C/C++ Program Design

CS205

Prof. Shiqi Yu (于仕琪)

yusq@sustech.edu.cn

http://faculty.sustech.edu.cn/yusq/
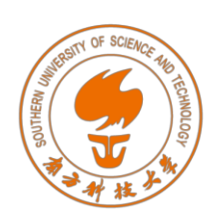
Arrays

Arrays

- A contiguously allocated memory

- Fixed number of objects (The array size cannot be changed)

- Its element type can be any fundamental type (int, float, bool, etc), structure, class, pointer, enumeration,

array.cpp

```cpp
int num_array1[5]; //uninitialized array, random values
int num_array2[5] = {0, 1, 2, 3, 4}; //initialization
```

```
yushiqi: examples $ g++ array.cpp
yushiqi: examples $ ./a.out
0 0 0 0 169944840
0 1 2 3 4
yushiqi: examples $ ./a.out
0 0 0 0 163256072
0 1 2 3 4
yushiqi: examples $ ./a.out
0 0 0 0 142948104
0 1 2 3 4
```

Variable-length arrays

- If the length is not an integer constant expression, the array will be a variable-length one.

```cpp
int len = 1;
while ( len < 10 )
{
    int num_array2[len]; //variable-length array
    cout << "len = " << len;
    cout << ", sizeof(num_array2)) = "
        << sizeof(num_array2) << endl;
    len ++;
}
```

变长数组

Arrays of unknown size

- The number is not specified in the declaration.

    int num_array[ ] = {1, 2, 3, 4}; // the type of num_array is "array of 4 int"

    未说明长度

- The arguments of a function

    float array_sum(float values[], size_t length);

    float array_sum(float *values, size_t length);

Element accessing

```cpp
int array1[4] = {9,8,7,6};
int array2[4];
array2 = array1; //error!
array2[0] = array1[0]; //okay
array2[1] = array1[1]; //okay
array2[2] = array1[2]; //okay
array2[3] = array1[3]; //okay
```

- No bounds-checking in C/C++.

index-bound.cpp

```cpp
int num_array[5];

for(int idx = -1; idx <= 5; idx++)
    num_array[idx] = idx * idx;

for(int idx = -1; idx <= 5; idx++)
    cout << num_array[idx] << endl;
```

Index	Value	Address
		p+19
		p+18
		p+17
		p+16
		p+15
		p+14
3	6	p+13
		p+12
		p+11
2	7	p+10
		p+9
		p+8
		p+7
		p+6
1	8	p+5
		p+4
		p+3
0	9	p+2
		p+1
		p+0
		p−1
		p−2
		p−3
		p−4

不检查越界

- **Arrays are not objects in C/C++ (different with Java);**
- **Arrays can be regarded as addresses**

Multidimensional arrays

```cpp
int mat[2][3] = {{11,12,13}, {14,15,16}};

for (int r = 0; r < rows; r++)
{
    for(int c = 0; c < cols; c++)
        cout << mat[r][c] << ",";
    cout << endl;
}
```

md-array.cpp

Arrays of unknown bound

```cpp
void init_2d_array(float mat[][], //error
        size_t rows, size_t cols)



void init_2d_array(float mat[][3],
        size_t rows, size_t cols)
```

必须注明

Index	Value	Address
		p+25
		p+24
		p+23
[1][2]	16	p+22
		p+21
		p+20
		p+19
[1][1]	15	p+18
		p+17
		p+16
		p+15
[1][0]	14	p+14
		p+13
		p+12
		p+11
[0][2]	13	p+10
		p+9
		p+8
		p+7
[0][1]	12	p+6
		p+5
		p+4
		p+3
[0][0]	11	p+2
		p+1
		p+0
		p-1
		p-2

`const` Arrays

const float PI = 3.1415926f;

PI += 1.f; // error

const float values[4] = {1.1f, 2.2f, 3.3f, 4.4f};
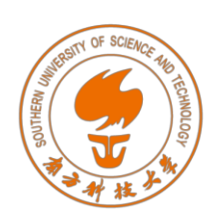
values[0] = 1.0f; // error

const 防止误操作

Used as function arguments

```cpp
float array_sum(const float values[], size_t length)
{
    float sum = 0.0f;
    for (int i = 0; i < length; i++)
    {
        sum += values[i];
        //values[i] = 0; //error
    }
    return sum;
}
```

const-array.cpp

```cpp
int main()
{
    float values[4] = {1.1f, 2.2f, 3.3f, 4.4f};
    float sum = array_sum(values, 4);
}
```

Strings

Array-Style Strings and `string` class

Array-style strings

- An array-style string (null-terminated strings/arrays of characters) is a series of characters stored in bytes in memory.

- This kind of strings can be declared as follows

```cpp
char rabbit[16] = {'P', 'e', 't', 'e', 'r'};

char bad_pig[9] = {'P', 'e', 'p', 'p', 'a', ' ', 'P', 'i', 'g'}; //a bad one!

char good_pig[10] = {'P', 'e', 'p', 'p', 'a', ' ', 'P', 'i', 'g', '\0'};
```

- size_t strlen(const char *str);

  Returns the number of characters, the first NULL will not be included.

```cpp
char name[10] = {'Y', 'u', '\0', 'S', '.', '0'};
cout << strlen(name) << endl;
```

'\0'

String literals   常量字符串

- It isn't convenient to initial a string character by character.

- String literals can help.

```cpp
char name1[] = "Southern University of Science and Technology";
char name2[] = "Southern University of "    "Science and Technology";
char name3[] = "ABCD"; //how many bytes for the array?
```

0	name3+4
'D'	name3+3
'C'	name3+2
'B'	name3+1
'A'	name3+0

```cpp
const wchar_t[] s5 = L"ABCD";
const char16_t[] s9 = u"ABCD"; //since C++11
const char32_t[] s6 = U"ABCD"; //since C++11
```

String manipulation and examination

- Copy

```cpp
char* strcpy( char* dest, const char* src );
```

  ➢ Safer one:

```cpp
char *strncpy(char *dest, const char *src, size_t count);
```

- Concatenate: appends a copy of src to dest

```cpp
char *strcat( char *dest, const char *src );
```

- Compare

```cpp
int strcmp( const char *lhs, const char *rhs );
```

stringop.cpp

`string` class

- Null-terminated strings are easy to be out of bound, and to cause problems.

- `string` class provides functions to manipulate and examinate strings.

  ```
  std::string str1 = "Hello";
  std::string str2 = "SUSTech";
  std::string result = str1 + ", " + str2;
  ```

  stdstring.cpp

- Different types of strings

  ```
  std::string
  std::wstring
  std::u8string //(C++20)
  std::u16string //(C++11)
  std::u32string //(C++11)
  ```

Structures, Unions and Enumerations

struct

- A `struct` is a type consisting of a sequence of members.
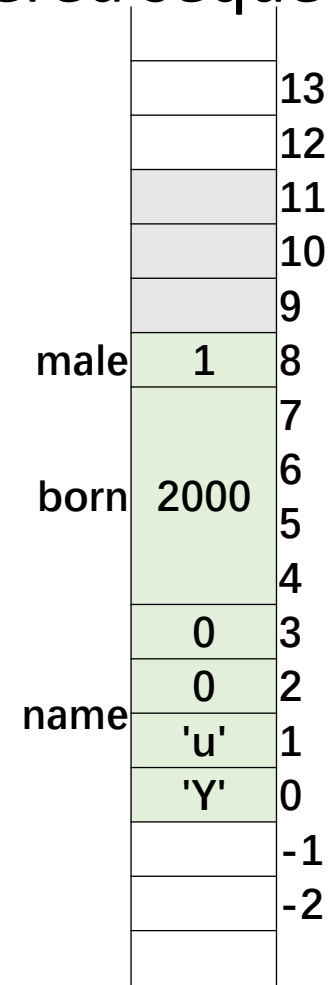- The members are allocated in an ordered sequence.

struct.c

```c
struct Student{
    char name[4];
    int born;
    bool male;
};

struct Student stu;
strcpy(stu.name, "Yu");
stu.born = 2000;
stu.male = true;

struct Student stu = {"Yu", 2000, true};

struct Student students[100];
students[50].born = 2002;
```
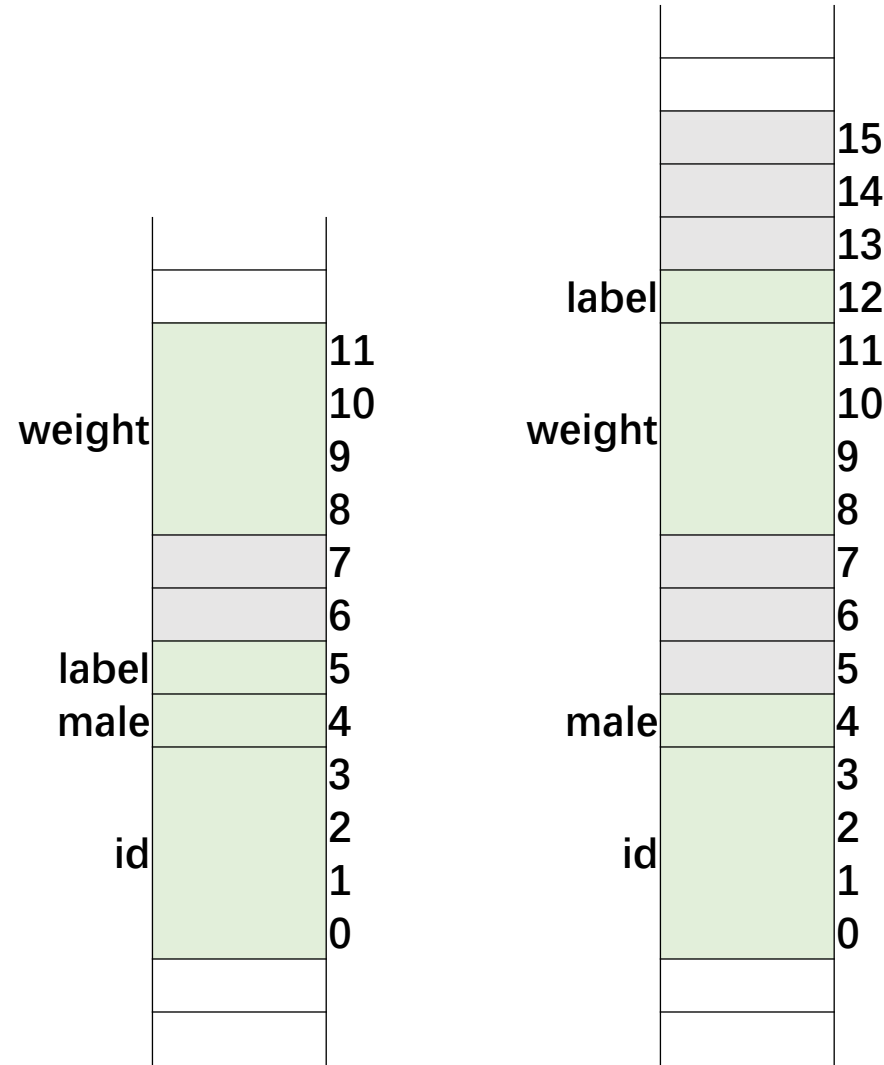
		13
		12
		11
		10
		9
male	1	8
		7
born	2000	6
		5
		4
	0	3
	0	2
name	'u'	1
	'Y'	0
		-1
		-2

Structure padding

- In order to align the data in memory, some empty bytes will be padded

```
struct Student1{
    int id;
    bool male;
    char label;
    float height;
};

struct Student2{
    int id;
    bool male;
    float height;
    char label;
};
```

`struct` in C and C++

- `struct` and `class` in C++ are identical except for several features.

```
struct Student1{
    int id;
    bool male;
    char label;
    float height;
};
Student1 stu;
stu.id = 123;
```

- No `typedef` needed in C++!

union

- `union` declaration is similar to `struct` declaration.

- The storage of members overlaps/shared.

  union ipv4address{

     std::uint32_t address32;

     std::uint8_t address8[4];

  };

  所有成员共享内存

- sizeof(union ipv4address) is 4.

union.cpp

		Union	Address
			p+4
address8[3]		127	p+3
address8[2]	address32	0	p+2
address8[1]		0	p+1
address8[0]		1	p+0
			p-1

enum

enum

- `enum` makes a new type.
- It provides an alternative to `const` for creating symbolic constants.
- Its members are integers, but they cannot be operands in arithmetic expressions.

```cpp
enum color {WHITE, BLACK, RED, GREEN, BLUE, YELLOW, NUM_COLORS};
enum color pen_color = RED;
pen_color = color(3);


cout << "We have " << NUM_COLORS << " pens." << endl;


pen_color += 1; //error!
int color_index = pen_color;
color_index += 1;
cout << "color_index = " << color_index << endl;
```

An example with struct, union and enum

```cpp
enum datatype {TYPE_INT8=1, TYPE_INT16=2, TYPE_INT32=4, TYPE_INT64=8};
```

```cpp
struct Point{
    enum datatype type;
    union {
        std::int8_t data8[3];
        std::int16_t data16[3];
        std::int32_t data32[3];
        std::int64_t data64[3];
    };
};
size_t datawidth(struct Point pt)
{
    return size_t(pt.type) * 3;
}
```

```cpp
int64_t l1norm(struct Point pt)
{
    int64_t result = 0;
    switch(pt.type)
    {
        case (TYPE_INT8):
            result = abs(pt.data8[0]) +
                     abs(pt.data8[1]) +
                     abs(pt.data8[2]);
            break;
        ...
```

enum.cpp

typedef

typedef

- typedef can create an alias for a type.
- It can be used to replace a possibly complex type name.

```cpp
typedef int myint;

typedef unsigned char vec3b[3];

typedef struct _rgb_struct{//name _rgb_struct can be omit
    unsigned char r;
    unsigned char g;
    unsigned char b;
} rgb_struct;

myint num = 32;

unsigned char color[3];
vec3b color = {255, 0, 255};

rgb_struct rgb = {0, 255, 128};
```

Typical `typedef` usages

`_uint8_t.h`
```
#ifndef _UINT8_T
#define _UINT8_T
typedef unsigned char uint8_t;
#endif /* _UINT8_T */
```

```
#if defined(_LP64)
typedef int wchar_t;
#else
typedef long wchar_t;
#endif
```