Programming Language

Pointers and References

(Day 3 Part 1)

By: Bhaskar Ghosh
bjghosh@gmail.com

Updated: 5 August 2024

C++ Programming Language

Pointers and references (Day 3 Part 1)

- Pointers and References
- Pointers and Arrays
- Call by Value VS Call by Reference VS Call by Pointer
- More on Pointers
    - Pointer to a pointer, Void pointer, Pointer to a function, etc.
- C++ Memory Management: new and delete

C++ Programming Language

Pointers

- A pointer is a variable that stores the memory address of another variable.
- Every variable has an associated location in the memory.
- Memory address of a variable can be accessed using the & operator.

```
int var1 = 3; cout << "Address of var1: "<< &var1;
```

- A pointer is declared using the * operator.

```
int* ptr = &var1;
```

- Value of the variable whose address is stored in a pointer variable can be accessed using the* operator.

```
cout << *ptr; //prints value of var1
*ptr = 5; //changes value of var1
```

- When * is used with pointers, it's called the ***dereference operator***.

C++ Programming Language

Pointers and Arrays

- If we have to declare two (or more) pointers together in the same line, we will need to use the asterisk symbol before each variable name.

```
int* var1, *var2; // Both var1 and var2 are pointers
```

```
int* var1, var2; // var1 is a pointer, var2 is an integer variable
```

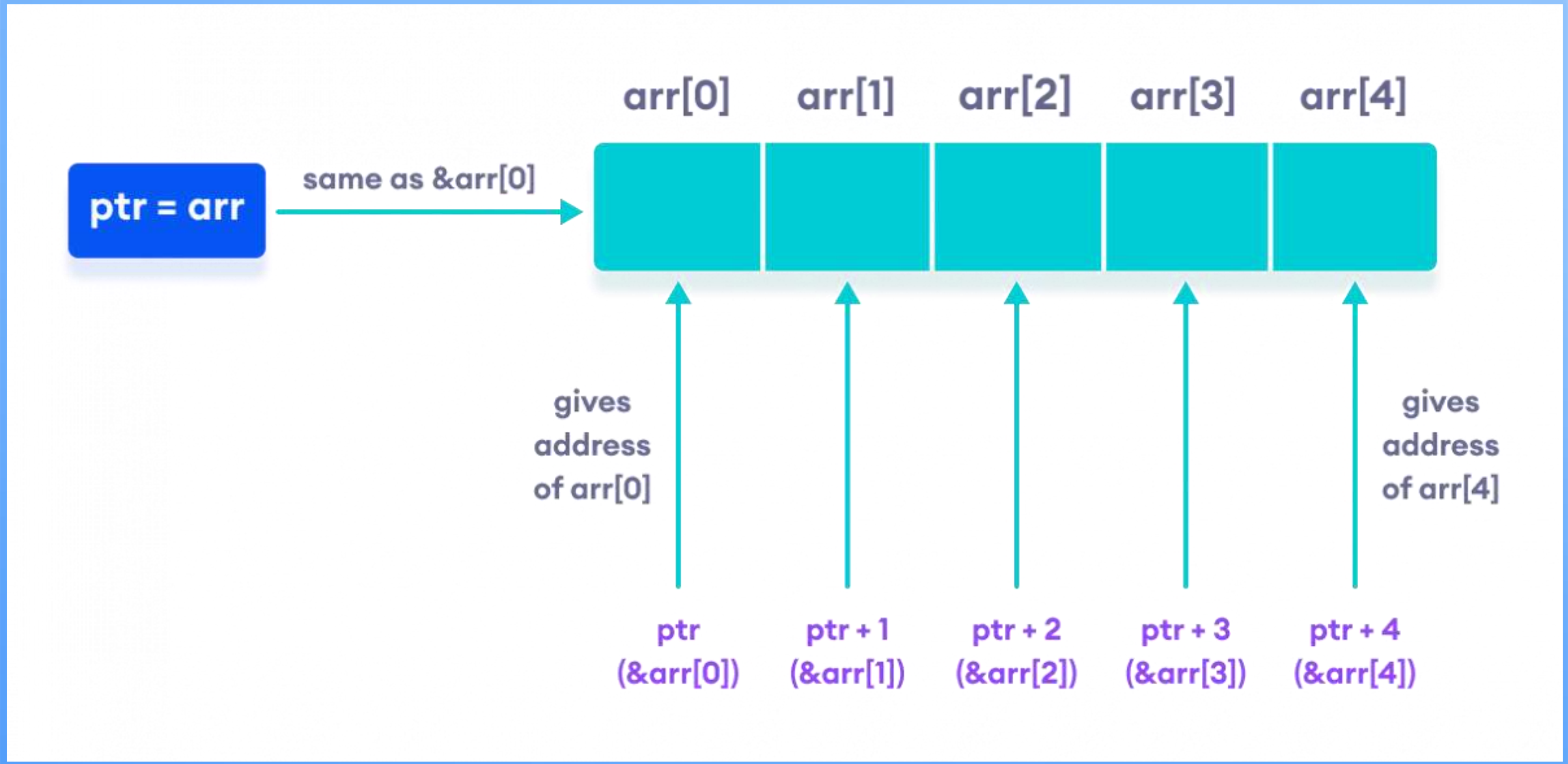
- A pointer can also store the address of a cell of an array.

```
int* ptr; int arr[5];
ptr = arr; // store the address of the first element of arr in ptr
```

- This is same as: `ptr = &arr[0];`

C++ Programming Language

Pointers and Arrays



C++ Programming Language

References

- In C++, we use a reference to create an alias for a variable.

```
int var1 = 5;
int& ref_var1; // ref_var1 is a reference (alias) of variable var1
```

- We can place the & sign with data type or with a variable while creating a reference.

```
int var1 = 10; // create a variable

int &ref_var1 = var1; // valid but not a standard practice

int& ref_var2 = var1; // valid and a standard practice
```

C++ Programming Language

References

- We must initialize references at the time of declaration.

```
int var1 = 10; // create a variable

// incorrect code [reference not initialized]
int& ref_var1;
ref_var1 = var1;

int& ref_var1 = var1; // correct code
```

- Once we create a reference to a variable, it cannot be changed to refer to another variable.

```
int var2 = 50;
// ref_var1 still is alias of var1. Value of var2 is copied to var1
ref_var1 = var2;
```

C++ Programming Language

References VS Pointers

Reference	Pointer
A reference is an alias for a variable.	A pointer is a different variable that stores the memory address of a variable.
We can directly use the reference without any operator.	We use the dereference operator * to access the value pointed by a pointer.
We can not change the reference to be the alias of another variable.	We can change the pointer to point to another variable.

C++ Programming Language

Ways to Pass Parameters to a Function

- In how many ways, can we pass parameters to a function?

```
void myFunc(int v){ /*do something*/ }
int a = 10; myFunc(a);
```

Call By Value

```
void myFunc2(int& r){ /*do something*/ }
int a = 10; myFunc2(a);
```

Call By Reference

```
void myFunc3(int* r){ /*do something*/ }
int a = 10; int* ptr = &a;
myFunc3(ptr);
```

Call By Pointer

- When the values of variables do not need to be changed, we can pass them as const references.

C++ Programming Language

Ways to Pass Parameters to a Function

- Use references instead of pointers
  - Easier
  - Less error-prone
  - As it doesn't involve direct pointer operations.
- Pointers should only be used to pass arguments in contexts where pointers are specifically needed.
- References in C++ are an alternative to pointers.
- Unlike pointers, references cannot be null and do not require dereferencing to access the value they refer to.

C++ Programming Language

More on Pointers

- Pointer to Pointer

```
int var1 = 5;
int *ptr1 = &var1; // ptr1 is a pointer that points to variable var1
int **ptr2 = &ptr1; // ptr2 is a pointer that points to pointer ptr1
```

- Void Pointer
    - Also known as void*
    - References a variable without a specific data type.
    - Can be typecasted to store any data type's address.
    - Cannot be directly dereferenced.
    - To access their content, we must convert them to a pointer of a specific data type.

C++ Programming Language

More on Pointers

- Null Pointer

```
int *ptr = NULL;
```

  - We can assign NULL to a pointer.
  - NULL has a value of zero.
  - Using NULL, we can create valid pointers without storing a variable's address in the pointer.
  - We should assign NULL during pointer declaration to prevent potential runtime errors.
- A valid pointer may become invalid when the object it references is deallocated, ending its lifecycle.
  - We should reset a pointer to NULL after the object it referenced is deallocated.

C++ Programming Language

Pointer to a function

- Address of a Function
  - Just like variables in our code, the functions we declare also have a corresponding memory address associated with them.
  - It points to the first instruction line of the function.
- Declaring a pointer to a function:
  - Syntax

```
return_type (*pointer_name)(data_type1, data_type2, ...);
```

  - Example:

```
int (*ptr)(int, int);
```

- Initializing a Function Pointer:
  - Address of a function can be accessed by just writing the function name without the brackets

```
int add(int a, int b){} //function
int (*ptr)(int, int); ptr = add; //function pointer
```

C++ Programming Language

C++ Memory Management: new and delete

- We can allocate and deallocate memory at runtime (dynamically) using the new and delete operators respectively.
- new

```
int* point_var; // declare an int pointer

point_var = new int; // dynamically allocate memory

*point_var = 45; // assign value to allocated memory

int* point_var = new int{45}; //allocate & assign value
```

- delete

```
delete point_var; // deallocate the memory

point_var = nullptr; // set pointer to nullptr
```

C++ Programming Language

C++ Memory Management: new and delete

- Dynamic Memory Allocation for Arrays
- new

```
// memory allocation of 10 number of integers
ptr = new int[10];
```

- delete

```
delete[] ptr; // deallocate the memory

ptr = nullptr; // set pointer to nullptr
```

- Dynamic memory allocation can make memory management more efficient, especially for arrays, where many times we may not know the size of the array until runtime.

C++ Programming Language

Smart Pointers

- Memory Leak
  - The problem with pointers is that if we forget to free the memory held by the pointer, then that memory will remain occupied, for the entire duration the program runs.
  - If this happens enough times, the program will run out of space, and will get a memory exception.
- Smart Pointers

```
int (*ptr)(int, int);
```

  - Automatically deallocates the memory held, if the pointer goes out of scope.
  - Defined inside the header file <memory>
  - Syntax:
  - Types

```
smart_pointer_type <data_type> pointer_name(new data_type());
```

    - unique_ptr, shared_ptr, weak_ptr
    - ***Homework: Find out more about Smart Pointers***

More on pointers: https://pointers.co.in/

Programming Language

struct, enum & OOPs intro

(Day 3 Part 2)

By: Bhaskar Ghosh

bjghosh@gmail.com

Updated: 5 August 2024

C++ Programming Language

Structures, Enumerations & Intro to OOPs (Day 3 Part 2)

- C++ Structures
- Structure and Function
- Pointers to Structure
- C++ Enumeration
- Problem Solving using C++ [Level 3]
  - Logic Building and Debugging

C++ Programming Language

Structures

- Structure is a user-defined data type.
- Structures are used to combine different types of data types, just like an array is used to combine same type of data types.
- A structure is declared by using the keyword struct.
- Define a Structure Variable

```
Student stud;
```

- Access Members of a Structure

```
stud.name = "Ajay";
```

- In C++, a structure can also have a member function.

```
struct Student
{
    string name;
    string dept;
    int sem;
    float gpa;
};
```

C++ Programming Language

Structures, Functions and Pointers

- Just like primitive data types, we can also pass and return structure variables to and from a function.

```
Student getStudentData();
void displayStudentData(const Student&);
```

- Just like with primitive data types, we can also create a pointer to point to a structure variable.

```
Student p;
Student *ptr = &p;
```

- We can access a structure member using the normal pointer notation, or using the arrow (->) operator.

```
 (*ptr).name;
ptr -> name;
```

C++ Programming Language

Enumerations

- An enumeration is a user-defined data type that consists of integral constants.
- We use the keyword enum to define an enumeration.

```
enum Day { SUN, MON, TUE, WED, THU, FRI, SAT };
```

- By default the constants defined as part of an enumeration will have values like 0, 1, 2, 3, and so on.
- We can also change the constant values.

```
enum Day { SUN=101, MON, TUE, WED, THU, FRI, SAT };
```

```
enum Day { SUN=10, MON=20, TUE=30, WED=40, THU=50,
           FRI=60, SAT=70 };
```

C++ Programming Language

Problem Solving using C++ [Level 3]

Logic Building and Debugging

C++ Programming Language

Problem Solving using C++ [Level 3] – (Pointers & References)

1. WAP to enter 2 numbers from user and display then using pointers.
2. WAP to enter N numbers from user and print the largest using pointers.
3. Given the string "A string." Print on one line the letter on the index 0, the pointer position and the letter t. Then, update the pointer to pointer +2. Then, in another line print the pointer and the letters r and g of the string (using the pointer).
4. Consider three local arrays, all the same size and type (say float). The first two are already initialized to values. Write a function called addarrays() that accepts the addresses of the three arrays as arguments; adds the contents of the first two arrays together, element by element; and places the results in the third array before returning. A fourth argument to this function can carry the size of the arrays.

C++ Programming Language

Problem Solving using C++ [Level 3] – (struct)

1. A structure brings together a group of
   a. items of the same data type.
   b. related data items.
   c. integers with user-defined names.
   d. variables.
2. Write a structure specification that includes three variables—all of type int—called hrs, mins, and secs. Call this structure time.
3. If you have three variables defined to be of type struct time, and this structure contains three int members, how many bytes of memory do the variables use together?

C++ Programming Language

Problem Solving using C++ [Level 3] – (enum)

4. Write a statement that declares an enumeration called players with the values B1, B2, SS, B3, RF, CF, LF, P, and C.
5. Assuming the enum type players as declared in Question 4, define two variables joe and tom, and assign them the values LF and P, respectively.
6. Assuming the statements of Questions 4 and 5, state whether each of the following statements is legal.
   a. joe = QB;
   b. tom = SS;
   c. LF = tom;
   d. difference = joe - tom;

C++ Programming Language

Problem Solving using C++ [Level 3] – (struct and enum)

7. A phone number, such as (+91) 612-2236478, can be thought of as having three parts: the country code (+91), the state code (612), and the number (2236478). Write a program that uses a structure to store these three parts of a phone number separately. Call the structure **Phone**. Create two structure variables of type phone. Initialize one, and have the user input a number for the other one. Then display both numbers.
8. Write a structure **Employee** to store name, designation and salary of employee. The designation should be an enum. Take user input for 2 employees and display them.
9. Modify the code in question 8, and input the employees in an array (array of structure), and keep entering values and storing them, until user types "done".