Lecture 7: Overview

- Structured Programming
  - Control flow charts
  - Structured Program Composition
  - Example Grade.c
  - Example Grade2.c
Structured Programming

- Control flow charts
- Graphical representation of program control flow
- Example:

  ![Flowchart Example]

  - Start
  - Input
  - Compute
  - Done?
  - Output
  - Finish
  - Loop
Structured Programming

- Sequential execution in C
  - Statement blocks: *Compound statements*
  - Sequence of statements grouped by braces: `{ }`
- Example:
  ```c
  { /* statement 1 */ /* statement 2 */ /* statement 3 */ /* ... */ /* statement n */
  }
  ```

Flow chart:

```
| Statement 1 | Statement 2 | Statement 3 | Statement n |
```
Structured Programming

- Sequential execution in C
  - Statement blocks: *Compound statements*
  - Sequence of statements grouped by braces: `{}`
- *Indentation* increases readability of the code
  - proper indentation is highly recommended!
- Example:

```c
/* some statements... */
if (x < 0) {
    printf("%d is negative!", x);
    /* handle negative values of x... */
    if (x < 100) {
        printf("%d is too small!", x);
        /* handle the problem... */
    } /* fi */
} /* fi */
if (x > 0) {
    printf("%d is positive!", x);
    /* handle positive values of x... */
} /* fi */
/* more statements... */
```
Structured Programming

- Sequential execution in C
  - Statement blocks: *Compound statements*
  - Sequence of statements grouped by braces: `{ }`
- *Indentation* increases readability of the code
  - proper indentation is highly recommended!
- Example:

```c
/* some statements... */
if (x < 0) {
    printf("%d is negative!", x);
    /* handle negative values of x... */
    if (x < 100) {
        printf("%d is too small!", x);
        /* handle the problem... */
        } /* fi */
    } /* fi */
if (x > 0) {
    printf("%d is positive!", x);
    /* handle positive values of x... */
    } /* fi */
/* more statements... */
```
Structured Programming

- Empty statement blocks
  - empty compound statement
  - does nothing (no operation, no-op)
- Example:

```c
{  /* nothing */
}
```

Flow chart:

- do nothing
Structured Programming

• Statement blocks
  • does whatever operation is in box
  • Example: 

```
x=x+1;
```
Structured Programming

• Selection: if statement
  • Flow chart:

• Example:

```c
if (grade >= 60)
{
    printf("You passed.");
}
/* fi */
```
Structured Programming

• Selection: if-else statement

• Flow chart:

• Example:

```c
if (grade >= 60)
    { printf("You passed.");
    } /* fi */
else
    { printf("You failed.");
    } /* esle */
```
Structured Programming

- Repetition: **while** loop
  - Flow chart:
  ```
  Condition?             Body
  ↓                   ↓
  false                true
  ```
  - Example:
    ```java
    int product = 2;
    while (product < 1000)
    {
        product *= 2;
    }
    /* elihw */
    ```
  - Note:
    - The condition is evaluated at the *beginning* of each loop!
Structured Programming

• Repetition: `break/continue` in `while` loop
• Flow chart:

  ![Flow chart diagram]

  - Condition?
    - true: Body
    - false: Continue

• Control flow:
  - `control flow with break`
  - `control flow with continue`

• Note:
  • The condition is evaluated at the *beginning* of each loop!
Structured Programming

- Repetition: **do-while** loop
  - Flow chart:
  - Example:
    ```
    int product = 2;
    do {
        product *= 2;
    } while (product < 1000);
    ```

- Note:
  - The condition is evaluated at the **end** of each loop!
Structured Programming

- Repetition: `break/continue` in `do-while` loop
- Flow chart:

```
Condition?

false
  \-- Body

true
  \-- Condition?
```

- Control flow:
  - `control flow with break`
  - `control flow with continue`

- Note:
  - The condition is evaluated at the *end* of each loop!
Structured Programming

• Repetition: **for** loop
  • Flow chart:
    
    ![Diagram of for loop flow chart]
    
    - Initialization
    - Condition?
      - true: Body
      - false: Increment
  
  • Example:

```c
for(i = 0; i < 10; i++)
    { printf("i = %d\n", i);
    } /* rof */
```

• Syntax:
  - **for**(initialization; condition; increment)
  - **body**
Structured Programming

- Repetition: `break/continue` in `for` loop
- Flow chart:

```
for(
  initialization;
  condition;
  increment)
body
```

- Control flow:
  - control flow with `break`
  - control flow with `continue`

- Syntax:
  - `for(initialization; condition; increment)`
  - `body`
Structured Programming

• Selection: `switch` statement

• Flow chart:

```
switch(LetterGrade)
{
    case 'A':
        { printf("Excellent!");
          break; }
    case 'B':
    case 'C':
    case 'D':
        { printf("Passed.");
          break; }
    case 'F':
        { printf("Failed!");
          break; }
    default:
        { printf("Invalid grade!");
          break; }
} /* hctiws */
```
Structured Programming

- **Selection:** `break` in `switch` statement

- **Flow chart:**

```c
switch(LetterGrade)
{
    case 'A':
        { printf("Excellent!");
        break; }
    case 'B':
    case 'C':
    case 'D':
        { printf("Passed.");
        break; }
    case 'F':
        { printf("Failed!");
        break; }
    default:
        { printf("Invalid grade!");
        break; }
}
/* hctiws */
```
Structured Program Composition

- Initial flow chart
  - Start
  - Program body
  - Finish
- Statement sequences
  - Statement blocks can be concatenated
  - Sequential execution
- Nested control structures
  - Control structures can be placed wherever statement blocks can be placed in the code
Structured Program Composition

- Example:
  - Initial flow chart
Structured Program Composition

- Example:
  - Sequential composition
Structured Program Composition

- Example:
  - insertion of another sequential statement
Structured Program Composition

- Example:
  - insertion of \texttt{if-else} statement
Structured Program Composition

- Example:
  - insertion of sequential statement
Structured Program Composition

- Example:
  - insertion of \texttt{if-else} statement
Structured Program Composition

- Example:
  - insertion of sequential statement
Structured Program Composition

- Example:
  - insertion of sequential statement (twice)
Structured Program Composition

• Example:
  • insertion of switch statement
  • etc. ...
• Grade calculation: Grade.c (part 1/3)

```c
/* Grade.c: convert score into letter grade */
/* author: Rainer Doemer */
/* modifications: */
/* 10/17/04 RD initial version */
#include <stdio.h>

/* main function */
int main(void){

    /* variable definitions */
    int score = 0;
    char grade;

    /* input section */
    while (score < 1 || score > 100){
        printf("Please enter your score (1-100): ");
        scanf("%d", &score);
    }/* elihw */

    ...
```
Example Program

- Grade calculation: Grade.c (part 2/3)

```c
/* computation section */
if (score >= 90)
    { grade = 'A'; }
else
    { if (score >= 80)
        { grade = 'B'; }
    else
    { if (score >= 70)
        { grade = 'C'; }
    else
    { if (score >= 60)
        { grade = 'D'; }
    else
    { grade = 'F'; }
    } /* else */
} /* else */
} /* else */
```
Example Program

- Grade calculation: Grade.c (part 3/3)

```c
...  
/* output section */
printf("Your letter grade is %c.\n", grade);  
/* exit */
return 0;
} /* end of main */
/* EOF */
```
Example Program

• Example session: Grade.c

```bash
% vi Grade.c
% gcc Grade.c -o Grade -Wall -ansi
% Grade
Please enter your score (1-100): 111
Your letter grade is A.
% Grade
Please enter your score (1-100): 85
Your letter grade is B.
% Grade
Please enter your score (1-100): 71
Your letter grade is C.
% Grade
Please enter your score (1-100): 69
Your letter grade is D.
% Grade
Please enter your score (1-100): 55
Your letter grade is F.
% 
```
Example Program

- Grade calculation: *Grade2.c* (part 1/3)

```c
/* Grade2.c: convert score into letter grade */
/* author: Rainer Doemer */
/* modifications: */
/* 10/18/04 RD use `switch` statement */
/* 10/17/04 RD initial version */

#include <stdio.h>

/* main function */
int main(void)
{
    /* variable definitions */
    int score = 0;
    char grade;

    /* input section */
    while (score < 1 || score > 100)
    {
        printf("Please enter your score (1-100): ");
        scanf("%d", &score);
    } /* elihw */
...
```
Example Program

- Grade calculation: Grade2.c (part 2/3)

```c
.../* computation section */
switch (score / 10)
    { case 10:
    case 9:
        { grade = 'A';
            break; }
    case 8:
        { grade = 'B';
            break; }
    case 7:
        { grade = 'C';
            break; }
    case 6:
        { grade = 'D';
            break; }
    default:
        { grade = 'F';
            break; }
} /* hctiws */
...
Example Program

- Grade calculation: Grade2.c (part 3/3)

```c
... 
/* output section */
printf("Your letter grade is %c.\n", grade);

/* exit */
return 0;
} /* end of main */

/* EOF */
```
Example Program

• Example session: Grade2.c

```bash
% cp Grade.c Grade2.c
% vi Grade2.c
% gcc Grade2.c -o Grade2 -Wall -ansi
% Grade2
Please enter your score (1-100): 111
Your letter grade is A.
% Grade2
Please enter your score (1-100): 99
Your letter grade is A.
% Grade2
Please enter your score (1-100): 85
Your letter grade is B.
% Grade2
Please enter your score (1-100): 71
Your letter grade is C.
% Grade2
Please enter your score (1-100): 69
Your letter grade is D.
% Grade2
Please enter your score (1-100): 55
Your letter grade is F.
%```