Bài thực hành số 01. KIỂU DỮ LIỆU MẢNG VÀ CON TRỎ

Mục tiêu: Sử dụng được các kiểu dữ liệu cơ bản trong các ngôn ngữ lập trình như: số nguyên, số thực, logic, mảng, con trỏ.

Nhắc lại kiến thức:

Kiểu dữ liệu mảng:

+ Mảng là kiểu dữ liệu có cấu trúc gồm một dãy các ô nhớ liên tiếp nhau, có kích thước bằng nhau.

+ Mỗi mảng có một tên biến để thao tác. Trong C++ tên biến mảng là con trỏ đến ô nhớ đầu tiên của mảng.

+ Khai báo biến mảng:

```
kiểu tên[số phần tử];
```

Ví dụ: khai báo biến mảng m gồm 10 ô nhớ kiểu int

```
int m[10];
```

Thao tác với biến mảng:

+ Thao tác với từng phần tử:

```
tên[vị trí]
```

Vị trí đầu tiên trong mảng của C là 0.

+ Khai báo và gán giá trị cho mảng:

```
int m[] = {1, 2, 3};
```

+ Nhập/xuất mảng: nhập/xuất từng phần tử của mảng.

Nội dung thực hành:

Bài 1. Kiểu dữ liệu cơ bản

Viết chương trình khai báo các biến với kiểu dữ liệu cơ bản, gán giá trị và in lên màn hình.

```
#include <iostream>
using namespace std;
int main () {
  // Creating variables
  int myNum = 5;               // Integer (whole number)
  float myFloatNum = 5.99;     // Floating point number
  double myDoubleNum = 9.98;   // Floating point number
```

```
  char myLetter = 'D';          // Character
  bool myBoolean = true;        // Boolean
  string myString = "Hello";    // String

  // Print variable values
  cout << "int: " << myNum << "\n";
  cout << "float: " << myFloatNum << "\n";
  cout << "double: " << myDoubleNum << "\n";
  cout << "char: " << myLetter << "\n";
  cout << "bool: " << myBoolean << "\n";
  cout << "string: " << myString << "\n";

  return 0;
}
```

Kết quả chạy chương trình:

```
int: 5
float: 5.99
double: 9.98
char: D
bool: 1
string: Hello
```

Bài 2. Kiểu dữ liệu mảng

Viết các hàm nhập, xuất, tìm số lớn nhất của một mảng số nguyên, tạo mảng chứa các số chẵn từ một mảng cho trước. Viết chương trình sử dụng các hàm trên nhập vào một mảng số nguyên, in mảng vừa nhập lên màn hình. Tìm số lớn nhất của mảng vừa nhập và in lên màn hình. Tạo mảng chứa chứa các số chẵn từ mảng vừa nhập và in mảng này lên màn hình.

Chương trình `Arr1.cpp` dưới đây minh họa các thao tác:

+ Nhập một mảng

+ In một mảng

+ Tìm số lớn nhất trong mảng

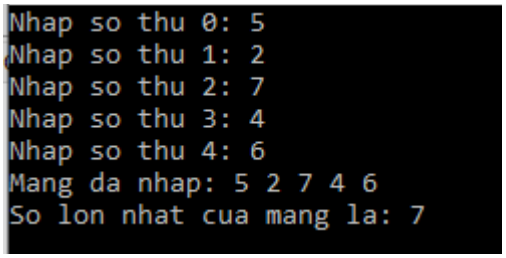
+ Chương trình thực hiện nhập một mảng, in mảng vừa nhập, tìm số lớn nhất của mảng vừa nhập và in lên màn hình.

```
#include <iostream>
```

```
using namespace std;

//Ham nhap mang n so nguyen
void inputArr(int a[], int n)
{
  for(int i = 0; i < n; i++)
  {
    cout << "Nhap so thu " << i << ": " ;
    cin >> a[i];
  }
}

//Ham in mang n so nguyen
void outputArr(int a[], int n)
{
  for(int i = 0; i < n; i++)
    cout << a[i] << " ";
  cout << endl;
}

// Ham tim so lon nhat cua mang a co n so
int maxArr(int a[], int n)
{
  int i, max;
  max = a[0];
  for(i = 1; i < n; i++)
    if(max < a[i]) max = a[i];
  return max;
}

// Ham main
int main()
{
  int x[10], m;
  inputArr(x, 5);
```

```
  cout << "Mang da nhap: ";
  outputArr(x, 5);
  m = maxArr(x, 5);
  cout << "So lon nhat cua mang la: " << m << endl;
  return 0;
}
```

Kết quả chạy chương trình:



Hướng dẫn các thao tác bổ sung: Hàm tạo mảng chứa các số chẵn từ mảng cho trước.

Thuật toán tạo mảng số chẵn từ mảng cho trước

Dữ liệu vào: mảng a có n phần tử
Dữ liệu ra: mảng b chứa các số chẵn của mảng a Thao tác:
Thao tác:
+ Gán biến k = 0
+ Duyệt lần lượt từng phần tử của mảng a
  Nếu phần tử đang xét là số chẵn thì:
    Đưa phần tử đang xét của mảng a vào vị trí k của mảng b
    Tăng k lên 1
+ Trả về mảng b

Cài đặt hàm tạo mảng các số chẵn từ mảng cho trước.

```
int evenArr(int a[], int n, int b[])
{
  int i, k;
  k = 0;
  for(i = 0; i < n; i++)
    if(a[i] % 2 == 0)
    {
      b[k] = a[i];
      k++;
    }
```

```
  return k;
}
```

Bổ sung các lệnh sau trong hàm main

```
  int y[10], n;
  n = evenArr(x, 5, y);
  cout << "Mang cac so chan gom " << n << " so: ";
  outputArr(y, n);
```

Kết quả chạy chương trình:

```
Nhap so thu 0: 5
Nhap so thu 1: 2
Nhap so thu 2: 3
Nhap so thu 3: 8
Nhap so thu 4: 4
Mang da nhap: 5 2 3 8 4
So lon nhat cua mang la: 8
Mang cac so chan gom 3 so: 2 8 4
```

Viết các hàm sau rồi gọi chúng trong hàm main.

a) Tính trung bình cộng của các số trong mảng.

Gợi ý: tính tổng các số trong mảng rồi chia cho số phần tử của mảng.

```
float avgArr(int a[], int n)
{



}
```

b) Tìm một số nguyên có trong mảng hay không?

```
bool findArr(int a[], int n, int x)
{



}
```

c) Sắp xếp mảng theo thứ tự tăng

```
void sortArr(int a[], int n)
{









}
```

d) Ghép hai mảng tăng vào một mảng tăng.

```
int mergeArr(int a[], int n, int b[], int m, int c[])
{









}
```

e) Đếm số lần xuất hiện từng số trong mảng.

Gợi ý: dùng một mảng b để lưu các số khác nhau của mảng a cho trước, mảng d dùng để lưu số lần xuất hiện từng số tương ứng trong mảng b.

Ví dụ: a = {4, 3, 1, 3, 4, 5, 1} thì b = {4, 3, 1, 5} và d = {2, 2, 2, 1}.

Hàm `countArr()` trả về số phần tử của mảng b.

```
int countArr(int a[], int n, int b[], int d[])
{




```

```
}
```

Câu hỏi:

a) *Làm thế nào để biết được số ô nhớ của một biến mảng?*
b) *Có thể thay đổi kích thước mảng trong chương trình không?*
c) *Có thể duyệt biến mảng không qua chỉ số không?*

Bài 3. Kiểu con trỏ

Thực hành một số thao tác trên biến kiểu con trỏ.

Ví dụ: Chương trình dưới đây minh họa các thao tác trên biến con trỏ gồm: khai báo, gán địa chỉ, gán giá trị cho biến con trỏ.

```
#include <iostream>

int main() {
  // Declare an integer variable and a pointer to it
  int x = 10;
  int* p = &x;

  // Print the value of x and the address of x
  std::cout << "x = " << x << ", &x = " << &x << std::endl;

  // Print the value of p and the value pointed to by p
  std::cout << "p = " << p << ", *p = " << *p << std::endl;

  // Modify the value of x through the pointer
  *p = 20;

  // Print the new value of x
  std::cout << "x = " << x << std::endl;

  return 0;
}
```

Kết quả chương trình:

```
x = 10, &x = 0x6ffe04
p = 0x6ffe04, *p = 10
x = 20
```

Câu hỏi: Hãy thêm một biến con trỏ q kiểu int, sau đó gán cho nó giá trị của biến con trỏ p. Sửa dữ liệu của ô nhớ do q đang trỏ đến rồi in lại biến x xem giá trị có thay đổi không?

Ví dụ: Chương trình Pointer1.cpp minh họa các thao tác cập phát và thu hồi ô nhớ dùng biến con trỏ.

```
#include <iostream>

int main() {
  // Dynamically allocate an integer and a pointer to it
  int* p = new int;
  *p = 10;

  // Print the value of p and the value pointed to by p
  std::cout << "p = " << p << ", *p = " << *p << std::endl;

  // Deallocate the memory pointed to by p
  delete p;
  // Set p to null to prevent it from being used again
  p = nullptr;

  return 0;
}
```

Kết quả chương trình:

```
p = 0x191510, *p = 10
```

Câu hỏi:

a) *Sau khi thu hồi ô nhớ p, nếu in ra giá trị p và *p thì kết quả thế nào? Giải thích lý do có kết quả như vậy.*
b) *Sau khi thu hồi ô nhớ p rồi cấp phát lại thì địa chỉ cấp phát cho ô nhớ p có giống như cấp phát lần đầu không? Hãy viết các lệnh để minh họa cho điều này. Giải thích lý do tại sao như vậy.*

Ví dụ: chương trình Pointer2.cpp minh họa biến con trỏ của con trỏ.

```
#include <iostream>
int main() {
```

```
  // Declare an integer variable and a pointer to it
  int x = 10;
  int* p = &x;

  // Declare a pointer to the pointer p
  int** pp = &p;

  // Print the value of x, the address of x,
  // the value of p, and the value pointed to by p
  std::cout << "x = " << x << ", &x = " << &x << std::endl;
  std::cout << "p = " << p << ", *p = " << *p << std::endl;

  // Modify the value of x through the pointer to the pointer
  **pp = 20;

  // Print the new value of x
  std::cout << "x = " << x << std::endl;

  return 0;
}
```

Kết quả chương trình:

```
x = 10, &x = 0x6ffe04
p = 0x6ffe04, *p = 10
x = 20
```

Câu hỏi: *Hãy giải thích tại sao giá trị cuối của x là 20.*

Con trỏ và mảng: trong C++ biến mảng là biến con trỏ và giá trị của nó là địa chỉ của ô nhớ đầu tiên trong mảng.

Ví dụ: nếu khai báo mảng int a[10]; thì biến a là biến con trỏ lưu địa chỉ ô nhớ a[0].

Chương trình Point3.cpp minh họa việc sử dụng biến con trỏ thao tác với mảng trong C++:

```
#include <iostream>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int* ptr = arr;
```

```
    std::cout << "Array elements using pointer: ";
    for (int i = 0; i < 5; i++) {
        std::cout << *ptr << " ";
        ptr++;
    }

    std::cout << std::endl;
    std::cout << "Array elements using array subscript notation: ";
    for (int i = 0; i < 5; i++) {
        std::cout << arr[i] << " ";
    }

    return 0;
}
```

Kết quả chạy chương trình:

```
Original array: 1 2 3 4 5
Array after doubling the values: 2 4 6 8 10
```

Chương trình Pointer4.cpp minh họa việc sử dụng con trỏ thay cho biến mảng.

```
#include <iostream>

int main() {
    const int SIZE = 5;
    int* arr = new int[SIZE];
    int* ptr = arr;

    // Initialize array
    for (int i = 0; i < SIZE; i++) {
        *ptr = i + 1;
        ptr++;
    }

    ptr = arr;  // Reset pointer to the beginning of the array
```

```
    std::cout << "Original array: ";
    for (int i = 0; i < SIZE; i++) {
        std::cout << *ptr << " ";
        ptr++;
    }

    ptr = arr; // Reset pointer to the beginning of the array
    for (int i = 0; i < SIZE; i++) {
        *ptr = *ptr * 2;
        ptr++;
    }

    ptr = arr;  // Reset pointer to the beginning of the array
    std::cout << std::endl;
    std::cout << "Array after doubling the values: ";
    for (int i = 0; i < SIZE; i++) {
        std::cout << *ptr << " ";
        ptr++;
    }

    delete[] arr;
    return 0;
}
```

Kết quả chạy chương trình:

```
Original array: 1 2 3 4 5
Array after doubling the values: 2 4 6 8 10
```

Chương trình sử dụng toán tử `new` để cấp phát ô nhớ cho biến con trỏ và `delete` để thu hồi các ô nhớ của biến con trỏ.

Bài 4. Mảng động

Chương trình DynamicArray1.cpp minh họa thao tác trên mảng động.

```
#include <iostream>
```

```
int main() {
    int size;
    std::cout << "Enter the size of the array: ";
    std::cin >> size;

    int* arr = new int[size];
    int* ptr = arr;

    // Initialize array
    for (int i = 0; i < size; i++) {
        *ptr = i + 1;
        ptr++;
    }

    ptr = arr;  // Reset pointer to the beginning of the array
    std::cout << "Original array: ";
    for (int i = 0; i < size; i++) {
        std::cout << *ptr << " ";
        ptr++;
    }

    //Resize the array
    int new_size;
    std::cout << "\nEnter the new size of the array: ";
    std::cin >> new_size;
    int* new_arr = new int[new_size];

    // copy the elements to the new array
    for (int i = 0; i < std::min(size, new_size); i++) {
        new_arr[i] = arr[i];
    }

    ptr = new_arr;  // Update pointer to point to the new array
    std::cout << "Array after resizing: ";
```

```
    for (int i = 0; i < new_size; i++) {
        std::cout << *ptr << " ";
        ptr++;
    }

    // Free the memory
    delete[] arr;
    delete[] new_arr;
    return 0;
}
```

Giải thích: ban đầu biến con trỏ arr được cấp phát size ô nhớ và gán các giá trị từ 1 đến size. Sau đó một con trỏ new_arr được cấp phát new_size ô nhớ và copy các giá trị từ arr sang new_arr. Như vậy ta có thể dùng biến con trỏ để tạo các mảng mà kích thước có thể thay đổi trong quá trình sử dụng.

Hãy dùng mảng động viết chương trình DynamicArray2.cpp thực hiện các yêu cầu sau:

a) Nhập một mảng động n số nguyên.
b) In mảng động n số nguyên.
c) Thêm một số vào cuối mảng động

Viết các hàm:

a) Hàm nhập mảng động n số nguyên. Trả về con trỏ đến phần tử đầu tiên của mảng.

```
int* inputDynamicArr(int n)
{




}
```

b) Hàm in mảng động n số nguyên có địa chỉ từ arr.

```
void printDynamicArr(int *arr, int n)
{



}
```

c) Hàm thêm một số x vào cuối mảng arr có n số.

```
void appendDynamicArr(int *arr, int &n, int x)
```

```
{




}
```

Viết hàm main() sử dụng các hàm trên.

```
int main()
{



}
```

Bài 5 Mảng hai chiều

Chương trình Arr2D_1.cpp minh họa thao tác đơn giản với mảng hai chiều.

```
#include <iostream>

int main() {
    // Declare a 2D array with 3 rows and 4 columns
    int myArray[3][4];

    // Initialize the elements of the array
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            myArray[i][j] = i + j;
        }
    }

    // Print the elements of the array
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            std::cout << myArray[i][j] << " ";
        }
```

```
        std::cout << std::endl;
    }

    return 0;
}
```

Chương trình Arr2D_2.cpp minh họa các chức năng nhập, xuất mảng hai chiều và cộng hai ma trận được lưu trong hai mảng hai chiều.

```
#include <iostream>

const int ROWS = 3;
const int COLS = 4;

void inputArray(int arr[][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            std::cout << "Enter element at position [" << i << "][" <<
j << "]: ";
            std::cin >> arr[i][j];
        }
    }
}

void outputArray(int arr[][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            std::cout << arr[i][j] << " ";
        }
        std::cout << std::endl;
    }
}

void addArray(int arr1[][COLS], int arr2[][COLS], int arr3[][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            arr3[i][j] = arr1[i][j] + arr2[i][j];
```

```
        }
    }
}

int main() {
    // Declare three 2D arrays
    int arr1[ROWS][COLS], arr2[ROWS][COLS], arr3[ROWS][COLS];

    // Input elements for the first array
    std::cout << "Enter elements for the first array: " << std::endl;
    inputArray(arr1);

    // Input elements for the second array
    std::cout << "Enter elements for the second array: " << std::endl;
    inputArray(arr2);

    // Add the elements of the second array to the first array
    addArray(arr1, arr2, arr3);

    // Output the elements of the third array (which now contains the
sum of the elements)
    std::cout << "Elements of the third array: " << std::endl;
    outputArray(arr3);

    return 0;
}
```

Viết chương trình Arr2D_3.cpp với các chức năng:

a) Nhập một mảng 2 chiều với số dòng m, số cột n.

```
void inputArr2D(int arr[][MAXCOLS], int m, int n)
{




}
```

b) In một mảng 2 chiều với số dòng m, số cột n.

```
void printArr2D(int arr[][MAXCOLS], int m, int n)
{



}
```

c) Cộng hai mảng hai chiều a, b có m dòng, n cột lưu vào mảng c.

```
void addArr2D(int a[][MAXCOLS], int b[][MAXCOLS], int m, int n, int
c[][MAXCOLS])
{




}
```

d) Tìm vị trí số lớn nhất trong mảng hai chiều.

```
int maxArr2D(int arr[][MAXCOLS], int m, int n, int &rowMax, int &colMax)
{




}
```

e) Nhân hai ma trận (mảng 2 chiều) a có m dòng p cột và b có p dòng n cột lưu vào ma trận c có m dòng, n cột.

```
void multiplyArr2D(int a[][MAXCOLS], int b[][MAXCOLS], int m, int p, int
n, int c[][MAXCOLS])
{



```

```
}
```

Hàm main sử dụng các hàm trên.

```
int main()
{



}
```
