4. Type Definitions (typedef)

4.1 What is typedef?

`typedef` creates aliases (alternative names) for existing data types. It makes code more readable and easier to maintain.

Basic Syntax:

```
typedef existing_type new_name;
```

4.2 typedef with Basic Types

```
// Create aliases for basic types
typedef int Integer;
typedef float Real;
typedef char Character;

// Usage
Integer age = 25;
Real temperature = 36.5;
Character grade = 'A';
```

4.3 typedef with Structures

Without typedef:

```
struct Student {
    int id;
    char name[50];
    float gpa;
};

// Must always use "struct Student"
struct Student s1;
struct Student students[100];
```

With typedef - Method 1:

```
struct Student {
    int id;
    char name[50];
    float gpa;
};

typedef struct Student Student;

// Now can use just "Student"
Student s1;
Student students[100];
```

With typedef - Method 2 (Combined):

```
typedef struct Student {
    int id;
    char name[50];
    float gpa;
} Student;

// Usage
Student s1;
Student students[100];
```

With typedef - Method 3 (Anonymous struct):

```
typedef struct {
    int id;
    char name[50];
    float gpa;
} Student;

// Usage
Student s1;
```

4.4 typedef with Enums

```
typedef enum {
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
} Day;

// Usage
Day today = WEDNESDAY;
Day weekend[2] = {SATURDAY, SUNDAY};
```

4.5 typedef with Pointers

```
typedef int* IntPtr;
typedef struct Student* StudentPtr;

// Usage
IntPtr p1, p2;  // Both are int pointers
StudentPtr sptr;  // Student pointer
```

4.6 typedef with Arrays

```
typedef int IntArray[10];
typedef char String[100];

// Usage
IntArray numbers;  // Same as: int numbers[10];
String name;       // Same as: char name[100];
```

4.7 Practical Example: Complex Data Types

```
#include <stdio.h>
#include <string.h>

// Define enumeration for course grades
```

```
typedef enum {
    GRADE_A = 4,
    GRADE_B = 3,
    GRADE_C = 2,
    GRADE_D = 1,
    GRADE_F = 0
} Grade;

// Define structure for a course
typedef struct {
    char code[10];
    char name[50];
    int credits;
    Grade grade;
} Course;

// Define structure for a student
typedef struct {
    int id;
    char name[50];
    Course courses[5];
    int numCourses;
    float gpa;
} Student;

// Function to calculate GPA
float calculateGPA(Student *s) {
    int totalCredits = 0;
    float totalPoints = 0.0;

    for (int i = 0; i < s->numCourses; i++) {
        totalCredits += s->courses[i].credits;
        totalPoints += s->courses[i].credits * s->courses[i].grade;
    }

    if (totalCredits == 0) return 0.0;
    return totalPoints / totalCredits;
}

int main() {
    Student student = {
```

```
        .id = 12345,
        .name = "Alice Johnson",
        .numCourses = 3
    };

    // Add courses
    strcpy(student.courses[0].code, "EE101");
    strcpy(student.courses[0].name, "Circuit Analysis");
    student.courses[0].credits = 3;
    student.courses[0].grade = GRADE_A;

    strcpy(student.courses[1].code, "MATH201");
    strcpy(student.courses[1].name, "Calculus II");
    student.courses[1].credits = 4;
    student.courses[1].grade = GRADE_B;

    strcpy(student.courses[2].code, "CS101");
    strcpy(student.courses[2].name, "Programming");
    student.courses[2].credits = 3;
    student.courses[2].grade = GRADE_A;

    // Calculate and display GPA
    student.gpa = calculateGPA(&student);

    printf("Student: %s (ID: %d)\n", student.name, student.id);
    printf("GPA: %.2f\n\n", student.gpa);

    printf("Courses:\n");
    for (int i = 0; i < student.numCourses; i++) {
        printf("  %s - %s (%d credits): Grade %d\n",
               student.courses[i].code,
               student.courses[i].name,
               student.courses[i].credits,
               student.courses[i].grade);
    }

    return 0;
}
```

Revision #1
Created 2025-10-05 14:45:27 UTC by DS
Updated 2025-10-05 14:45:43 UTC by DS