References

An alternative name for an object

Concept of References

- A reference (variable) is a name that acts as an alias, or an alternative name, for a previously defined variable
- References provide a convenient alternative to pointers (more on those later) when working with large data structures
- Unlike pointers, no arithmetic manipulation

Declaring a reference variable

- You can declare and define a reference variable like this:

```
int playerHealth = 100;
int &myReference = playerHealth;  //Create a reference
to an int
```

- There are a few rules on declaring references.
  - You cannot declare a reference and later assign it with a value.

```
int playerHealth = 100;
int &myReference;

myReference = playerHealth; //Does not work
```

Declaring a reference - Rules

- You also cannot assign an address to a reference

```
int playerHealth = 100;
int &myReference = &playerHealth;  //Does not work, must be
an object
```

- Structs or class references are treated like an object

```
struct ComplexObject
{
    //Data here
};

ComplexObject anObject;
ComplexObject referenceToComplexObject = anObject; //Works
```

Declaring a reference - Rules

- Once set you cannot change what a reference refers to:

```cpp
int testVar1 = 1; // create an int
int testVar2 = 2; //and another int
int& intRef = testVar1; //create a reference to the first int
intRef = testVar2; //this doesn't change what the reference points to!
//it changes the value of testVar1 to 2...
//intRef will always refer to testVar1
```

Pass By Value

- Typically, when you pass in an object into a function, a copy is created.
- Modifying the copy does not modify the original!

```
void PassByValue(int);

//Main function
void main()
{
    int myValue = 10;

    PassByValue(myValue);

    cout << myValue << endl;

    system("pause");
}

void PassByValue(int valueToModify)
{
    valueToModify *= 2; //Does not modify the variable being passed in!
}
```

References and Functions

- References are great as function parameters
- The use of references and pointers as function arguments is called "pass by reference"
  - If the parameter type is a struct or class, the memory used is only 4 bytes as opposed to much larger usage
  - Can modify the argument (consider using const)

Pass By Reference Example

```cpp
void PassByReference(int&);

//Main function
void main()
{
    int myValue = 10;

    PassByReference(myValue);

    cout << myValue << endl;

    system("pause");
}

void PassByReference(int &valueToModify)
{
    valueToModify *= 2; //Modifies the value being passed in
}
```

Returning a reference

- Functions can also return a reference
- However, the reference must NOT be a local variable

```
int& DoubleNumber(int number);

//Main function
void main()
{
    int myValue = 10;

    cout << DoubleNumber(myValue) << endl;

    system("pause");
}

int& DoubleNumber(int number)
{
    int temp;

    temp = number * 2;

    return temp; //Works and compiles, but unsafe
}
```

Canberra Institute of Technology

aie

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- References are great for passing variables into functions.
- Some big performance optimizations can be achieved just by doing this
- Be careful as you can edit them – use const to avoid this.
- Pointers do a similar thing, we’ll talk them later…