LAPORAN PEMROGRAMAN DASAR

TUGAS STRUCT DAN UNION

Disusun oleh:
Muhammad Fachrizal Giffari
22/504570/TK/55192

DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS GADJAH MADA
YOGYAKARTA

2025

1. Consider the following statements:

```
C/C++
struct nameType
{
    string first;
    string last;
}
```

```
C/C++
struct courseType
{
    string name;
    int callNum;
    int credits;
    char grade;
}
```

```
C/C++
struct studentType
{
    nameType name;
    double gpa;
    courseType course;
}
```

```
C/C++
studentType student;
studentType classList[100];
courseType course;
nameType name;
```

Mark the following statements as valid or invalid. If a statement is invalid, explain why.

```
C/C++
a.   student.course.callNum = "CSC230";
```

- **Invalid.** There is a type mismatch. `student.course.callNum` is defined as an `int` (integer), but `"CSC230"` is a string literal. You cannot assign a string to an integer.

```
C/C++
b.   cin >> student.name;
```

- **Invalid.** The `>>` operator (used with `cin`) is not automatically defined for an entire `struct` like `nameType`. You must read input into the *individual members* of the struct, like this:

`cin >> student.name.first >> student.name.last;`

```C/C++
c.   classList[0] = name;
```

- **Invalid.** This is a type mismatch. `classList[0]` is a variable of type `studentType`, but `name` is a variable of type `nameType`. You cannot assign a `nameType` struct directly to a `studentType` struct. (You could, however, assign it to the correct member: `classList[0].name = name;`)

```C/C++
d.   classList[1].gpa = 3.45;
```

- **Valid.** `classList[1]` refers to the second `studentType` object in the array. Its `.gpa` member is a `double`, and `3.45` is a valid `double` value.

```C/C++
e.   name = classList[15].name;
```

- **Valid.** The variable `name` is of type `nameType`. The member `classList[15].name` is also of type `nameType`. You can assign one struct to another struct of the **same type**.

```C/C++
f.   student.name = name;
```

- **Valid.** Both `student.name` and the variable `name` are of type `nameType`. This assignment is valid, just like in statement **e**.

```C/C++
g.   cout << classList[10] << endl;
```

- **Invalid.** Much like statement **b** with `cin`, the `<<` operator (used with `cout`) is not automatically defined for an entire `studentType` struct. You must print the *individual members* you want to see (e.g., `cout << classList[10].name.first;`).

```C/C++
h.   for (int j = 0; j < 100; j++)
            classList[j].name = name;
```

- **Valid.** This loop iterates through every `studentType` object in the `classList` array. In each iteration, it assigns the `nameType` variable `name` to the `name` member of the student (`classList[j].name`). This is a valid struct-to-struct assignment (as seen in **e** and **f**).

```C/C++
i.   classList.course.credits = 3;
```

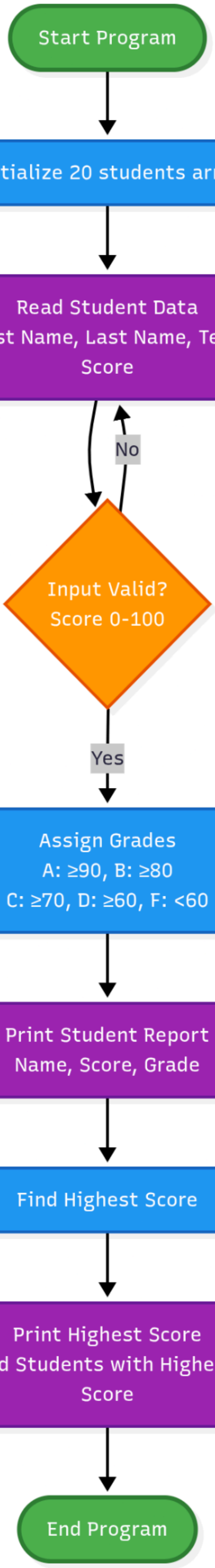
- **Invalid.** `classList` is an **array**, not a single `struct` instance. You cannot use the dot operator (`.`) directly on an array. You must first specify *which element* of the array you want to access using an index (e.g., `classList[0].course.credits = 3;`).

```C/C++
j.   course = studentType.course;
```

- **Invalid.** `studentType` is the *name of the struct type* (like a blueprint), not a variable. You can only use the dot operator (`.`) to access members of an *instance* (a variable) of that type, such as `student`. The valid statement would be `course = student.course;`.

2.

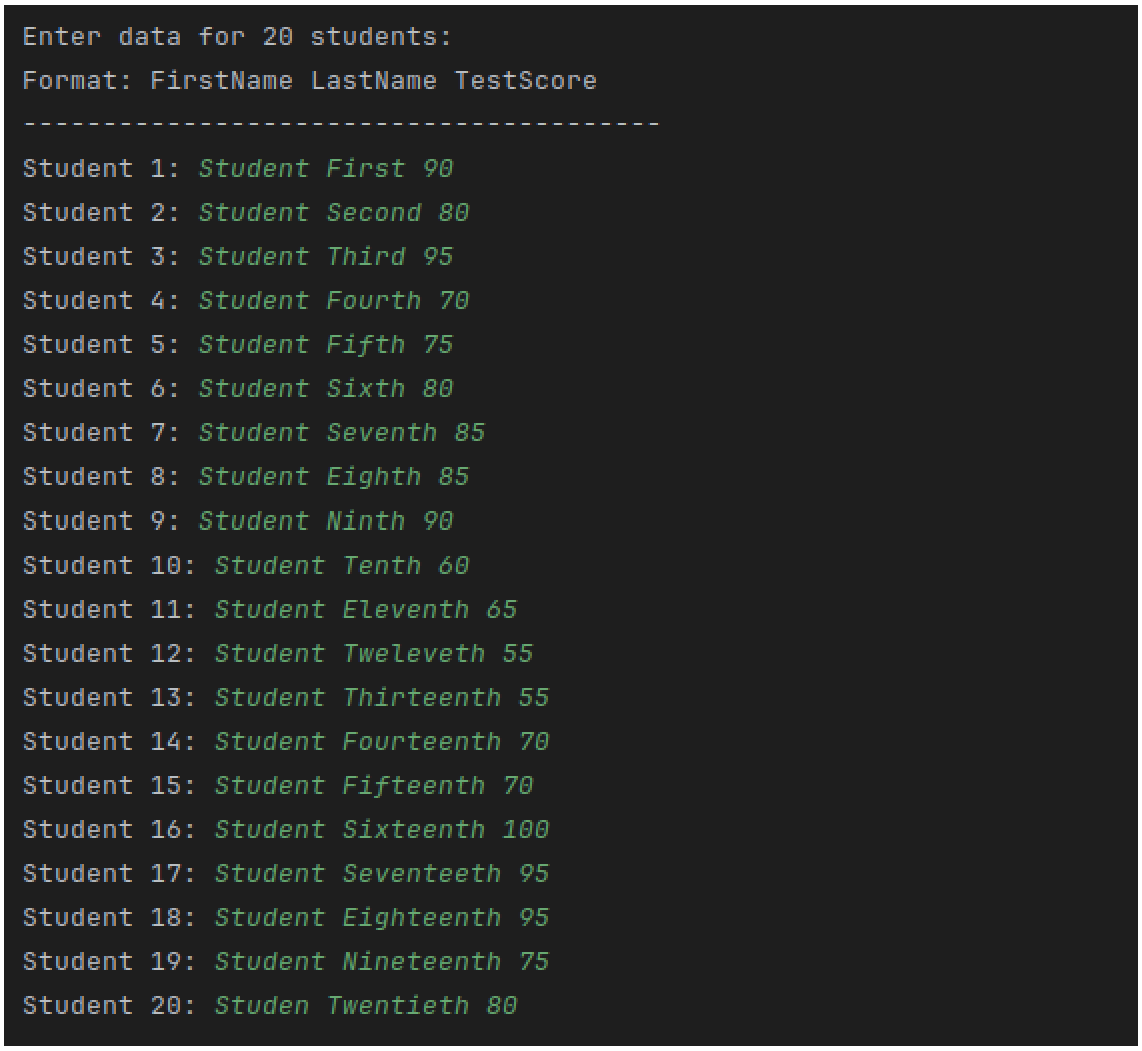
- Flowchart:


Start Program
Initialize 20 students array
Read Student Data
First Name, Last Name, Test Score
No
Input Valid?
Score 0-100
Yes
Assign Grades
A: ≥90, B: ≥80
C: ≥70, D: ≥60, F: <60
Print Student Report
Name, Score, Grade
Find Highest Score
Print Highest Score
and Students with Highest Score
End Program

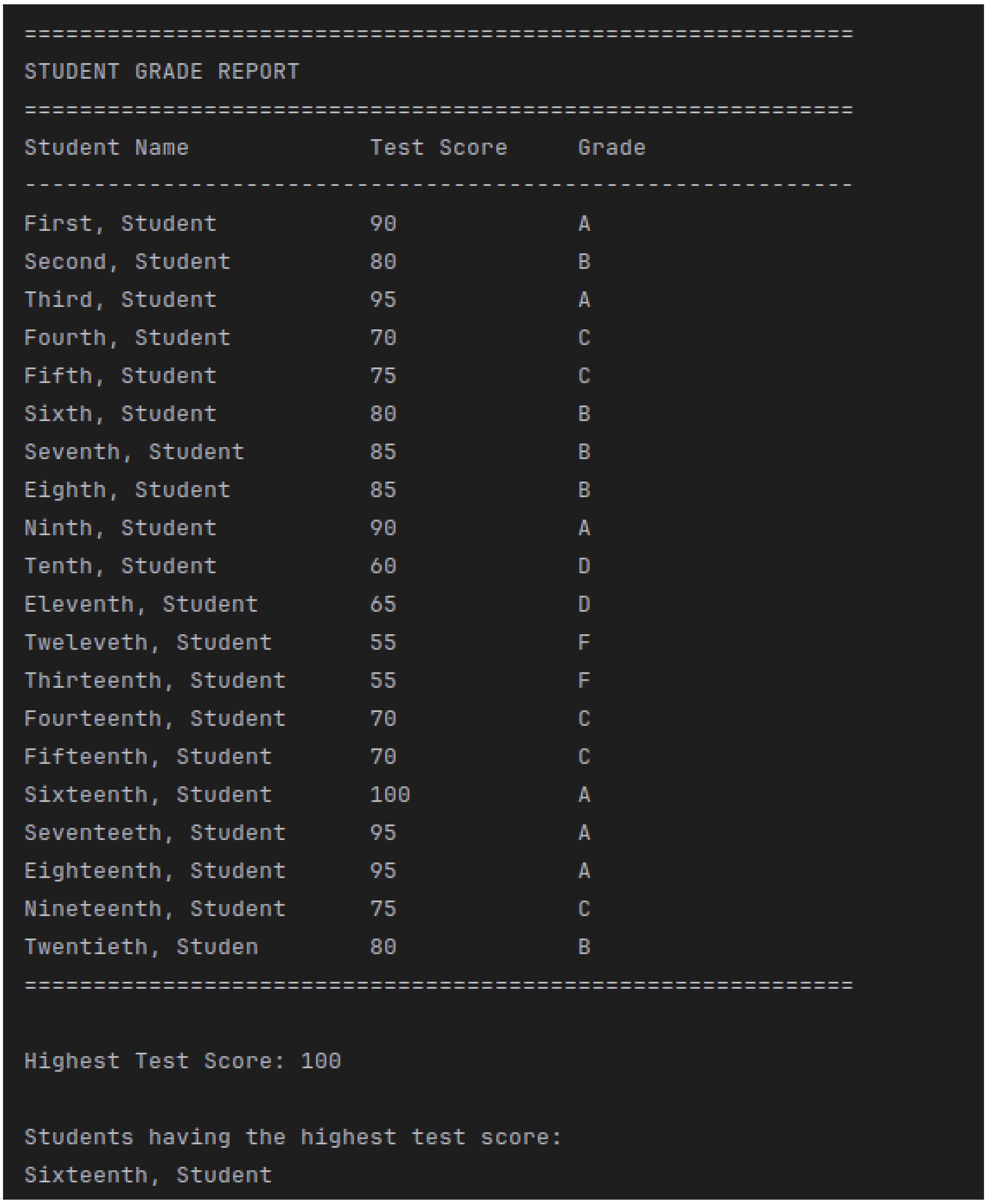
- Explanation:

  The code begins by **initializing an array** to hold data for 20 students. It then enters a process to **read student data**, including their name and test score, and immediately enters an **input validation loop** to ensure the score is within the valid 0-100 range. Once valid data is received, the program **assigns a letter grade** (A-F) based on the score and **prints a report** for that student. After this data handling (which is implicitly looped for all students), the program finds the **highest score** from the entire array, **prints that score** along with the name(s) of the student(s) who earned it, and then terminates.

- Screenshoot:



```
Enter data for 20 students:
Format: FirstName LastName TestScore
----------------------------------------
Student 1: Student First 90
Student 2: Student Second 80
Student 3: Student Third 95
Student 4: Student Fourth 70
Student 5: Student Fifth 75
Student 6: Student Sixth 80
Student 7: Student Seventh 85
Student 8: Student Eighth 85
Student 9: Student Ninth 90
Student 10: Student Tenth 60
Student 11: Student Eleventh 65
Student 12: Student Tweleveth 55
Student 13: Student Thirteenth 55
Student 14: Student Fourteenth 70
Student 15: Student Fifteenth 70
Student 16: Student Sixteenth 100
Student 17: Student Seventeeth 95
Student 18: Student Eighteenth 95
Student 19: Student Nineteenth 75
Student 20: Studen Twentieth 80
```



```
============================================================
STUDENT GRADE REPORT
============================================================
Student Name             Test Score     Grade
------------------------------------------------------------
First, Student           90             A
Second, Student          80             B
Third, Student           95             A
Fourth, Student          70             C
Fifth, Student           75             C
Sixth, Student           80             B
Seventh, Student         85             B
Eighth, Student          85             B
Ninth, Student           90             A
Tenth, Student           60             D
Eleventh, Student        65             D
Tweleveth, Student       55             F
Thirteenth, Student      55             F
Fourteenth, Student      70             C
Fifteenth, Student       70             C
Sixteenth, Student       100            A
Seventeeth, Student      95             A
Eighteenth, Student      95             A
Nineteenth, Student      75             C
Twentieth, Studen        80             B
============================================================


Highest Test Score: 100

Students having the highest test score:
Sixteenth, Student
```

- Code Repository
  github.com/mfachrizalg/tugas-struct-dan-union