

Indirect Access

Name:

Student ID:

Date:

Learning Journey and Evidence

Review pointers, and references.

What did you think while reviewing this in the [Programmer's Guide](#)? Which aspects did you already know, which were challenging? Which explanations helped, which were confusing? Etc.

Pointers: A pointer is a data type built into the programming language. A pointer has a value, that stores the location of another value. The pointer's value is the memory address of the value it *points* to this means that we can point to any value in memory, regardless of where it is. A pointer value is the same as any other value it can be stored in local variables, global variables, it can be passed to a function in parameter and it can be returned from a function.

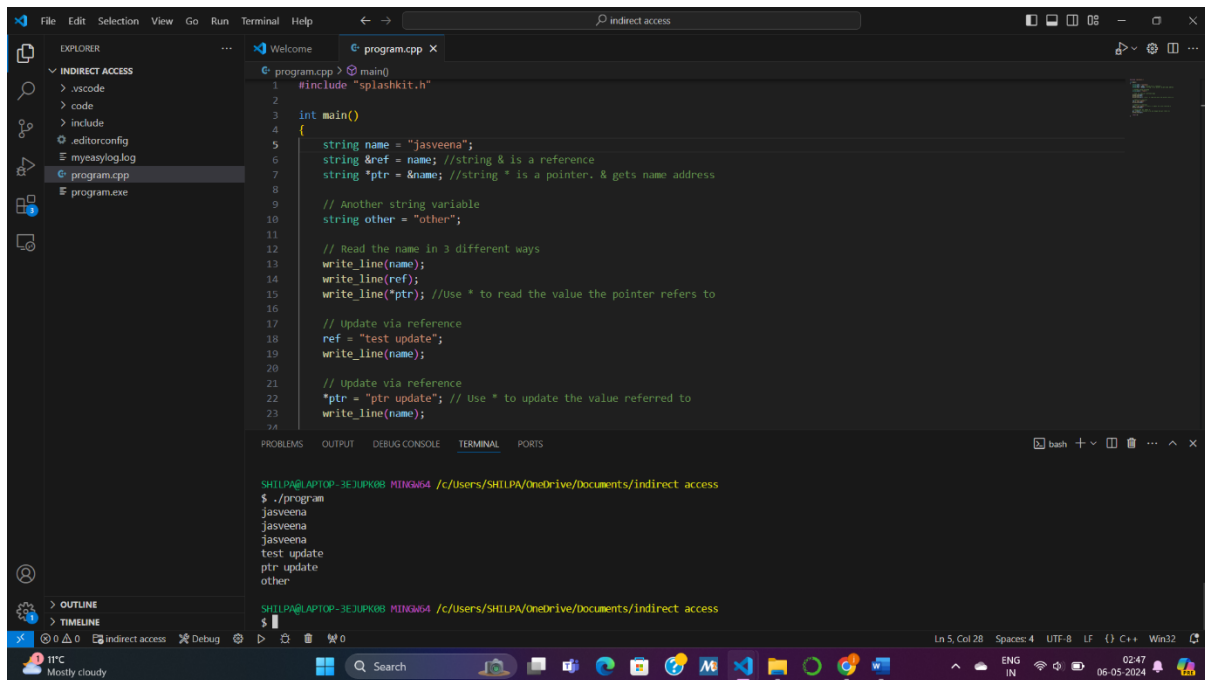
References: Pointers and references both store the address of another value in memory. They are both used to refer to another value. While in pointer we need to manually get the address and store it in the pointer in reference the compiler takes care of this for us.

What other resources did you use to help understand these concepts? Any good resources you would recommend to others?

I used youtube to deepen my understanding of different headers.

Create a small program to demonstrate the use of pointers.

Show your process for this and highlight any realisations you gain.



The screenshot shows the Visual Studio Code editor with a C++ file named `program.cpp` open. The code defines a `main` function that demonstrates string references and pointers. It includes `<stringkit.h>` and uses `write_line` for output. The program initializes a string `name` with the value "jasveena", creates a reference `ref` and a pointer `ptr` both pointing to `name`, and also declares a separate string `other` with the value "other". It then prints the values of `name`, `ref`, `ptr`, and `other` in three different ways. After printing, it updates `ref` to "test update" and `ptr` to "ptr update", and prints the values again to show that the changes are reflected in the original memory location.

```
1 #include "splashkit.h"
2
3 int main()
4 {
5     string name = "jasveena";
6     string &ref = name; //string & is a reference
7     string *ptr = &name; //string * is a pointer, & gets name address
8
9     // Another string variable
10    string other = "other";
11
12    // Read the name in 3 different ways
13    write_line(name);
14    write_line(ref);
15    write_line(*ptr); //Use * to read the value the pointer refers to
16
17    // Update via reference
18    ref = "test update";
19    write_line(name);
20
21    // Update via reference
22    *ptr = "ptr update"; // Use * to update the value referred to
23    write_line(name);
24 }
```

The terminal window shows the execution of the program. The output is as follows:

```
SHILPA@LAPTOP-3EJURK08 MINGW64 /c/Users/SHILPA/OneDrive/Documents/indirect access
$ ./program
jasveena
jasveena
jasveena
test update
ptr update
other
```

Include a hand execution of the program to explore how it works.

HAND EXECUTION (PROGRAM-1)

VARIABLES

- NAME \Rightarrow (initialized with the value "Jasveena")
- REF \Rightarrow a reference ('&') to the 'name' variable.
- PTR \Rightarrow a pointer (*), it holds the memory address of 'name'.
- OTHER \Rightarrow It is another 'String Variable'.

OUTPUTS

`write_line (name);` \Rightarrow outputs what is said in "name"
i.e. Jasveena

`write_line (ref);` \Rightarrow same as (name)

`write_line (*ptr);` \Rightarrow same as (name).

UPDATE VIA REFERENCE

`ref = "test update";`
`write_line (name);`

output \Rightarrow test update

UPDATE VIA POINTER (*PTR)

`*ptr = "ptr update";`
`write_line (name);`

CHANGE POINTER REFERENCE

`ptr = &other;`
`write_line (*ptr);`

Terminal

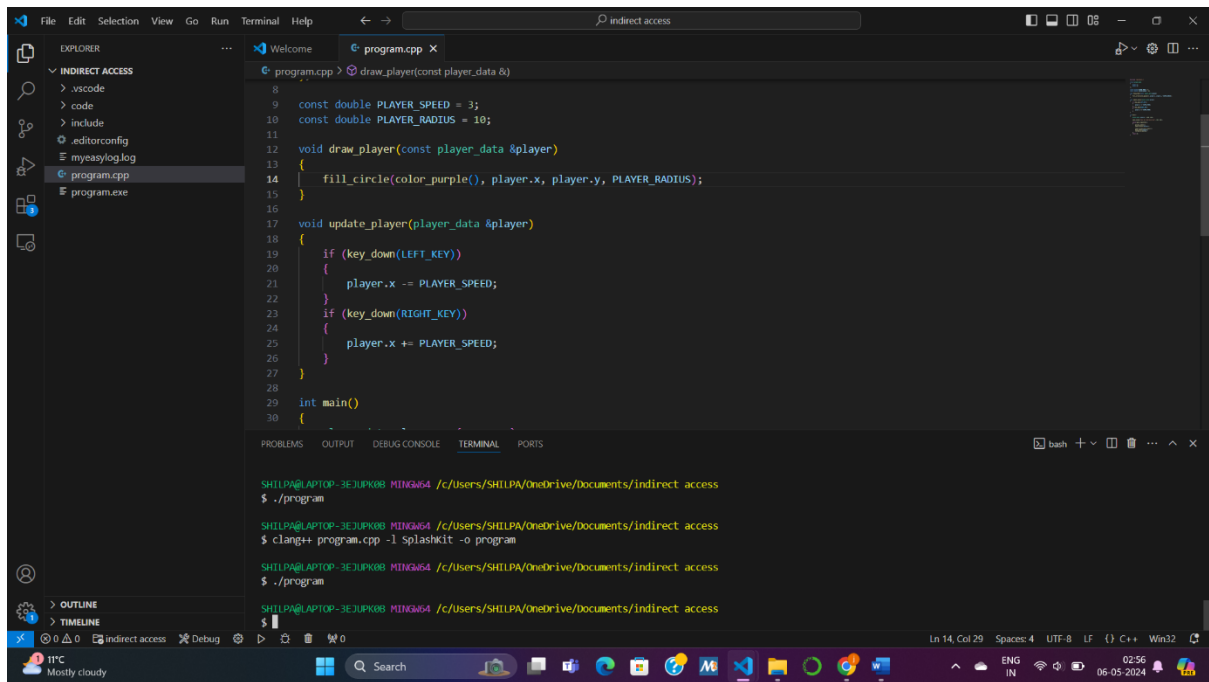
```
Jasveena
Jasveena
Jasveena
```

Final Terminal

```
Jasveena
Jasveena
Jasveena
Test update
ptr update
other
```

Create a small program to demonstrate the use of pass-by reference.

Show your process for this and highlight any realisations you gain.



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project named 'INDIRECT ACCESS' with files: .vscode, code, include, editorconfig, myeasylog.log, program.cpp, and program.exe. The main editor displays the contents of 'program.cpp', which includes constants for player speed and radius, and functions for drawing and updating a player. The terminal at the bottom shows the execution of the program, which outputs 'SHILPA@LAPTOP-3EJUPK08 MINGW64 /c/Users/SHILPA/OneDrive/Documents/indirect access' and then 'SHILPA@LAPTOP-3EJUPK08 MINGW64 /c/Users/SHILPA/OneDrive/Documents/indirect access'.

```
8
9  const double PLAYER_SPEED = 3;
10 const double PLAYER_RADIUS = 10;
11
12 void draw_player(const player_data &player)
13 {
14     fill_circle(color_purple(), player.x, player.y, PLAYER_RADIUS);
15 }
16
17 void update_player(player_data &player)
18 {
19     if (key_down(LEFT_KEY))
20     {
21         player.x -= PLAYER_SPEED;
22     }
23     if (key_down(RIGHT_KEY))
24     {
25         player.x += PLAYER_SPEED;
26     }
27 }
28
29 int main()
30 {
```

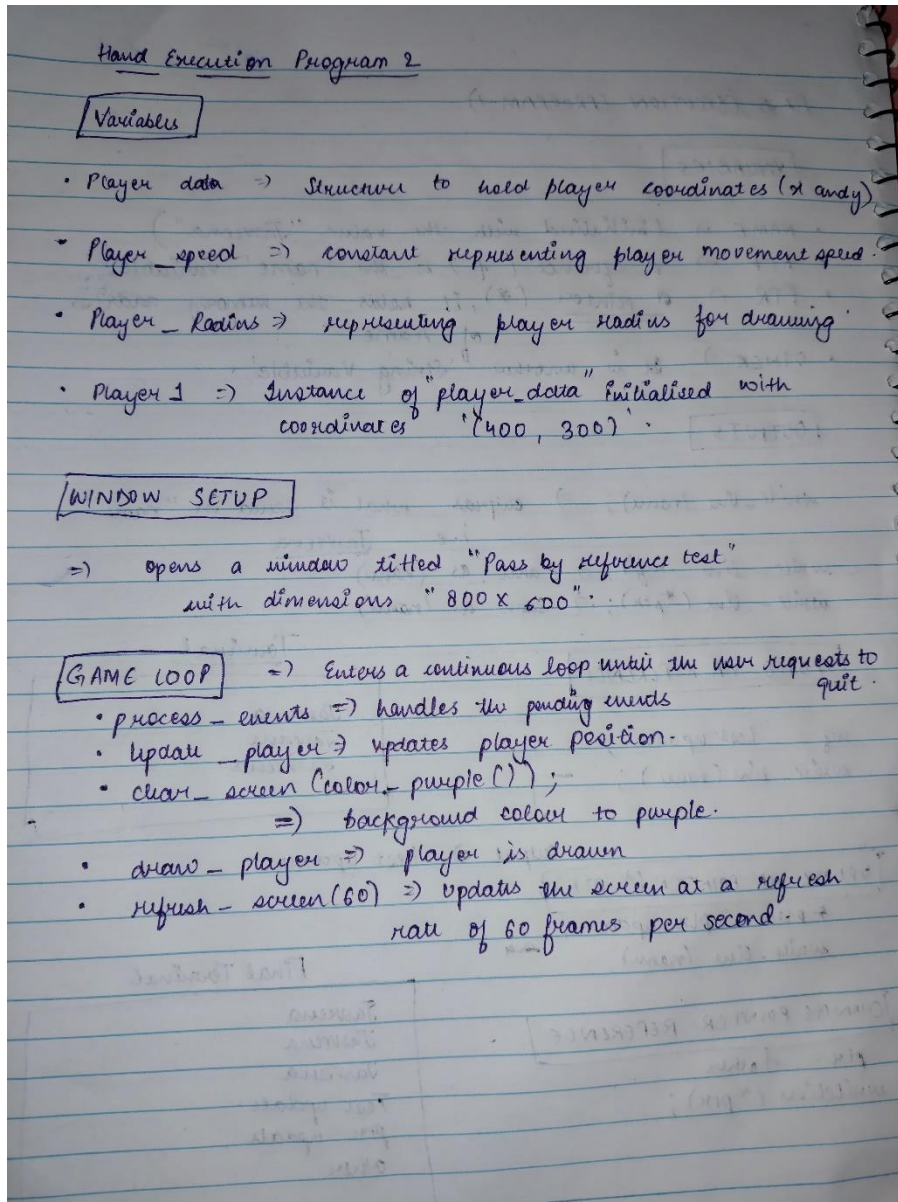
```
SHILPA@LAPTOP-3EJUPK08 MINGW64 /c/Users/SHILPA/OneDrive/Documents/indirect access
$ ./program

SHILPA@LAPTOP-3EJUPK08 MINGW64 /c/Users/SHILPA/OneDrive/Documents/indirect access
$ clang++ program.cpp -l SplashKit -o program

SHILPA@LAPTOP-3EJUPK08 MINGW64 /c/Users/SHILPA/OneDrive/Documents/indirect access
$ ./program

SHILPA@LAPTOP-3EJUPK08 MINGW64 /c/Users/SHILPA/OneDrive/Documents/indirect access
$
```

Include a hand execution of one of the calls to swap to explore how it works.



Update your Fly Catch program to make use of pass by reference

Show your process for this and highlight any realisations you gain.

```

1 #include <iostream>
2 #include <string>
3 #include <limits>
4
5 using namespace std;
6
7 // Function to read an integer within a specified range
8 int read_integer_range(const string& prompt, int min_val, int max_val) {
9     int value;
10    while (true) {
11        cout << prompt;
12        if (cin >> value && value >= min_val && value <= max_val) {
13            break; // Valid input, break the loop
14        } else {
15            cout << "Invalid input. Please enter an integer between " << min_val << " and " << max_val << ". " << endl;
16            cin.clear(); // Clear error state
17            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Discard invalid input
18        }
19    }
20    return value;
21 }
22
23 // Function to read and print product details
24 void read_and_print_product() {
25     string name, brand, category;
26     int stock_level;
27
28     cout << "Enter product details: " << endl;
29
30     cout << "Name: ";
31     cin >> name;
32
33     cout << "Brand: ";
34     cin >> brand;
35
36     cout << "Category: ";
37     cin >> category;
38 }
39
40 int main() {
41     read_and_print_product();
42     return 0;
43 }

```

Capture any other study and practice used to master these concepts.

Show evidence of any additional study and practice you did to master these concepts.

Complete one of the [Test Your Knowledge](#) activities

Show your process for this and highlight any realisations you gain.

```

1 #include <iostream>
2 #include <string>
3 #include <limits>
4 #include <vector>
5
6 using namespace std;
7
8 // Enum for product categories
9 enum ProductCategory {
10    DAIRY,
11    BAKERY,
12    SNACK,
13    FROZEN
14 };
15
16 // Struct to represent a product
17 struct Product {
18     string name;
19     string brand;
20     ProductCategory category;
21     int stockLevel;
22 };
23
24 // Function to add a product to StorePOS
25 void add_product(Product& product) {
26     clear_input_buffer();
27     product.name = read_string("Enter product name: ");
28     product.brand = read_string("Enter product brand: ");
29
30     // Display category options and get user choice
31     write_line("Select product category:");
32     write_line("0. Dairy");
33     write_line("1. Bakery");
34     write_line("2. Snack");
35     write_line("3. Frozen");
36     int categoryChoice = read_integer("Enter category number (0-3): ");
37     product.category = static_cast<ProductCategory>(categoryChoice);
38
39     // Get valid stock level from user
40     product.stockLevel = read_integer("Enter product stock level (0-1000): ");
41     while (product.stockLevel < 0 || product.stockLevel > 1000) {
42         write_line("Invalid stock level. Please enter a value between 0 and 1000.");
43         product.stockLevel = read_integer("Enter product stock level (0-1000): ");
44     }
45 }
46
47 int main() {
48     add_product(Product());
49     return 0;
50 }

```

Brief Summary of Concepts

Concept	Key Idea / Concept
Pass by Value vs Pass by Reference	These are the terms that explain how data is passed to a parameter. Pass by value copies value to the parameter while pass by reference is like passing the variable itself to the parameter.
Pointers	It is a datatype built into the programming language
References	While in pointer we need to manually get the address and store it in the pointer in reference the compiler takes care of this for us.
Null	Indicates an invalid or non-binding value or associated with the value 0.
Segmentation Fault	if you attempt to dereference a pointer and perform an action that is not permitted at that location in memory, this will result in a segmentation fault.
Dangling Pointers	Dangling pointer is a pointer that does not point to a valid value.

Reflection

What gives you confidence you have achieved the learning goals?

Use pass-by reference to accept and update values

I used the pass by reference to accept and update values in the program shown above.

*Use **const (constant)** references to provide read-only access to data to improve performance.*

Read through the programmers field guide and used it in the small program.

Use pointers to refer to and interact with other values in memory.

Covered.

What is the most important thing you learned from this and why?

Answer this question here. A few sentences to a paragraph should be sufficient here.

I learnt more about including different headers and used enum and struct commands more efficiently .