The first

Thursday, March 11, 2021 10:30 AM

Bùi Lê Phi Long	16619

Arrays

- Arrays are sequences of objects
  int arr[5] ={ 10,20,30,40,50}
- We can access individual array elements through a subcript []
  arr[0]= 2 // change the value of first element

When declaring an array, 5 is the number of elements, but when we want to access the element, remember to starts the index from 0.

Pointers

- Another way to access an object in memory is through a pointer
- the same as C.

References

- A reference type is an alias to an existing object in emmory
- Ref must be initialized

```
int main()
{
int x = 123;
int& y = x;
}
```

  - These two variables x,y share the same reference so if we change a differnet value to either of them, they both change as we have one object in memory, but 2 different names.

String

Definition

- to use string type, we need to include <string> header in our program

Concatenating Strings

- we can add string literal to our string using the compound operator

```
string s = "hello";
s+=" world!";
```

- we can add another strings to our string using the + operator

```
string s1= "hello";
string s2="world!";
string s3 =s1+s2;
```

s3 is now equal to" helloworld!"

Accessing Characters

- Individual characters of a string can be accessed through a subscript operator[] or via a member function

Comparing Strings

- A string can be compared to string literals and other strings using the equality == operator

std::string s1 = "Hello";
std::string s2 = "World.";
if (s1 == s2)
{
std::cout << "The strings are
	Output the strings are not equal since s1 and s2 has each different characters

equal.";
}
else
{
std::cout << "The strings are not equal.";
}	

String input

- Preferred way of accepting a string from the standard input is via std::getline function which takes std::cin and our string as parameters:

string s;
cout<<"Enter a string";
getline(cin,s);
cout<<"String entered: "<<s<<endl;
	the string you enter would appear on the screen

- ★ We use the getline because our string can contain white spaces, and if we used cin function alone, it woud accept only a part of the string
- The getline function has the following signarute getline(readfrom, into); the funtion reads a line of text from the standard input(cin) into a string (s) variable

A pointer to a string

A string has member funtion .c_str() which returns a pointer to its first element. It's also said it returns a pointer to a null-terminated character array our string is made of:

#include <iostream>
#include <string>
int main()
{
std::string s = "Hello World.";
std::cout << s.c_str();
}
	This member function is of type const char and useful when we want to pass out string variable to a function accepting a const char* parameter

Substrings

To create a substring from a string, we use the *.substr()* member function

- The function returns a substring that starts at a certain pos in the main string and is of a certain length : *.substring(starting_pos,length)*

Finding a substring

- TO find a substring in a string, we use find() member function. It searches for the substring in a string.
- If the substring is found, the function returns the position of the first found substring. This position is the position of a character where thee substring starts tin the main string
- If the substring is not found, the function returns a value that is npos

#include <iostream>
#include <string>
int main()
{
std::string s = "This is a Hello World string.";
std::string stringtofind = "Hello";
std::string::size_type found = s.find(stringtofind);
if (found != std::string::npos)
{
std::cout << "Substring found at position: " << found;
}
	This program print" Substring found" when s.find returns true otherwise print not found

else
{
std::cout << "The substring is not found.";
}
}	

Automatic Type Deduction

- We can automatically deduce the type of an object using the auto specifier which deduces the type of an object based on its initializer type
  Ex :

auto c = 'a'	automatically declare variable c to character type
auto x = 123	int type
auto d =123.456/789.10	double type

We can use auto as part of the reference type:

int x =123;
auto& y = x;
	now y is of int& type

or the constant type:

const auto x =123	x is of const int type

★ We use the auto specifier when the type is håd to dedcude manually or cumbersome to type due to the length.

Exercises

1. Array Definition

Write a program that defines and initializes an array of five doubles. Change and print
the values of the first and last array elements.

```
#include<iostream>
using namespace std;
int main()
{
int array[]={1,2,3,4,5};
array[0]=6;
arr[4]=7;
}
```

2. Pointer to an Object

Write a program that defines an object of type double. Define a pointer that points to that
object. Print the value of the pointed-to object by dereferencing a pointer.

```
#include<iostream>
using namespace std;
int main()
{
 double x = 36.9;
double *p=&x;
cout<<" x = "<<*p;
}
```

3. Reference Type

Write a program that defines an object of type double called mydouble. Define an

object
of reference type called myreference and initialize it with mydouble. Change the value of
myreference. Print the object value using both the reference and the original variable.
Change the value of mydouble. Print the value of both objects.

```
#include<iostream>
using namespace std;
int main()
{
double mydouble=6.9;
double& myreference;
myreference = 9.6;
cout<<"mydouble="<<mydouble<<endl;
```

4. Strings

Write a program that defines two strings. Join them together and assign the result to a
third-string. Print out the value of the resulting string.

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
string s1="Long";
string s2=" dep trai";
string s = s1+s2;
cout<<s;
}
```

5. Strings from Standard Input

Write a program that accepts the first and the last name from the standard input using
the std::getline function. Store the input in a single string called fullname. Print out the string

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
string name;
cout<<"Enter first name and lát name)"
getline(cin,name);
cout<<"Your name is:" <<name;
}
```

6. Creating a substring

Write a program that creates two substrings from the main string. The main string is
made up of first and last names and is equal to “John Doe.” The first substring is the first
name. The second substring is the last name. Print the main string and two substrings
afterward.

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
string name = ""John Doe";
string firstname =
name.substr(0,4);
string
lastname=name.substring(5,3);
}
```

the function name.substr(0,4) get the substring of the *string name* from index 0 to the length of 4, and so do the name.substring(5,3), then after that we will create two substrings of firstname and lastname.

7. Finding a single Character

Write a program that defines the main string with a value of “Hello C++ World.” and checks if a single character ‘C’ is found in the main string

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
string s= "Hello C++ World";
if(s.find('C')!=npos)
cout<<"found"<<endl;
else
cout<<"not found"<<endl;
}
```

the function s.find('C') help us with the work of searching the character C in string *s*, if it is not found, it would return npos. We use the if else to create the program

8. Finding a Substring

Write a program that defines the main string with a value of “Hello C++ World.” and checks if a substring “C++” is found in the main string

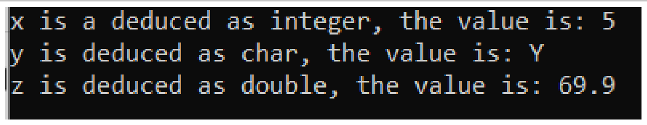
```
#include<iostream>
#include<string>
using namespace std;
int main()
{
string s = " hello c++ world";
string a = "C++"
auto found_substr= s.find(a);
if(found_substr!=npos)
cout<<"found"<<endl;
else
cout<<"not found"<<endl;
```

In this case, I use the auto deduction to a variable call found_substr, this variable would return the same as program mentioned above.
SO string.find() function can search substrings, not only character.

9. Automatic type deduction

Write a program that automatically deduces the type for char, int, and double objects
based on the initializer used. Print out the values afterward.

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
auto x = 5;
```



```
x is a deduced as integer, the value is: 5
y is deduced as char, the value is: Y
z is deduced as double, the value is: 69.9
```

As expected, these variables are automatically deduced as they should be deduced, no comments.

```
auto y = "Y";
auto z = 69.9;
cout<<"x is a deduced as integer,
the value is: "<<x<<endl;
cout<<"y is deduced as char, the
value is: "<<y<<endl;
cout<<"z is deduced as double,
the value is: " << z<<endl;
}
```