GROUP FIVE PRESENTATION

By writing a program:

a)Demonstrate how case sensitivity and special words affect naming

Types of cases

Camel case

Lowercase

Uppercase

Snakecase

C case sensitive language hence the name rose, Rose and ROSE are viewed as different values.

Program example

#include<stdio.h>

Int main(){

Int sum,a,b;

Sum=a+b;

}

The above code would not work since the compiler interprets sum and Sum as two different things.

In languages such as FORTRAN where there is no case sensitivity they are interpreted as the same thing.

Use of special words

Special words can either be keywords or a reserved word.

A keyword is a word that is special only in certain contexts.

A reserved word is a special word that cannot be used as a user-defined name. As a language design choice, reserved words are better than keywords. Ex: Fortran

Special words to some extent limit the scope to which one can create variable names.

b)Demonstrate how to associate a variable with an address.

Aliasing refers to the situation where different variables point to the same memory location.

It can be done through use of pointers, variable referencing and c/c++ unions.

Example: \*reference: geeks for geeks; strict aliasing\*

#include <stdio.h>

Void fun( int \*a, int\*b){

\*b=\*b+10;

return \*a;

}

int main(){

int data= 20;

int result= fun(&data,&data);

printf(“%d”,result);

}

**Output=30**

**c)Explain datatypes and their values**

In programming, data types define the kind of data that can be stored and manipulated by a computer program. Each data type has a specific set of values and operations associated with it. The values that can be assigned to a particular data type are constrained by its definition. Here are some common data types and their values:

Integer: Represents whole numbers without a fractional component. Examples of integer values are -3, 0, 42, and 1000.

Floating-point: Represents numbers with a fractional component. It includes decimal numbers. Examples of floating-point values are -3.14, 0.5, 3.14159, and 2.0.

String: Represents a sequence of characters. Any combination of letters, digits, symbols, or spaces enclosed in quotes (single or double) can be a string value. For example, "Hello, World!", "123", and “Nguthiru”

Boolean: Represents either true or false. Boolean values are used for logical operations and conditional statements. The two possible boolean values are true and false.

In most programming languages, including C, Java, Python, and JavaScript, the boolean data type is built-in and has the values true and false. These values are typically used for logical operations and conditional expressions. Here's an example of using booleans in C;

#include <stdio.h>

#include <stdbool.h> // Required for using bool type in C

int main() {

bool isTrue = true;

bool isFalse = false;

if (isTrue) {

printf("This statement is true.\n");

} else {

printf("This statement is false.\n");

}

if (isFalse) {

printf("This statement is true.\n");

} else {

printf("This statement is false.\n");

}

return 0;

}

In the above example, we define two variables isTrue and isFalse of type bool and assign them the boolean values true and false, respectively. We then use these variables in conditional statements (if statements) to control the flow of the program based on the truth value of the condition.

Boolean values are also the result of logical operations and comparisons. For example:

bool result1 = (10 > 5); // true

bool result2 = (5 == 7); // false

bool result3 = (true && false); // false

bool result4 = (true || false); // true

In the above code snippets, we assign the result of logical comparisons and operations to boolean variables, based on the evaluated conditions.

Booleans are fundamental in decision-making and controlling program flow, enabling you to execute specific blocks of code based on conditions that evaluate to true or false.

Character: Represents a single character, such as a letter, digit, or symbol. It can be enclosed in single quotes. For example, 'a', '7', or '+'.

Array: Represents a collection of elements of the same type. The elements in an array are accessed by their position or index. For example, [1, 2, 3, 4] is an array of integers.

Object: Represents a complex data structure that can store multiple values of different types. Objects are composed of key-value pairs, where each key is a string and each value can be of any data type. For example, { "name": "John", "age": 30, "isStudent": true } is an object with string, integer, and boolean values.

If you're looking for an example of an object data type in a program, you can consider using a language like JavaScript or Python, which natively support objects. Here's an example in JavaScript:

// Define a person object

var person = {

name: "John",

age: 30,

isStudent: true,

greet: function() {

console.log("Hello, my name is " + this.name + " and I'm " + this.age + " years old.");

}

};

// Access object properties

console.log(person.name); // Output: John

console.log(person.age); // Output: 30

console.log(person.isStudent); // Output: true

// Call object method

person.greet(); // Output: Hello, my name is John and I'm 30 years old.

In this example, we create an object called person with properties such as name, age, and isStudent, as well as a method greet() that prints a greeting message using the object's properties. We can access the properties using dot notation (object.property) and call the method using dot notation as well (object.method()).

It's important to note that the concept of objects and their behavior may vary depending on the programming language you're using. The example above demonstrates object usage in JavaScript. Other languages like Python have their own syntax and conventions for working with objects.

Null: Represents the absence of a value. It is often used to indicate the intentional absence of an object or value.

The null data type is commonly found in languages such as Java, C#, JavaScript, and others. Here's an example of using null in JavaScript:

var myVariable = null;

console.log(myVariable); // Output: null

In the example above, the variable myVariable is assigned the value of null. This indicates that the variable does not currently refer to any object or have any meaningful value.

It's important to note that null is distinct from undefined in JavaScript. While null represents an intentional absence of a value, undefined indicates that a variable has been declared but has not been assigned any value yet.

In some programming languages, null is also used to check if an object or variable is uninitialized or has not been assigned a value. However, it's crucial to handle null values properly in your code to avoid potential null reference errors or unexpected behavior when attempting to access properties or perform operations on null values.