南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Advanced Programming

Lab 03

CONTENTS

- ✓ Objectives
- ✓ Knowledge points
- ✓ Exercises

1 Objectives

- Master the use of arrays
- Master character arrays and strings
- Master the use of Structure
- Learn about Union

2 Knowledge Points

2.1 Array

2.2 Character arrays and strings

2.3 Structure

2.4 Union

2.1 Array

- Arrays are fixed-size collections consisting of data items of the same type.
- The array name represents the first address of the contiguous storage space.
- The index of an array is from 0.
- C/C++ compiler does not report whether the array is out-off-bounds.

One-dimensional Array Sample 1

```
#include <iostream>
using namespace std;

int main()
{
    int foo[] = {16,2,77,40,12071};
    int a = 1;

    foo[0] = a;
    foo[1] = -34;
    a = foo[2];

    cout << "foo[0] = " << foo[0] <<endl;
    cout << "foo[1] = " << foo[1] <<endl;
    cout << "foo[2] = " << foo[2] <<endl;
    cout << "a = " << a <<endl;

    cout << "The size of foo is: " << sizeof(foo) << " bytes." << endl;
    cout << "The size of an element of foo is: " << sizeof(foo[0]) << " bytes." << endl;
    cout << "There are " <<sizeof(foo)/sizeof(foo[0]) << " element in foo." << endl;

    return 0;
}
```

Define and initialize a one-dimension array

Use [ ] operator to access the elements of the array

Index from 0.

Get the amount of bytes of an array.

Count the number of elements in an array

Result:

```
foo[0] = 1
foo[1] = -34
foo[2] = 77
a = 77
The size of foo is: 20 bytes.
The size of an element of foo is: 4 bytes.
There are 5 element in foo.
```

One-dimensional Array Sample 2

```cpp
#include <iostream>
#define SIZE 2    //Use preprocessor  #define to define a symbolic constant
const int arrSize = 3;  //Use C++ style to define a constant

int main()
{
    using namespace std;
    //Initialize an array partially, the remaining elements are 0.
    //The number of elements must be specified.
    int a[SIZE] = {0};
    double b[arrSize] = {1};

    cout << "The elements in a are: "<< a[0] << " , " << a[1] << endl;
    cout << "The elements in b are: "<< b[0] << " , " << b[1]  << " , " << b[2] << endl;

    return 0;
}
```

Result:

```
The elements in a are: 0 , 0
The elements in b are: 1 , 0 , 0
```

Two-dimensional array

int a[3][4]

Two-dimensional array a is consists of 3 elements

a[0]	a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1]	a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2]	a[2][0]	a[2][1]	a[2][2]	a[3][3]

row name

Each element a[i] is consists of one-dimensional array containing 4 elements

0	a[0][0]	a[0]
1	a[0][1]	
2	a[0][2]	
3	a[0][3]	
4	a[1][0]	a[1]
5	a[1][1]	
6	a[1][2]	
7	a[1][3]	
8	a[2][0]	a[2]
9	a[2][1]	
10	a[2][2]	
11	a[2][3]	

Two-dimensional array Sample

```cpp
#include <iostream>
using namespace std;

int main()
{
    //Define and initialize a two-dimension array
    int test[3][2] =
    {
        {2,-5},
        {4,0},
        {9,1}
    };
    //Use [ ] [ ]operator to access the elements of the array
    cout << "test[0][1] = "<< test[0][1] << endl;
    cout << "test[2][0] = "<< test[2][0] << endl;

    cout << "The size of test is: " << sizeof(test) << endl;
    cout << "The size of the first row in test is: " << sizeof(test[0]) << endl;
    cout << "The size of one element in test is: " << sizeof(test[0][0]) << endl;

    return 0;
}
```

Result:

```
test[0][1] = -5
test[2][0] = 9
The size of test is: 24
The size of the first row in test is: 8
The size of one element in test is: 4
```

2.2 Character array and strings

2.2.1 Define a C-style string

You can use one of the four ways below to define a character array:

char str[ ] = “C++” ; **\\Strings end with ‘\0’**

char str[4] = “C++” ; **\\Strings end with ‘\0’**

char str[ ] = {‘C’, ‘+’, ‘+’, ‘\0’};

char str[4] = {‘C’, ‘+’, ‘+’, ‘\0’}

```
char str1[] = "C++";
char str2[] = {'C','+','+'};
//char str2[] = {'C','+','+','\0'};

cout << "The sizeof str1 is "<< sizeof(str1) << ",the length of str1 is "<< strlen(str1) <<endl;
cout << "The sizeof str2 is "<< sizeof(str2) << ",the length of str2 is "<< strlen(str2) <<endl;
```

Result:

```
The sizeof str1 is 4,the length of str1 is 3
The sizeof str2 is 3,the length of str2 is 6
```

String Sample

string is a class of C++, it can be used as a type.

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
    string str1 = "C";
    string str2 = "C and C++";
    string str3 = str2 + "programming is very interesting.";

    cout << "The sizeof str1 is "<< sizeof(str1) <<  ",the length of str1 is "<< str1.length() <<endl;
    cout << "The sizeof str2 is "<< sizeof(str2) << ",the length of str2 is "<< str2.length() << endl;
    cout << "The sizeof str3 is "<< sizeof(str3) << ",the length of str3 is "<< str3.size() << endl;

    return 0;
}
```

Result:

```
The sizeof str1 is 32,the length of str1 is 1
The sizeof str3 is 32,the length of str3 is 9
The sizeof str4 is 32,the length of str4 is 41
```

2.2.2 Keyboard input and terminal output

```c
#include <stdio.h>

int main()
{
    char str[20];
    printf("Enter a string:\n");
    scanf("%s",str);
    printf("You entered: %s\n",str);

    return 0;
}
```

1. C: scanf & printf

   %d ----int

   %f ----float

   %c -----char

   %s -----string

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week03$ ./a.out
Enter a string:CS219
You entered: CS219
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week03$ ./a.out
Enter a string:hello CS219
You entered: hello
```

Why only hello?

scanf uses whitespace—spaces, tabs, and newlines to delineate a string.

2. C: gets/fgets

```c
#include <stdio.h>

int main()
{
    char str[20];
    printf("Enter a string:");
    // gets(str);
    fgets(str, sizeof(str), stdin);
    printf("You entered:%s\n",str);

    return 0;
}
```

When you use the gets(str) , a warning appears, because it doesn't check the length of the input, so if inputs more characters than the array can hold, it might cause an overflow, which can lead to security issues.

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week03$ ./a.out
Enter a string:hello CS219
You entered:hello CS219
```

3. C++: cin & cout

```cpp
#include <iostream>
using namespace std;

int main()
{
    char str[100];

    cout << "Enter a string: ";
    cin >> str;
    cout << "You entered: " << str << endl;

    cout << "Enter another string: ";
    cin >> str;
    cout << "You entered: " << str << endl;

    return 0;
}
```

Same question

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week03$ ./a.out
Enter a string: Advanced Programming
You entered: Advanced
Enter another string: You entered: Programming
```

The cin is to use whitespace-- spaces, tabs, and newlines to delineate a string.

4. C++: cin.get( )

```cpp
#include <iostream>
using namespace std;

int main()
{
    char str[20];

    cout << "Enter a string: ";
    cin.get(str,sizeof(str));
    cout << "You entered: " << str << endl;
    cin.get();

    cout << "Enter another string: ";
    cin.get(str,sizeof(str));
    cout << "You entered: " << str << endl;

    return 0;
}
```

Input a single character:
istream& get(char&);
int get(void);

Input a string:
istream& get(char*,int);

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_202
Enter a string: hello CS219
You entered: hello CS219
Enter another string: I like Advanced Programming.
You entered: I like Advanced Pro
```

If the length of input string is greater than 20, it can only store first 19 characters in it.

5. C++: cin.getline( )

Input a string: **istream& getline(char*,int);**

```
#include <iostream>
using namespace std;

int main()
{
    char str[20];

    cout << "Enter a string: ";
    cin.getline(str,sizeof(str));
    cout << "You entered: " << str << endl;

    cout << "Enter another string: ";
    cin.getline(str,sizeof(str));
    cout << "You entered: " << str << endl;

    return 0;
}
```

What's the difference between get() and getline()?

If the length of input string is greater than 20, it can only store first 19 characters in it.

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week03$ ./a.out
Enter a string: hello CS219
You entered: hello CS219
Enter another string: I like Advanced Programming.
You entered: I like Advanced Pro
```

cin.get( ) vs cin.getline( )

getline() and get() both read an entire input line—that is, up until a newline character. However, getline() discard the newline character, whereas get() leave it in the input queue.

6. string class I/O

getline() function takes the input stream as the first parameter which is cin and str as the location of the line to be stored.

```
#include <iostream>
using namespace std;

int main()
{
    string str;

    cout << "Enter a string: ";
    getline(cin,str);
    cout << "You entered: " << str << endl;

    cout << "Enter another string: ";
    getline(cin,str);
    cout << "You entered: " << str << endl;

    return 0;
}
```

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week03$ ./a.out
Enter a string: hello CS219
You entered: hello CS219
Enter another string: I like Advanced Programming.
You entered: I like Advanced Programming.
```

2.3 Structure

A **structure** is a user defined data type available in C that allows to combine **data items of different kinds** under a single name. **When a structure is declared, no memory is allocated.**

```
struct Employee
{
    int id;
    string name;
};
struct Person
{
    char name[7];
    int age;
    double salary;
};

int main()
{
     Employee emp;
    Person per;
     cout << "The sizeof Employee is " << sizeof(emp) <<endl;
    cout << "The sizeof Person is " << sizeof(Person) <<endl;
    return 0;
}
```

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week03$ ./a.out
The sizeof Employee is 40
The sizeof Person is 24
```

Normally, the size of a structure is not the sum of the sizes of its members because of memory alignment.

2.4 Union

A **union** is a special data type available in C that allows to **store different data types in the same memory location.** You can define a union with many members, but **only one member can contain a value at any given time**. The memory occupied by a union will be large enough to hold the largest member of the union. Unions provide an efficient way of using the same memory location for multiple-purpose.

```
#include <iostream>
#include <cstring>
using namespace std;

union Data
{
    int i;
    double d;
    char charr[20];
};

int main()
{
    Data data;
    data.i=4;
    data.d=1.4;
    strcpy(data.charr, "CS219");

    cout << "The size of Data is:" << sizeof(data)
<< endl;
    cout << "data.i is: " << data.i  << endl;
    cout << "data.d is: " << data.d  << endl;
    cout << "data.charr is: " << data.charr  << endl;

}
```

A union is big enough to hold the " widest" member, and the alignment is appropriate for all of the types in the union.

Result:

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/
The size of Data is: 24 bytes.
data.i is: 825381699
data.d is: 1.37505
data.charr is: CS219
```

Only one value is valid.

3 Exercise

Exercise 1

Complete the code, explain the result to SA.If there are bugs, please fix them.

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
    int cards[4]{};
    int price[] = {2.8,3.7,5,9,-1};
    char direction[4] {'L',82,'U',68};
    char title[] = "ChartGPT is an awesome tool.";
    cout << "sizeof(cards) = " << sizeof(cards) << ",sizeof of cards[0] = " << sizeof(cards[0]) << endl;
    cout << "sizeof(price) = " << sizeof(price) << ",sizeof of price[0] = " << sizeof(price[1]) << endl;
    cout << "sizeof(direction) = " << sizeof(direction) << ",length of direction = " << strlen(direction)
<< endl;
    cout << "sizeof(title) = " << sizeof(title) << ",length of title = " << strlen(title) << endl;

  //Print the address of each array variable.
  ......
    return 0;
}
```

Exercise 2

Run the program and explain the result to SA.

```c
#include <stdio.h>
union data
{
    int n;
    char ch;
    short m;
};
int main()
{
    union data a;
    printf("%lu, %lu\n", sizeof(a), sizeof(union data) );
    a.n = 0x40;
    printf("%X, %c, %hX\n", a.n, a.ch, a.m);
    a.ch = '9';
    printf("%X, %c, %hX\n", a.n, a.ch, a.m);
    a.m = 0x2059;
    printf("%X, %c, %hX\n", a.n, a.ch, a.m);
    a.n = 0x3E25AD54;
    printf("%X, %c, %hX\n", a.n, a.ch, a.m);

    return 0;
}
```

Exercise 3

The **CandyBar** structure contains three members: name(character array), weight(float) and the number of calories(integer). Write a program that creates an array of **three CandyBar** structures, initializes them to value of your input, and then displays the contents of each structure. Find the greatest calories per weight, display the name and calories per weight of which satisfies the condition.

Sample structure:

```
struct CandyBar
{
    char name[20];
    float weight;
    int calories;
};
```

Sample output:

```
Enter brand name of a Candybar: Ferro Rocher
Enter weight of the Candybar: 23.6
Enter calories (an integer value) in the Candybar: 893
Enter brand name of a Candybar: Hershey's
Enter weight of the Candybar: 13.2
Enter calories (an integer value) in the Candybar: 658
Enter brand name of a Candybar: Mars Wrigley
Enter weight of the Candybar: 3.2
Enter calories (an integer value) in the Candybar: 127

Display the CandyBar array contents
Brand name: Ferro Rocher
Weight: 23.6
Calories: 893
Brand name: Hershey's
Weight: 13.2
Calories: 658
Brand name: Mars Wrigley
Weight: 3.2
Calories: 127
The greatest calories per weight is:
Brand name: Hershey's
Calories per weight: 49.8485
```