code\C++\OOP>

```

## Nạp chồng toán tử:

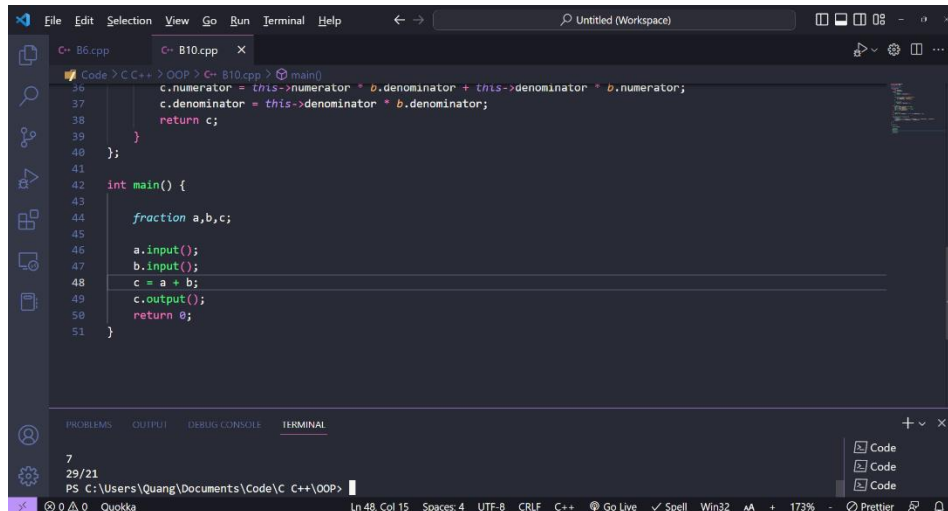
- Được sử dụng để có thể sử dụng các toán tử với các kiểu dữ liệu khác. Và không phải toán tử nào cũng có thể nạp chồng.
- Ví dụ như phép + chỉ áp dụng cho các số chứ không được áp dụng cho 1 chuỗi kí tự tuy nhiên với nạp chồng toán tử ta có thể sử dụng phép + với các kiểu dữ liệu khác
- Cú pháp:
  - o Toán tử 1 ngôi, 2 ngôi: <Kiểu trả về> operator <toán tử> (danh sách tham số) { }
  - o Toán tử nhập xuất: ostream& operator<<(istream& os, <ten class> obj) { return os; }

```

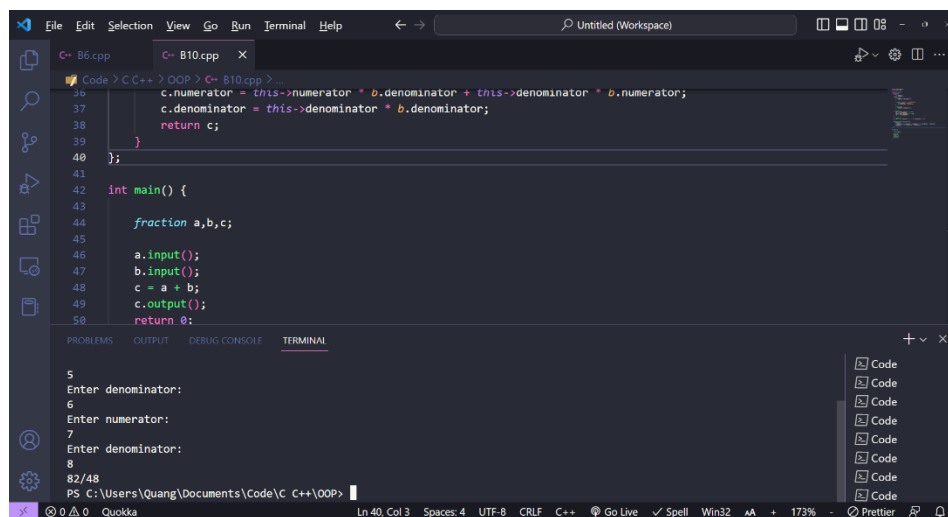
1  using namespace std;
2
3  //Operator overloading
4  class fraction {
5  private:
6      int numerator;
7      int denominator;
8  public:
9      fraction() {
10         numerator = denominator = 0;
11     }
12
13     fraction(int numerator, int denominator) {
14         this->numerator = numerator;
15         this->denominator = denominator;
16     }
17
18     ~fraction() {
19         numerator = denominator = 0;
20     }
21 }
  
```

```

21 }
22
23 void input() {
24     cout << "Enter numerator: " << endl;
25     cin >> this->numerator;
26     cout << "Enter denominator: " << endl;
27     cin >> this->denominator;
28 }
29
30 void output() {
31     cout << this->numerator << "/" << this->denominator << endl;
32 }
33
34 fraction operator +(fraction b) {
35     fraction c;
36     c.numerator = this->numerator * b.denominator + this->denominator * b.numerator;
37     c.denominator = this->denominator * b.denominator;
38     return c;
39 }
  
```



```
File Edit Selection View Go Run Terminal Help
C++ B6.cpp C++ B10.cpp
Code > C++ > OOP > C++ B10.cpp > main()
36 c.numerator = this->numerator * b.denominator + this->denominator * b.numerator;
37 c.denominator = this->denominator * b.denominator;
38 return c;
39 }
40 };
41
42 int main() {
43     fraction a,b,c;
44     a.input();
45     b.input();
46     c = a + b;
47     c.output();
48     return 0;
49 }
50
51
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
7
29/21
PS C:\Users\Quang\Documents\Code\C C++\OOP>
```



```
File Edit Selection View Go Run Terminal Help
C++ B6.cpp C++ B10.cpp
Code > C++ > OOP > C++ B10.cpp > ...
36 c.numerator = this->numerator * b.denominator + this->denominator * b.numerator;
37 c.denominator = this->denominator * b.denominator;
38 return c;
39 }
40 };
41
42 int main() {
43     fraction a,b,c;
44     a.input();
45     b.input();
46     c = a + b;
47     c.output();
48     return 0;
49 }
50
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
5
Enter denominator:
6
Enter numerator:
7
Enter denominator:
8
82/48
PS C:\Users\Quang\Documents\Code\C C++\OOP>
```

## 1.11 Tính kế thừa

Các lớp con sẽ kế thừa lại toàn bộ các thuộc tính cũng như các hàm có trong lớp cha. Tính kế thừa sinh ra nhằm tái sử dụng code tránh phải code đi code lại nhiều lần

Ví dụ như trong trường học sẽ có nhiều vị trí và mỗi vị trí sẽ có những thuộc tính giống nhau như họ tên, tuổi, địa chỉ, số điện thoại, ... ngoài ra sẽ có các thuộc tính khác như mã sinh viên, lương giáo viên, lương nhân viên, ... thì việc tạo một lớp cha là những người hoạt động trong trường sẽ bao gồm các đối tượng trên và thuộc tính cơ bản của chúng

## 2. Chương trình cờ Caro:

### 2.1 Ý tưởng xây dựng chương trình

Viết ra một chương trình Cờ Caro hiển thị lên màn hình cho 2 người chơi. Người chơi sử dụng chuột đánh vào vị trí muốn đánh trên màn hình và hiển thị quân cờ của người chơi đó. Người thắng là người có 5 quân cờ cùng màu liên tiếp nhau theo hàng ngang, dọc hoặc chéo. Khi đã có người chiến thắng hiển thị thông báo người chiến thắng lên màn hình

### 2.2 Yêu cầu khi viết chương trình

Học về đồ họa máy tính, sử dụng được ngôn ngữ C để vẽ được graphic lên trên màn hình

Học cách debug chương trình và test từng hàm nhỏ

Hiểu rõ được mình đang code gì, có flow chương trình rõ ràng trình bày bằng ngôn ngữ thông thường rồi chuyển hóa thành code

Từ m và n cho trước vẽ lên màn hình bàn cờ kích thước mxn ô vuông (vẽ sao cho cân xứng và đẹp với màn hình)

Học cách bắt sự kiện click chuột, bắt được tọa độ vừa click từ đó xác định vị trí trên bàn cờ và điền quân cờ vào vị trí vừa đánh

Mỗi nước đi đều được kiểm tra xem có ai thắng hay chưa, nếu có thì hiển thị thông báo lên màn hình

### 2.3 Các lớp và hàm dự kiến sẽ có trong chương trình

Người chơi (Player)

Tên (Thuộc tính)

Quân cờ (Thuộc tính)

Đánh cờ (Hành động)

Bàn cờ (Board)

Ô cờ (Đối tượng con)

Trống hoặc không trống (Thuộc tính)

Kích thước (Thuộc tính)

Chiều dài (Thuộc tính)

Chiều rộng (Thuộc tính)

Vẽ bàn cờ (Hành động)

Trọng tài (Check)

Kiểm tra thắng (Hành động)

Kiểm tra hòa (Hành động)

Thông báo kết quả lên màn hình (Hành động)

## 2.4 Flow chương trình

Bước 1: Vẽ và hiển thị bàn cờ

Bước 2: Người chơi 1 chọn 1 ô

- Nếu ô đó còn trống thì đánh X
- Nếu ô đó không còn trống hoặc người chơi đánh ra ngoài bàn cờ thì không có gì xảy ra

Bước 3: Kiểm tra điều kiện thắng và hòa

- Nếu có 5 ô liên tiếp ngang, dọc hoặc chéo cùng 1 người chơi thì tuyên bố người chơi đó thắng cuộc
- Nếu tất cả bàn cờ đã đầy mà vẫn không có người thắng thì thông báo kết quả hòa

Bước 4: Người chơi 2 lặp lại từ bước 2

## 2.5 Code

```
#include <iostream>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>

using namespace std;

const int ROW = 30;
const int COL = 50;
const int CELL_SIZE = 30;
const int TEXT_SIZE = 80;
char X[] = "X";
char O[] = "O";
int board[ROW][COL];
const int EMPTY = 0;
const int PLAYER_X = 1;
const int PLAYER_O = 2;
const int WIN_LENGTH = 5;
int turn = 1;

class Board {
public:
    // ve ban co
    void draw_board() {
        // ve nen ban co
        setfillstyle(SOLID_FILL, WHITE);
        bar(0, 0, COL * CELL_SIZE, ROW * CELL_SIZE);
```

```
// ve cot
setlinestyle(SOLID_LINE, 1, 1);
setcolor(BLACK);
for (int i = 0; i < COL; i++) {
    for (int j = 0; j < ROW; j++) {
        rectangle(j * CELL_SIZE, i * CELL_SIZE, (j + 1) * CELL_SIZE, (i +
1) * CELL_SIZE);
    }
}
// ve hang
for (int i = 0; i <= COL; i++) {
    line(i * CELL_SIZE, 0, i * CELL_SIZE, ROW * CELL_SIZE);
    line(0, i * CELL_SIZE, COL * CELL_SIZE, i * CELL_SIZE);
}
// khoi tao cac o co trong
for (int i = 0; i < ROW; i++) {
    for (int j = 0; j < COL; j++) {
        board[i][j] = EMPTY;
    }
}
}
};
```

```
class Player {
private:
    string name;
    int symbol;
public:
    Player(string _name, int _symbol) {
        name = _name;
        symbol = _symbol;
    }
    string getName() {
        return name;
    }
    int getSymbol() {
        return symbol;
    }

    void drawSymbol(int row, int col, char symbol) {
        // Tính tọa độ pixel của ô cờ tại vị trí (row, col)
        int x = col * CELL_SIZE - 25;
        int y = row * CELL_SIZE - 25;
```



```
// Thiết lập màu vẽ và độ dày đường viền
settextstyle(DEFAULT_FONT, HORIZ_DIR, TEXT_SIZE);

// Vẽ ký hiệu của người chơi vào ô cờ
if (turn % 2 == 1) {
    // Vẽ chữ X
    symbol = 1;
    setbkcolor(WHITE);
    setcolor(RED);
    outtextxy(x, y, X);
    setbkcolor(getbkcolor());
} else if (turn % 2 == 0) {
    // Vẽ hình tròn
    symbol = 2;
    setcolor(BLUE);
    outtextxy(x, y, O);
    setbkcolor(getbkcolor());
}
}

void play(Player &player, int board[ROW][COL]) {
    int x, y;
    while (true) {
        if (ismouseclick(WM_LBUTTONDOWN)) {
            getmouseclick(WM_LBUTTONDOWN, x, y);
            int row = y / CELL_SIZE + 1;
            int col = x / CELL_SIZE + 1;
            if (board[row][col] == EMPTY) {
                board[row][col] = player.getSymbol();
                drawSymbol(row, col, player.getSymbol());
                turn++;
                break;
            }
        }
    }
}

};

class Check {
public:
    bool checkWin(int board[][COL], int player) {
        // Kiểm tra hàng ngang
```

```
for (int i = 0; i < COL; i++) {
    for (int j = 0; j < ROW - WIN_LENGTH + 1; j++) {
        bool win = true;
        for (int k = 0; k < WIN_LENGTH; k++) {
            if (board[i][j+k] != player) {
                win = false;
                break;
            }
        }
        if (win) {
            return true;
        }
    }
}
```

// Kiểm tra cột dọc

```
for (int i = 0; i < ROW - WIN_LENGTH + 1; i++) {
    for (int j = 0; j < COL; j++) {
        bool win = true;
        for (int k = 0; k < WIN_LENGTH; k++) {
            if (board[i+k][j] != player) {
                win = false;
                break;
            }
        }
        if (win) {
            return true;
        }
    }
}
```

// Kiểm tra đường chéo chính

```
for (int i = 0; i < ROW - WIN_LENGTH + 1; i++) {
    for (int j = 0; j < COL - WIN_LENGTH + 1; j++) {
        bool win = true;
        for (int k = 0; k < WIN_LENGTH; k++) {
            if (board[i+k][j+k] != player) {
                win = false;
                break;
            }
        }
        if (win) {
            return true;
        }
    }
}
```

```
    }
  }
}

// Kiểm tra đường chéo phụ
for (int i = 0; i < ROW - WIN_LENGTH + 1; i++) {
  for (int j = WIN_LENGTH - 1; j < COL; j++) {
    bool win = true;
    for (int k = 0; k < WIN_LENGTH; k++) {
      if (board[i+k][j-k] != player) {
        win = false;
        break;
      }
    }
    if (win) {
      return true;
    }
  }
}
// Nếu không tìm thấy chiến thắng, trả về false
return false;
}

bool checkDraw() {
  for (int row = 0; row < ROW; row++) {
    for (int col = 0; col < COL; col++) {
      if (board[row][col] == 0) {
        return false;
      }
    }
  }
}
return true;
}

void showWinner(int current_Player) {
  char GAME[] = "GAME WINNER";
  char Player1[] = "Player 1";
  char Player2[] = "Player 2";
  settextstyle(DEFAULT_FONT, HORIZ_DIR, 3);
  setcolor(GREEN);
  outtextxy(getmaxx() / 2 - 100, getmaxy() / 2 - 30, GAME);
  if(current_Player == 1) {
    outtextxy(getmaxx() / 2 - 40, getmaxy() / 2 + 10, Player1);
```

```
    }
    if (current_Player == 2) {
        outtextxy(getmaxx() / 2 - 40, getmaxy() / 2 + 10, Player2);
    }
}
};

int main() {
    Board obj;
    Player p1("Player 1", PLAYER_X);
    Player p2("Player 2", PLAYER_O);
    Check check;
    int currentPlayer = PLAYER_X;
    bool gameOver = false;
    char GD[] = "GAME DRAW";

    initwindow(CELL_SIZE * COL, CELL_SIZE * ROW, "Caro Game");
    obj.draw_board();

    while(!gameOver) {
        p1.play(p1, board);
        if(check.checkWin(board, currentPlayer)) {
            check.showWinner(currentPlayer);
            gameOver = true;
            break;
        } else if(check.checkDraw()) {
            settxtstyle(DEFAULT_FONT, HORIZ_DIR, 3);
            setcolor(GREEN);
            outtextxy(getmaxx() / 2 - 100, getmaxy() / 2 - 30, GD);
            gameOver = true;
            break;
        }
        currentPlayer = (currentPlayer == 1) ? 2 : 1;
        p2.play(p2, board);
        if(check.checkWin(board, currentPlayer)) {
            check.showWinner(currentPlayer);
            gameOver = true;
            break;
        } else if(check.checkDraw()) {
            settxtstyle(DEFAULT_FONT, HORIZ_DIR, 3);
            setcolor(GREEN);
            outtextxy(getmaxx() / 2 - 100, getmaxy() / 2 - 30, GD);
            gameOver = true;
        }
    }
}
```

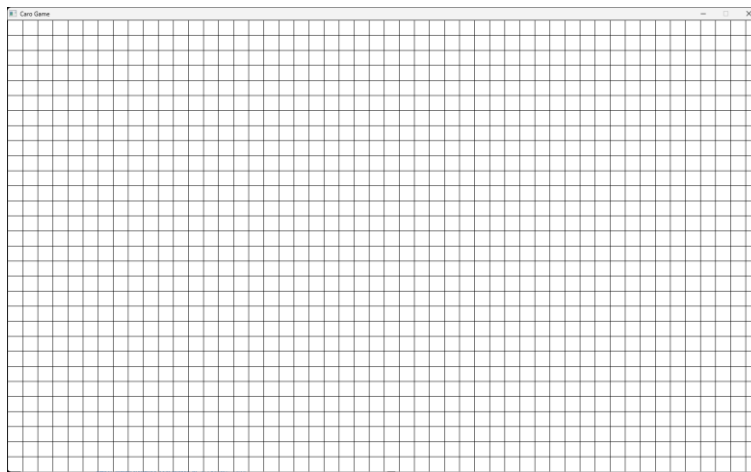
```
        break;
    }
}
getch();

closegraph();

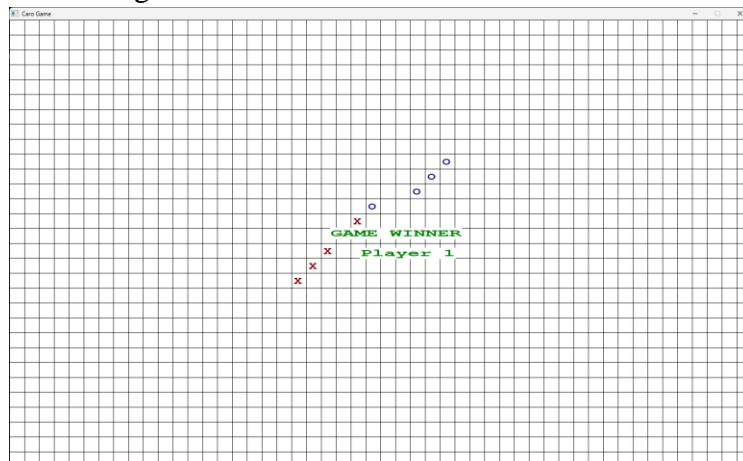
return 0;
}
```

## 2.6 Kết quả chạy chương trình

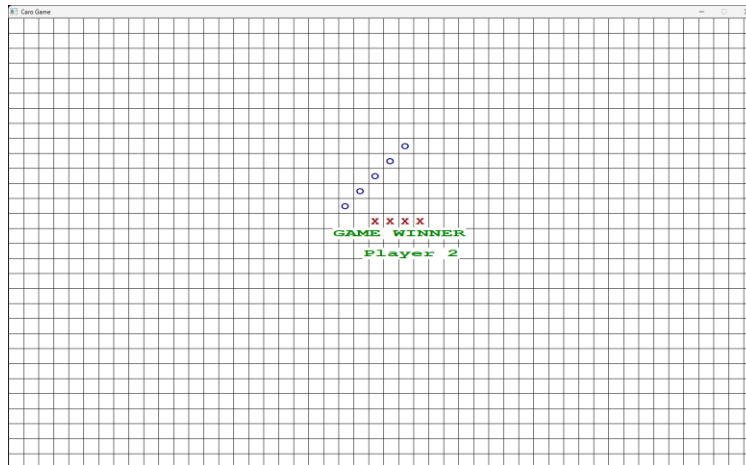
Tạo bàn cờ:



Người chơi 1 thắng



Người chơi 2 thắng



## 2.7 Đánh giá và rút ra nhận xét

Tổng quan thì chương trình đã chạy được như dự tính ban đầu, đã hoàn thành được những yêu cầu cơ bản như hiển thị, kiểm tra, thông báo kết quả lên màn hình. Tuy nhiên vẫn còn một vài vấn đề như chưa áp dụng được hết những gì học được cho quá trình làm chương trình, chưa đạt được những yêu cầu chi tiết hơn cũng như chưa tối ưu được chương trình. Và vẫn gặp một vài lỗi nhỏ trong quá trình chạy chương trình.

Trong tương lai em sẽ cố gắng hoàn thiện chương trình hơn để khắc phục được những lỗi còn hiện hữu, làm giao diện của chương trình hoàn chỉnh và đẹp hơn và đạt được những yêu cầu cao hơn cũng như có thể để đánh với máy.

## 2.8 Tài liệu tham khảo

[https://www.youtube.com/watch?v=h4kUiFOb\\_v0](https://www.youtube.com/watch?v=h4kUiFOb_v0)

<https://www.youtube.com/watch?v=wN0x9eZLix4>

<https://www.youtube.com/watch?v=NTaNsV-DPY>

<https://thaynhuom.edu.vn/ham-do-hoa-co-ban-trong-ngon-ngu-c-c/>

<https://tuicocach.com/click-chuot-trong-man-hinh-do-hoa-ngon-ngu-c-lap-trinh-graphics-trong-c/>

<https://tuicocach.com/ham-do-hoa-co-ban-trong-thu-vien-graphics-h-ngon-ngu-lap-trinh-c-c-graphics-trong-c/>

<https://tuicocach.com/mot-so-chuong-trinh-xay-dung-tren-graphics-c-c/>

[https://www.youtube.com/watch?v=HnFD6D\\_2cGw&t=5s&ab\\_channel=CtyQu%C3%A0t%E1%BA%B7ngNg%C3%A2nTh%C3%A0nhL%C3%A2mT%E1%BA%A5nCh%C3%AD](https://www.youtube.com/watch?v=HnFD6D_2cGw&t=5s&ab_channel=CtyQu%C3%A0t%E1%BA%B7ngNg%C3%A2nTh%C3%A0nhL%C3%A2mT%E1%BA%A5nCh%C3%AD)

[https://www.youtube.com/watch?v=lj7piXFE8N0&t=1719s&ab\\_channel=VoThiHongTuyet](https://www.youtube.com/watch?v=lj7piXFE8N0&t=1719s&ab_channel=VoThiHongTuyet)

<https://www.youtube.com/shorts/TTKUJUbElWU>

Và nhiều tài liệu khác ..