Bài 2.12: Lớp template array

- ✓ Cú pháp khai báo mảng
- ✓ Khởi tạo mảng
- ✓ Duyệt mảng
- ✓ Truyền mảng vào hàm
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Khai báo mảng

- Lớp nguyên mẫu array là một lớp được xây dựng sẵn hỗ trợ các thao tác với các phần tử trong mảng.
- Để sử dụng lớp array ta nạp thư viện array vào qua cú pháp:
  #include <array>
- Cú pháp khai báo biến mảng với lớp array:
  array<type, arraySize> arrayName;
- Trong đó:
  - **array** là tên lớp mặc định, giữ nguyên.
  - Sau đó là cặp ngoặc <> chứa hai thành phần bên trong.
  - **type** là kiểu của mảng, có thể là bất kì kiểu dữ liệu hợp lệ nào trong C++.
  - **arraySize** là kích thước tối đa của mảng, luôn là số nguyên dương.
  - **arrayName** là tên mảng, đặt sao cho thể hiện được tên tập hợp.
  - Kết thúc khai báo có dấu chấm phẩy **;**

Ví dụ

- Khai báo mảng numbers kiểu int, chứa tối đa 10 phần tử:

```
const size_t SIZE = 10;
// khai báo mảng numbers với 10 phần tử tối đa
array<int, SIZE> numbers;
// gán các phần tử cho mảng:
for (size_t i = 0; i < SIZE; i++)
{
      numbers[i] = i * (i + 1);
}
```

Khởi tạo mảng

- Cú pháp: array<type, size> arrayName = { elements };
- Trong đó:
  - Elements là danh sách phần tử cần khởi tạo cho mảng, chúng được bao bởi cặp ngoặc {}
  - Nếu số phần tử trong danh sách khởi tạo nhiều hơn 1, các phần tử sẽ ngăn cách nhau bằng dấu phẩy.
  - Số lượng phần tử khởi tạo cho mảng phải nhỏ hơn hoặc bằng số phần tử tối đa của mảng nếu không chương trình sẽ bị lỗi.
  - Nếu khởi tạo thiếu, các phần tử còn lại sẽ tự động được gán giá trị mặc định của kiểu.
  - Kết thúc khởi tạo là dấu chấm phẩy.

```
const size_t SIZE = 10;
// khởi tạo đầy đủ:
array<int, SIZE> ages =
{ 2, 5, 6, 4, 8, 3, 78, 99, 10, 21 };
// khởi tạo không đầy đủ:
array<int, SIZE> numbers = {50, 12, 37, 28};
// khởi tạo thiếu hoàn toàn
array<float, SIZE> grades = {};
```

Duyệt mảng

- Để duyệt mảng, sử dụng vòng for thường hoặc for rút gọn.
- Cú pháp gán, truy xuất phần tử mảng thực hiện như mảng thông thường.
- Luôn lưu ý rằng chỉ số mảng nằm trong đoạn [0, n-1].
- Sử dụng vòng for rút gọn cho phép ta duyệt từ đầu đến cuối mảng và không cần quan tâm chỉ số mảng.

```
for(declare : expression) {
    // statement
}
```

Duyệt mảng

```
array<int, SIZE> ages =
{ 2, 5, 6, 4, 8, 3, 78, 99, 10, 21 };

// hiển thị giá trị các phần tử mảng ages:
cout << "Mang goc ban dau: " << endl;
for (auto item : ages)
{
      cout << item << " ";
}

// update các phần tử của mảng:
for (auto& item : ages) {
      item += 25; // cộng thêm cho mỗi phần tử giá trị 25
}
```

Truyền mảng vào hàm

- Ta sử dụng hàm template để truyền mảng template vào hàm:

```
template<class T> void showArrayElements(array<T, SIZE> mArray) {
   for (auto item : mArray)
   {
         cout << item << " ";
   }
   cout << endl;
}

template<class T> void updateArrayElements(array<T, SIZE> mArray)
{
   for (auto& item : mArray) {
         item += 25; // cộng thêm cho mỗi phần tử giá trị 25
   }
}

int main() {
   // khởi tạo đầy đủ:
   array<int, SIZE> ages =
   { 2, 5, 6, 4, 8, 3, 78, 99, 10, 21 };

   // hiển thị giá trị các phần tử mảng ages:
   cout << "Mang goc ban dau: " << endl;
   showArrayElements(ages);
```

Sắp xếp mảng

- Ta sử dụng hàm sort của thư viện algorithm để sắp xếp các phần tử mảng.

```
array<int, SIZE> ages =
{ 2, 5, 6, 4, 8, 3, 78, 99, 10, 21 };

// mảng chứa tên các màu kiểu string
array<string, SIZE> colors =
{ "red", "blue", "green", "white", "pink",
  "purple",     "gray", "silver", "orange", "magenta"
};
// hiển thị giá trị các phần tử mảng ages:
cout << "Mang goc ban dau: " << endl;
showArrayElements(ages);
showArrayElements(colors);
// sắp xếp phần tử mảng theo thứ tự tăng dần:
sort(ages.begin(), ages.end());
sort(colors.begin(), colors.end());
```

Sắp xếp mảng

- Để sắp xếp mảng theo thứ tự giảm dần, ta sử dụng đối số thứ 3 cho hàm là greater():

```
// sắp xếp phần tử mảng theo thứ tự giảm dần:
sort(ages.begin(), ages.end(), greater());
sort(colors.begin(), colors.end(), greater());
```

```
Mang goc ban dau:
2 5 6 4 8 3 78 99 10 21
red blue green white pink purple gray silver orange magenta

Mang sau khi sap xep:
99 78 21 10 8 6 5 4 3 2
white silver red purple pink orange magenta green gray blue
```

Tìm kiếm trên mảng

- Ta sử dụng hàm binary_search() của thư viện algorithm để tìm kiếm phần tử trong mảng.
- Lưu ý rằng mảng phải được sắp xếp theo thứ tự tăng dần của các phần tử trước khi tìm kiếm.
- Kết quả trả về bởi hàm binary_search() là một giá trị bool cho giá trị true nếu tìm thấy và false trong trường hợp ngược lại.
- Ví dụ sau tìm kiếm giá trị “red” và “yellow” trong mảng colors:

```
bool isFoundRed =
     binary_search(colors.begin(), colors.end(), "red");
bool isFoundYellow =
     binary_search(colors.begin(), colors.end(), "yellow");

cout << endl << "Search result: " << endl;
cout << "red color " << (isFoundRed ? "was" : "was not")
     << " found in colors array" << endl;
cout << "yellow color " << (isFoundYellow ? "was" : "was not")
     << " found in colors array" << endl;
```

Nội dung tiếp theo

Giới thiệu vector