# DEPARTMENT OF COMPUTER ENGINEERING

| | |
|---|---|
| Semester | T.E. Semester V – Computer Engineering |
| Subject | Software Engineering |
| Subject Professor In-charge | Dr. Sachin Bojewar |
| Assisting Teachers | Prof. Sneha Annappanavar |
| Laboratory | M313B |

| | |
|---|---|
| Student Name | Deep Salunkhe |
| Roll Number | 21102A0014 |
| TE Division | A |

**Title: Calculation of Function Point (FP)**

**Explanation:**
Functional Point (FP) Analysis is a structured technique for measuring the functionality delivered by software systems. It is primarily used for software estimation, project management, and controlling software development and maintenance costs. Here is a brief overview of Functional Point Analysis:

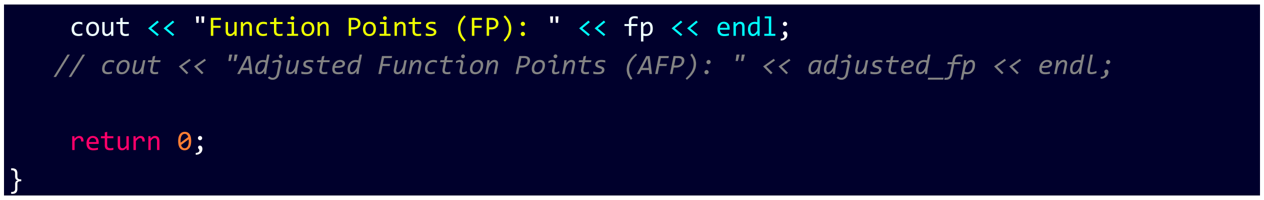
1. Objective: FP Analysis aims to quantify the functionality provided by a software application from the user's perspective. It doesn't focus on how the functionality is implemented but rather on what it does for the user.
2. Measurement Units: The fundamental measurement unit in FP Analysis is the Functional Point (FP). FP is a unit of measure for software functional size. It represents the sum of all the user interactions or functions provided by the software.
3. Components of Functional Points:
    - External Inputs (EI): These are user-initiated data inputs that the software processes. For example, user registration in a web application.
    - External Outputs (EO): These are user-initiated data outputs generated by the software. For instance, generating a report in a business application.
    - External Inquiries (EQ): These are user-initiated queries for information from the software. For example, searching for a product on an e-commerce website.
    - Internal Logical Files (ILF): These are data maintained by the software and used within the software. For example, a database of customer information.
    - External Interface Files (EIF): These are data maintained by external applications and referenced by the software.
4. Complexity and Weights: Each of the components (EI, EO, EQ, ILF, and EIF) is assigned a complexity level, typically on a scale of 1 to 3, based on factors like data elements, processing logic, and user interactions. These complexity levels are then used to calculate the FP.
5. Benefits:
    - Provides a standardized way to measure software functionality.
    - Helps in estimating project effort and cost.
    - Aids in project planning and resource allocation.
    - Supports performance measurement and benchmarking.
6. Limitations:
    - FP Analysis can be complex and time-consuming.
    - It relies on subjective assessments of complexity.

- The accuracy of estimates depends on the quality of input data.
- It may not account for all aspects of software quality.

---

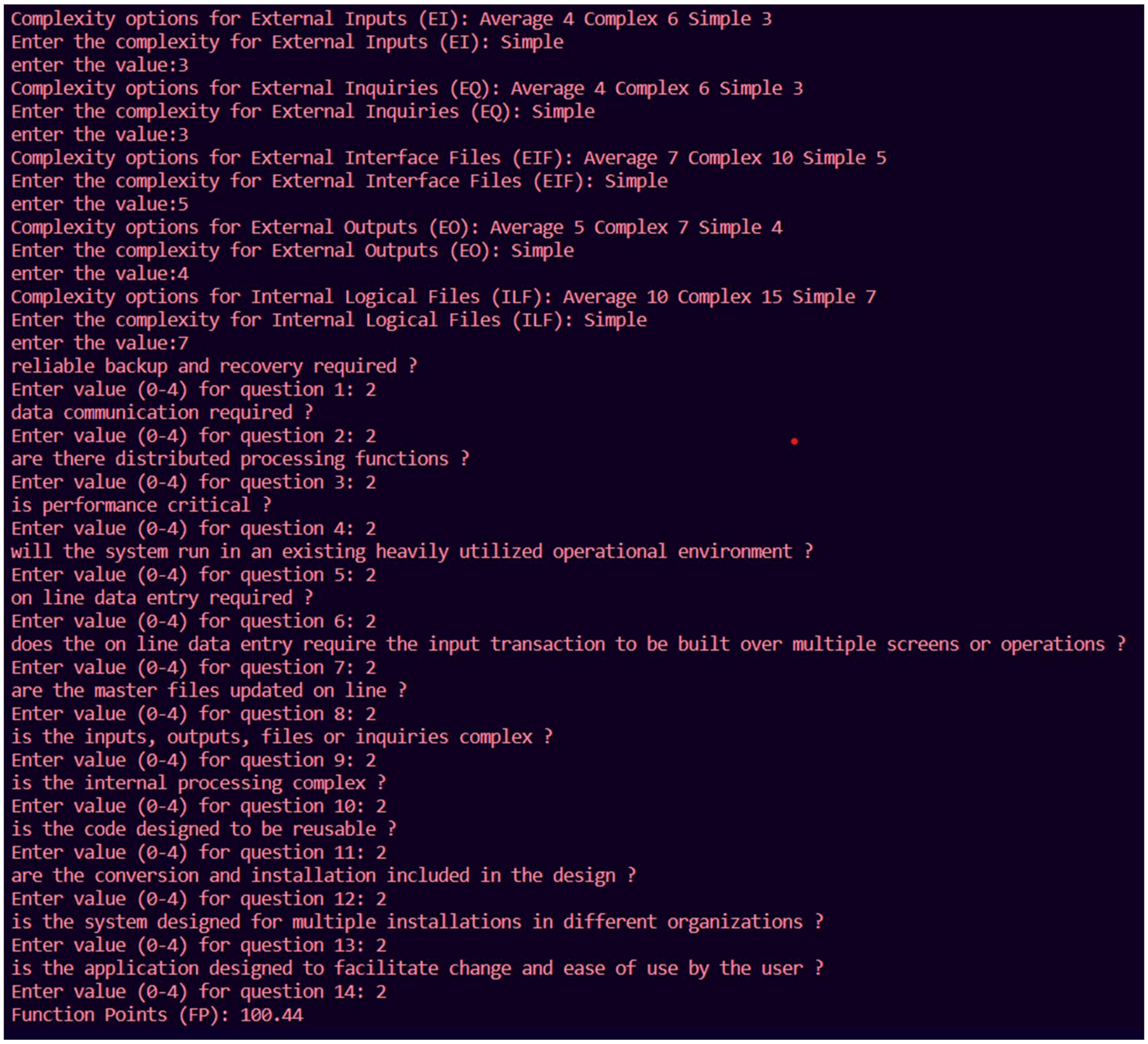
**Implementation:**

```
#include <iostream>
#include <map>
#include <vector>
using namespace std;

int main() {
    // Map to store Function Point types and their corresponding complexities
    map<string, map<string, int>> complexity_table;
    complexity_table["External Inputs (EI)"] = {{"Simple", 3}, {"Average", 4},
{"Complex", 6}};
    complexity_table["External Outputs (EO)"] = {{"Simple", 4}, {"Average", 5},
{"Complex", 7}};
    complexity_table["External Inquiries (EQ)"] = {{"Simple", 3}, {"Average", 4},
{"Complex", 6}};
    complexity_table["Internal Logical Files (ILF)"] = {{"Simple", 7},
{"Average", 10}, {"Complex", 15}};
    complexity_table["External Interface Files (EIF)"] = {{"Simple", 5},
{"Average", 7}, {"Complex", 10}};


    // 14 factors
    string aspects[14] = {
        "reliable backup and recovery required ?",
        "data communication required ?",
        "are there distributed processing functions ?",
        "is performance critical ?",
        "will the system run in an existing heavily utilized operational
environment ?",
        "on line data entry required ?",
        "does the on line data entry require the input transaction to be built
over multiple screens or operations ?",
        "are the master files updated on line ?",
        "is the inputs, outputs, files or inquiries complex ?",
        "is the internal processing complex ?",
        "is the code designed to be reusable ?",
        "are the conversion and installation included in the design ?",
        "is the system designed for multiple installations in different
organizations ?",
        "is the application designed to facilitate change and ease of use by the
user ?"
    };

    // Value Adjustment Factor (VAF)
    double sum = 0.0;

    // Get user input for complexities
    for (const auto& entry : complexity_table) {
        cout << "Complexity options for " << entry.first << ": ";
        for (const auto& complexity : entry.second) {
            cout << complexity.first << " " <<complexity.second<<" ";
        }
        cout << endl;

        string key;
        cout << "Enter the complexity for " << entry.first << ": ";
        cin >> key;
        int temp;
        cout<<"enter the value:";
        cin>>temp;

        // Use const_cast to temporarily remove const qualifier for the map value
        sum += (const_cast<map<string, int>&>(entry.second)[key])*temp;
    }

    // Get user input for 14 questions
    int addition = 0;

    for (int i = 1; i <= 14; ++i) {
        int value;
        cout<<aspects[i-1]<<endl;
        cout << "Enter value (0-4) for question " << i << ": ";
        cin >> value;
        addition += value;
    }



    double fp = sum * (0.65 + 0.01 * addition);
    //double adjusted_fp = fp * (1 + sum);

    // Display the results
```



```
    cout << "Function Points (FP): " << fp << endl;
   // cout << "Adjusted Function Points (AFP): " << adjusted_fp << endl;

    return 0;
}
```

**End Result:**



```
Complexity options for External Inputs (EI): Average 4 Complex 6 Simple 3
Enter the complexity for External Inputs (EI): Simple
enter the value:3
Complexity options for External Inquiries (EQ): Average 4 Complex 6 Simple 3
Enter the complexity for External Inquiries (EQ): Simple
enter the value:3
Complexity options for External Interface Files (EIF): Average 7 Complex 10 Simple 5
Enter the complexity for External Interface Files (EIF): Simple
enter the value:5
Complexity options for External Outputs (EO): Average 5 Complex 7 Simple 4
Enter the complexity for External Outputs (EO): Simple
enter the value:4
Complexity options for Internal Logical Files (ILF): Average 10 Complex 15 Simple 7
Enter the complexity for Internal Logical Files (ILF): Simple
enter the value:7
reliable backup and recovery required ?
Enter value (0-4) for question 1: 2
data communication required ?
Enter value (0-4) for question 2: 2
are there distributed processing functions ?
Enter value (0-4) for question 3: 2
is performance critical ?
Enter value (0-4) for question 4: 2
will the system run in an existing heavily utilized operational environment ?
Enter value (0-4) for question 5: 2
on line data entry required ?
Enter value (0-4) for question 6: 2
does the on line data entry require the input transaction to be built over multiple screens or operations ?
Enter value (0-4) for question 7: 2
are the master files updated on line ?
Enter value (0-4) for question 8: 2
is the inputs, outputs, files or inquiries complex ?
Enter value (0-4) for question 9: 2
is the internal processing complex ?
Enter value (0-4) for question 10: 2
is the code designed to be reusable ?
Enter value (0-4) for question 11: 2
are the conversion and installation included in the design ?
Enter value (0-4) for question 12: 2
is the system designed for multiple installations in different organizations ?
Enter value (0-4) for question 13: 2
is the application designed to facilitate change and ease of use by the user ?
Enter value (0-4) for question 14: 2
Function Points (FP): 100.44
```

**Conclusion:**

FP Analysis is a valuable tool in software project management and estimation, helping organizations plan and control software development efforts more effectively by quantifying the functionality delivered to users.