Laporan Tugas Pemrograman Dasar *User-defined function*

Cornelius Arden Satwika Hermawan
22/505482/TK/55313

1. Mark the following statements as true or false:

Statement	Answer	Explanation
To use a predefined function in a program, you need to know only the name of the function and how to use it.	**True**	You primarily need the name and the signature (parameters, return type, and what it does) but not the internal implementation.
A value-returning function returns only one value.	**True**	In most basic programming languages (like C++, Java, Python, etc.), a function can have only **one** return type and, consequently, returns **one** primary value using the `return` statement. (Complex types like structs or objects can *contain* many values, but the function still returns only *one instance* of that complex type.)
Parameters allow you to use different values each time the function is called.	**True**	Parameters are placeholders for values passed to the function, enabling it to operate on different data with each call.
When a `return` statement executes in a user-defined function, the function immediately exits.	**True**	The `return` statement terminates the function execution and returns control (and potentially a value) to the calling code.

A value-returning function returns only integer values.	**False**	A value-returning function can return any valid data type, such as `int`, `double` (floating-point), `char`, `bool`, or a custom class/struct.
A function that changes the value of a **reference parameter** also changes the value of the actual parameter.	**True**	A reference parameter (like C++'s `&`) acts as an **alias** for the actual parameter. Changes to the reference parameter directly modify the original variable.
A variable name cannot be passed to a **value parameter**.	**False**	A variable name is very commonly passed to a value parameter. The *value* stored in that variable is copied into the value parameter.
If a C++ function does not use parameters, parentheses around the empty parameter list are still required.	**True**	Even with no parameters, the function call and the function definition/prototype still need the parentheses, e.g., `void myFunction()` or `myFunction();`.
In C++, the names of the corresponding formal and actual parameters must be the same.	**False**	The **formal parameters** (in the function definition) and the **actual parameters** (in the function call) are typically different variable names, although they can coincidentally be the same. Only their **order** and **data types** must match.
Whenever the value of a **reference parameter** changes, the value of the **actual parameter** changes.	**True**	This is the defining characteristic of a reference parameter: it provides direct access to the actual argument, so changes are synchronized.

In C++, function definitions can be nested; that is, the definition of one function can be enclosed in the body of another function.	**False**	C++ does not allow the nesting of complete function definitions (i.e., defining a function inside another function's body). The only exception is a *lambda function* or a local class containing a function, but complete traditional function definitions must stand alone.
Using **global variables** in a program is a better programming style than using **local variables**, because extra variables can be avoided.	**False**	Using **local variables** and passing data via parameters is generally considered **much better style** because it reduces side effects, makes code easier to test, and limits the scope of variables, which is key to avoiding unintended modifications. This is often called **minimizing scope**.
In a program, **global constants** are as dangerous as **global variables**.	**False**	**Global constants** are generally safe and often good practice (e.g., for math constants like PI), as their value cannot change, eliminating a major source of bugs (unintended modification) associated with global *variables*.
The memory for a **static variable** remains allocated between function calls.	**True**	The static keyword ensures the variable is initialized only once and retains its value throughout the program's execution, even after the function it's defined in has finished.

2. Soal Chapter 6, Programming Exercise No. 8.

8. The following formula gives the distance between two points, (x_1, y_1) and (x_2, y_2) in the Cartesian plane:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Given the center and a point on the circle, you can use this formula to find the radius of the circle. Write a program that prompts the user to enter the center and a point on the circle. The program should then output the circle's radius, diameter, circumference, and area. Your program must have at least the following functions:

a. `distance`: This function takes as its parameters four numbers that represent two points in the plane and returns the distance between them.

b. `radius`: This function takes as its parameters four numbers that represent the center and a point on the circle, calls the function `distance` to find the radius of the circle, and returns the circle's radius.

c. `circumference`: This function takes as its parameter a number that represents the radius of the circle and returns the circle's circumference. (If r is the radius, the circumference is $2\pi r$.)

d. `area`: This function takes as its parameter a number that represents the radius of the circle and returns the circle's area. (If r is the radius, the area is πr^2.)

Assume that $\pi = 3.1416$.

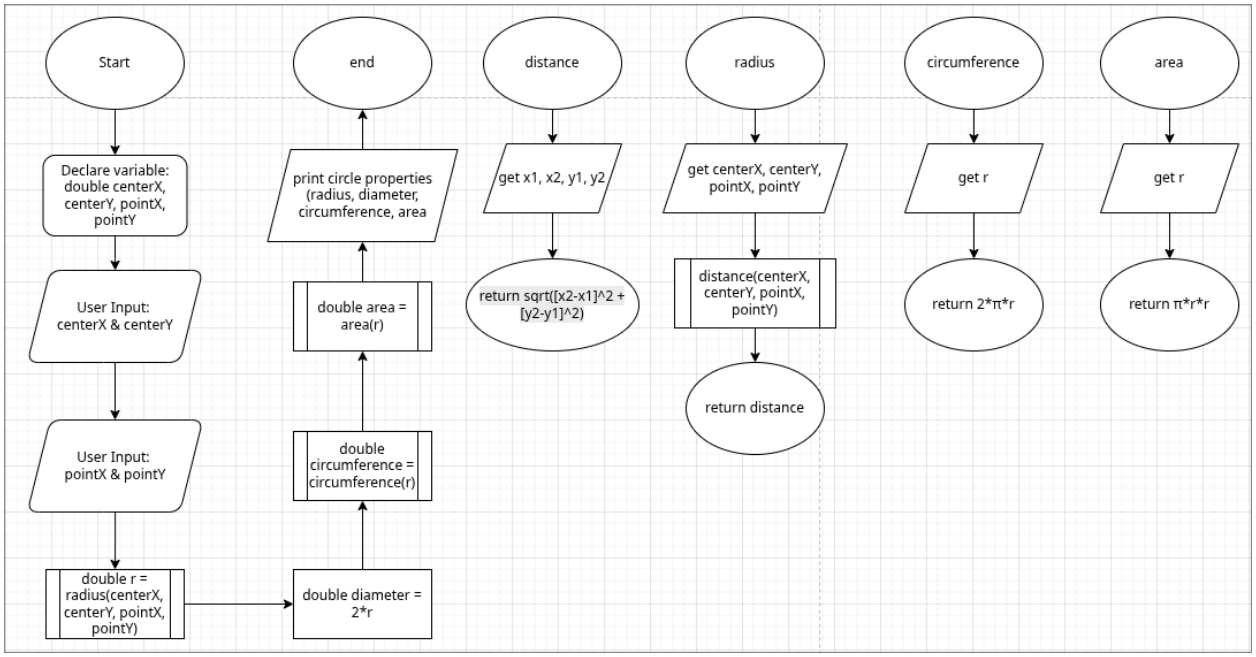
Dari soal di atas, program diharapkan mampu menerima input dari user berupa koordinat tengah sebuah lingkaran dan input koordinat titik yang berada di garis lingkaran. Dengan input tersebut, program diharapkan dapat menghitung jari-jari, diameter, keliling, dan luas lingkaran. Dalam melakukan perhitungan digunakan dasar rumus sebagai berikut:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Kode: https://github.com/arden1601/progdas/tree/main/tugas_user-define

```cpp
#include <iostream>
#include <cmath>
using namespace std;

// Function prototypes (signatures)
double distance(double x1, double y1, double x2, double y2);
double radius(double centerX, double centerY, double pointX, double pointY);
double circumference(double r);
double area(double r);

int main() {
    double centerX, centerY, pointX, pointY;

    // User input for center coordinates
    cout << "Enter the center coordinates (x, y): ";
    cin >> centerX >> centerY;

    // User input for point on circle coordinates
    cout << "Enter a point on the circle (x, y): ";
    cin >> pointX >> pointY;

    // Calculate radius
    double r = radius(centerX, centerY, pointX, pointY);

    double diameter = 2 * r;
    double circ = circumference(r);
    double a = area(r);

    cout << "\nCircle Properties:" << endl;
    cout << "Radius: " << r << endl;
    cout << "Diameter: " << diameter << endl;
    cout << "Circumference: " << circ << endl;
    cout << "Area: " << a << endl;

    return 0;
}

// Function to calculate distance between two points
double distance(double x1, double y1, double x2, double y2) {
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
}

// Function to calculate radius of circle
double radius(double centerX, double centerY, double pointX, double pointY) {
    return distance(centerX, centerY, pointX, pointY);
}

// Function to calculate circumference of circle
double circumference(double r) {
    const double PI = 3.1416;
    return 2 * PI * r;
}

// Function to calculate area of circle
double area(double r) {
    const double PI = 3.1416;
    return PI * r * r;
}
```

Flowchart:



Cara kerja:

Program ini menghitung dan menampilkan properti dari lingkaran berdasarkan input user. Program dapat dibagi menjadi 4 fase:

1. Input
    a. User memasukkan koordinat titik tengah dan disimpan dalam variable centerX dan center Y
    b. User memasukkan koordinat titik permukaan lingkaran dan disimpan dalam variable pointX dan pointY
2. Kalkulasi
    a. Kalkulasi Radius (Jari-jari)
    Dilakukan pemanggilan fungsi radius dengan parameter centerX, centerY, pointX, pointY. Fungsi me-*return* nilai radius dan disimpan dalam variabel r.
    b. Kalkulasi Diameter
    Variabel r dikalikan dengan 2 dan disimpan dalam variabel diameter
    c. Kalkulasi Keliling
    Dilakukan pemanggilan fungsi circumference dengan parameter r. Fungsi me-*return* nilai keliling dan disimpan dalam variable circ.
    d. Kalkulasi area
    Dilakukan pemanggilan fungsi area dengan parameter r. Fungsi me-*return* nilai keliling dan disimpan dalam variabel area.
3. Output
    Print variabel r, diameter, circ, dan area untuk menampilkan properti lingkaran
4. Terminate
    Program diakhiri dengan *return 0*.

```
double distance(double x1, double y1, double x2, double y2) {
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
}
```

➔ Formal Parameter: parameter yang didefinisikan ketika fungsi dibuat. Pada gambar di atas, x1, x2, y1, y2 merupakan parameter formal

```
double circ = circumference(r);
```

➔ Actual Parameter: parameter yang diteruskan ketika fungsi dipanggil. Pada gambar di atas, r merupakan parameter aktual.

➔ Function Signatures:

- ◆ distance(x1, y1, x2, y2)
  - Purpose: Calculates Euclidean distance between two points
  - Parameters: Coordinates of two points (x_1, y_1) and (x_2, y_2)
  - Returns: Distance as a double
  - Formula: $\sqrt{[(x_2-x_1)^2 + (y_2-y_1)^2]}$
- ◆ radius(centerX, centerY, pointX, pointY)
  - Purpose: Calculates radius of circle
  - Parameters: Center coordinates and point on circle coordinates
  - Returns: Radius as a double
  - Implementation: Calls distance() function
- ◆ circumference(r)
  - Purpose: Calculates circumference of circle
  - Parameters: Radius ®
  - Returns: Circumference as a double
  - Formula: $2 \times \pi \times r$
- ◆ area(r)
  - Purpose: Calculates area of circle
  - Parameters: Radius ®
  - Returns: Area as a double
  - Formula: $\pi \times r^2$