MECHTRON 2MD3

Data Structures and Algorithms for Mechatronics

Winter 2022

02 Fundamentals of C++

Department of Computing and Software

Instructor:

Omid Isfahanialamdari

January 12, 2022

Administration

- There will be **two mid-terms**
  - Mid-term 1: February 16, 2022
  - Mid-term 2: March 23, 2022
- first assignment on January 21, 2022. (deadline February 1, 2022)
- Check the announcements: The classes will be virtual until the week of Feb 7, 2022
- Office hour: Today at 15:00
- In case of problems:
  - First talk to me or TAs
  - Then, if not solved, to Associate Dean
  - Then, if not solved, to Dean
  - …

A Simple C++ Program

- Two integer inputs **x** and **y**
- Output their **sum**

```
1   #include <cstdlib>
2   #include <iostream>
3   /* This program inputs two numbers x and y and outputs their sum */
4   int main( ) {
5       int x, y;
6       std::cout << "Please enter two numbers: ";
7       std::cin >> x >> y; // input x and y
8       int sum = x + y; // compute their sum
9       std::cout << "Their sum is " << sum << std::endl;
10      return EXIT_SUCCESS; // terminate successfully
11  }
```

A Simple C++ Program

- Two integer inputs **x** and **y**
- Output their **sum**
- using namespace std; => You don't need to put std:: anymore.

```
1   #include <cstdlib>
2   #include <iostream>
3   using namespace std;
4   /* This program inputs two numbers x and y and outputs their sum */
5   int main( ) {
6       int x, y;
7       cout << "Please enter two numbers: ";
8       cin >> x >> y; // input x and y
9       int sum = x + y; // compute their sum
10      cout << "Their sum is " << sum << endl;
11      return EXIT_SUCCESS; // terminate successfully
12  }
```

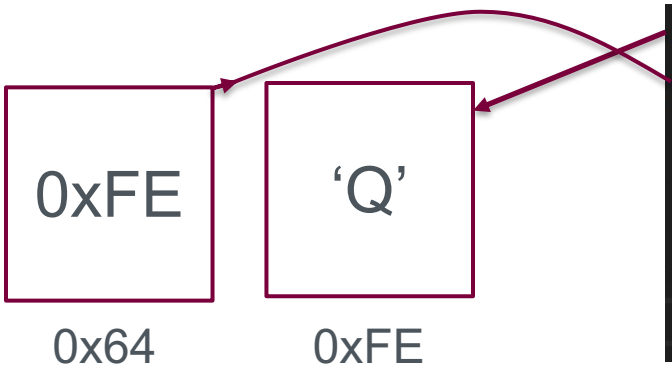
```
Please enter two numbers: 23 56
Their sum is 79
```

Fundamental Types

- **bool** Boolean value, either **true** or **false**
- **char** character
  - ‘a’ or ‘Q’
  - Backslash for special characters: ‘\n’ for newline, ‘\t’ for tab, ‘\\’ for backslash and …
- **int** integer
  - int octalNumber = 0400; //the same as 0x100 and 256
  - decimal numbers, takes 32 bits
- **short** short integer
  - takes at least 16 bits
- **long** long integer
  - takes at least 32 bits
- **float** single-precision floating-point number
- **double** double-precision floating-point number

Pointers

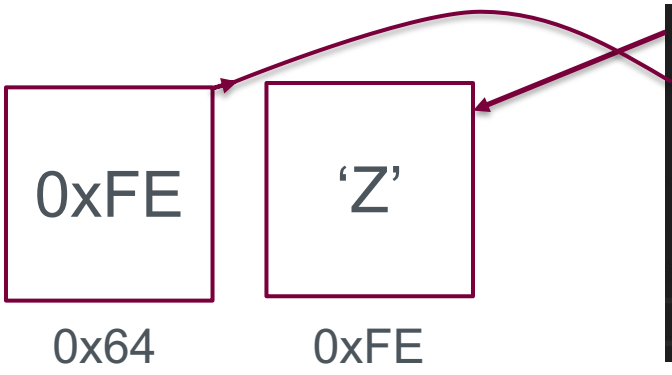
- A **pointer** is a variable that holds the address where a variable is stored
- Given a type **T**, the type **T*** denotes a pointer to a variable of type T
- Two important operators:
  - address-of operator: &
  - dereferencing operator: *




```
char ch = 'Q';
char* p = &ch;          // p holds the address of ch
cout << *p;             // outputs the character 'Q'
ch = 'Z';               // ch now holds 'Z'
cout << *p;             // outputs the character 'Z'
*p = 'X';               // ch now holds 'X'
cout << ch;             // outputs the character 'X'
```

Pointers

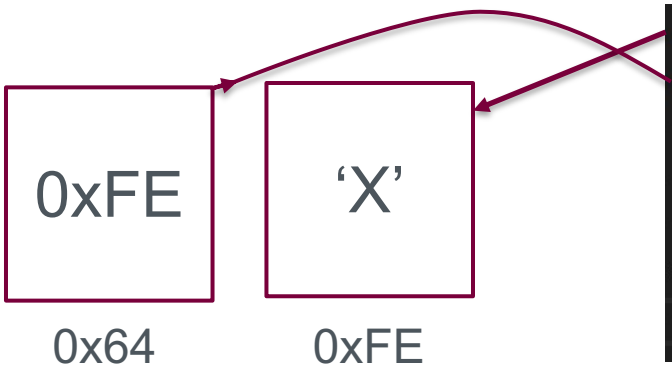
- A **pointer** is a variable that holds the address where a variable is stored
- Given a type **T**, the type **T*** denotes a pointer to a variable of type T
- Two important operators:
  - address-of operator: &
  - dereferencing operator: *




```
char ch = 'Q';
char* p = &ch;          // p holds the address of ch
cout << *p;             // outputs the character 'Q'
ch = 'Z';               // ch now holds 'Z'
cout << *p;             // outputs the character 'Z'
*p = 'X';               // ch now holds 'X'
cout << ch;             // outputs the character 'X'
```

Pointers

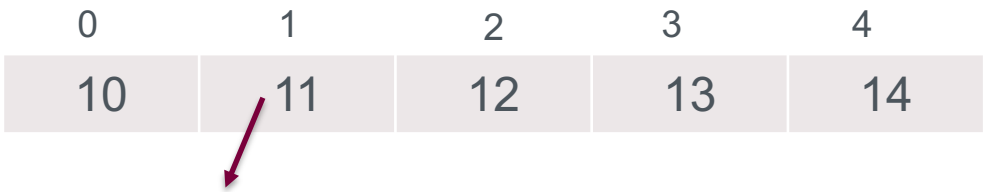
- A **pointer** is a variable that holds the address where a variable is stored
- Given a type **T**, the type **T*** denotes a pointer to a variable of type T
- Two important operators:
  - address-of operator: &
  - dereferencing operator: *




```
char ch = 'Q';
char* p = &ch;          // p holds the address of ch
cout << *p;             // outputs the character 'Q'
ch = 'Z';               // ch now holds 'Z'
cout << *p;             // outputs the character 'Z'
*p = 'X';               // ch now holds 'X'
cout << ch;             // outputs the character 'X'
```

Arrays and Pointers

- An **array** is a collection of elements of the same type
- Given any type **T** and a constant **N**, a variable of type **T[N]** holds an array of **N** elements, each of type **T**
- int m[5];
- initialization:
  - int m[ ] = {10, 11, 12, 13, 14};
- access using index

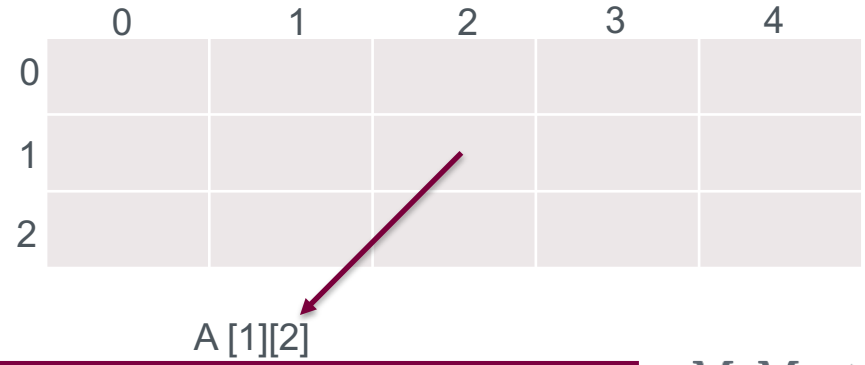



```
int a[]  = {10, 11, 12, 13};      // declares and initializes a[4]
bool b[] = {false, true};         // declares and initializes b[2]
char c[] = {'c', 'a', 't'};       // declares and initializes c[3]
```

- **array name: pointer to the first element**

```
char c[] = {'c', 'a', 't'};
char* p = c;                      // p points to c[0]
char* q = &c[0];                  // q also points to c[0]
cout << c[2] << p[2] << q[2];     // outputs "ttt"
```

- int A [3][5];



C-Style Structures and **enum** types

- Structures are used to store an aggregation of elements.

```
enum MealType { NO_PREF, REGULAR, LOW_FAT, VEGETARIAN };

struct Passenger {
    string      name;          // passenger name
    MealType    mealPref;      // meal preference
    bool        isFreqFlyer;   // in the frequent flyer program?
    string      freqFlyerNo;   // the passenger's freq. flyer number
};
```

- An enumeration is a user-defined type that can hold any of a set of discrete values:
- **enum** Day { SUN, MON, TUE, WED, THU, FRI, SAT };
- **enum** Mood { HAPPY = 3, SAD = 1, ANXIOUS = 4, SLEEPY = 2 };
- Day today = THU; // today may be any of MON ... SAT
- Mood myMood = SLEEPY; // myMood may be HAPPY, ..., SLEEPY

C-Style Structures and **enum** types

- Structures are used to store an aggregation of elements.

```
enum MealType { NO_PREF, REGULAR, LOW_FAT, VEGETARIAN };

struct Passenger {
    string      name;          // passenger name
    MealType    mealPref;      // meal preference
    bool        isFreqFlyer;   // in the frequent flyer program?
    string      freqFlyerNo;   // the passenger's freq. flyer number
};
```

- Passenger pass = { "John Smith", VEGETARIAN, true, "293145" };
- pass.name = "Pocahontas"; // change name
- pass.mealPref = REGULAR; // change meal preference

C & C++ in Abstraction View

- C supports Procedure-Oriented programming
  - Procedure (function) + data structure
    - Procedure (function) : manipulate data
- C++ supports Object-Oriented programming
- Object-oriented programming (OOP) is a programming paradigm that uses objects and their interactions to design applications and computer programs.
- Data abstract + reusable subtypes with features such as:
  - Encapsulation, Polymorphism, Inheritance

C++ Classes

```
class Passenger{
    public:
      // member variables
      string name;
      MealType mealPref;
      bool isFreqFlyer;
      string freqFlyerNo;

      //member methods
      void setName(string newName);
      …
   private:
      bool _isNameValid(string nameToSet);
```

Questions?