EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 8

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 8: Overview

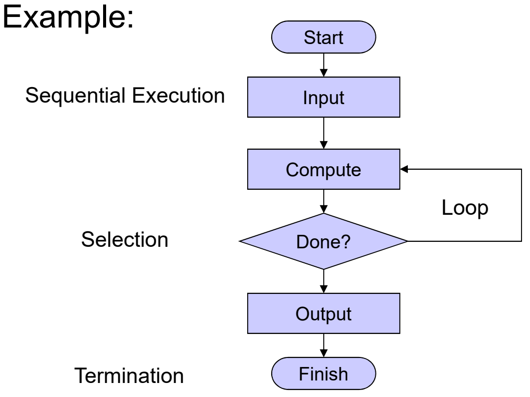
- Think before you program!
- Structured Programming
  - Sequential statements
  - Conditional statements
  - Repetition statements
    - **`while`** loop
    - **`do-while`** loop
    - **`for`** loop
- Arbitrary jump statements
    - **`goto`** statement
- Program Development and Debugging
  - Example **`Interest.c`**
  - Source-level debugger **`gdb`**
  - Example **`Interest2.c`**

Programming == Thinking

- Programming ...
  - ... is *not* a mechanic procedure!
  - ... requires *thinking*!
- Program ...
  - ... *writing* requires an *intelligent human being*!
  - ... *execution* can be performed by a *dumb machine*.
- General programming steps:
  1. Understand the problem
  2. Define the input and output data
  3. Develop the algorithm (and specify it in pseudo code)
  4. Define the control flow (e.g. use control flow charts)
  5. Write the program in programming language
  6. Compile, test and debug the program

Structured Programming

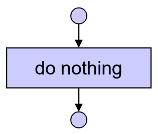
- Control flow charts
  - Graphical representation of program control flow
  - Example:



Structured Programming

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

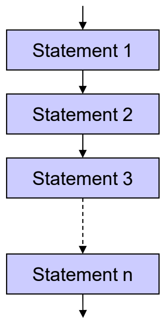
```
{
 /* nothing */
}
```



Structured Programming

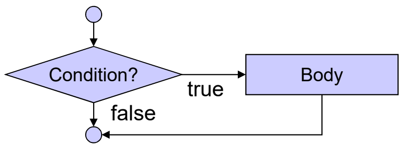
- Sequential execution in C
  - Statement blocks: *Compound statements*
  - Sequence of statements grouped by braces: **{ }**
- Example: Flow chart:

```
{
  /* statement 1 */

  /* statement 2 */

  /* statement 3 */

  /* ... */

  /* statement n */
}
```



Structured Programming

- Selection: **`if`** statement
  - Flow chart:

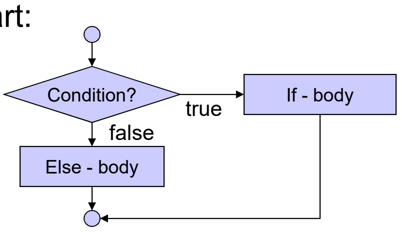


  - Example:

```
if (grade >= 60)
   { printf(“You passed.”);
    } /* fi */
```

Structured Programming

- Selection: **`if-else`** statement
  - Flow chart:

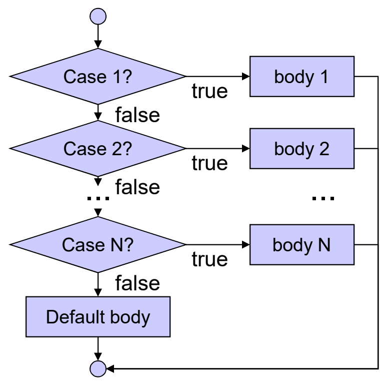


  - Example:

```
if (grade >= 60)
   { printf(“You passed.”);
    } /* fi */
else
   { printf(“You failed.”);
    } /* esle */
```

Structured Programming

- Selection: **`switch`** statement
  - Flow chart:

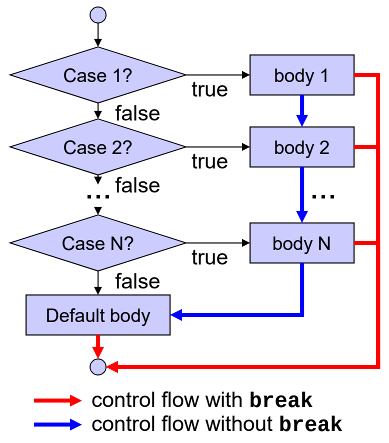


Example:

```
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:

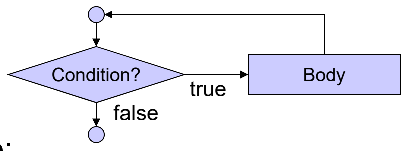


Example:

```
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

- Repetition: `while` loop
  - Flow chart:



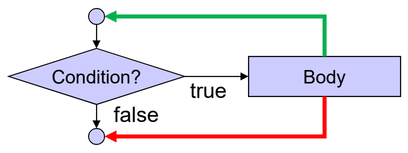
  - Example:

```
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:

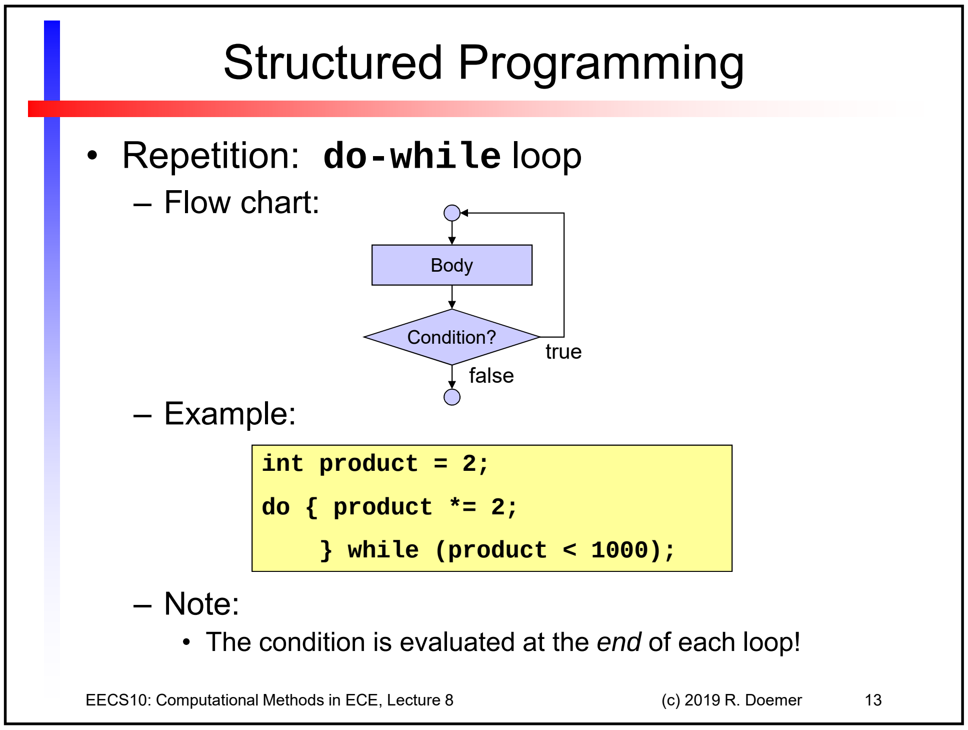


  - Control flow:

    → control flow with `break`

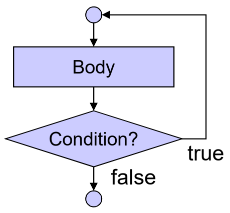
    → control flow with `continue`

  - Note:
    - The condition is evaluated at the *beginning* of each loop!



Structured Programming

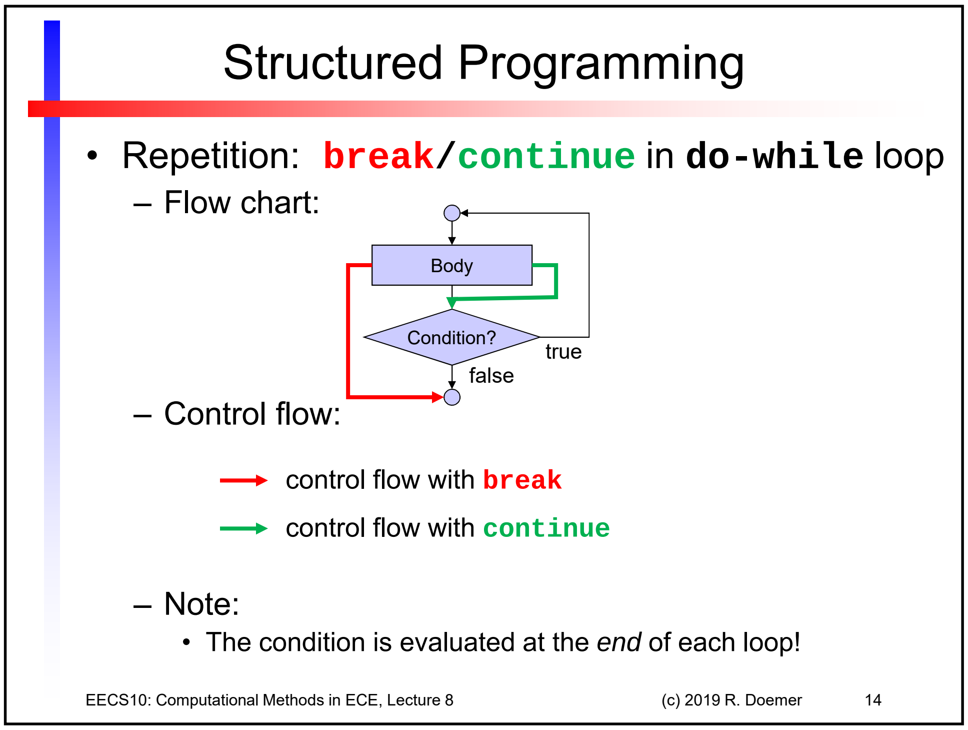
- Repetition: **`do-while`** loop
  - Flow chart:



  - Example:

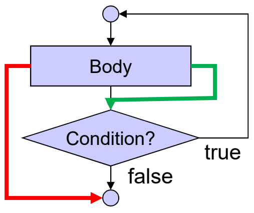
```
int product = 2;
do { product *= 2;
    } while (product < 1000);
```

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



Structured Programming

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



  - Control flow:

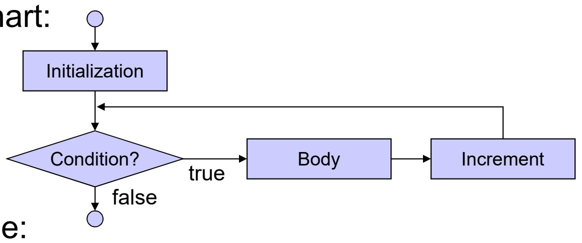
    → control flow with **`break`**

    → control flow with **`continue`**

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

Structured Programming

- Repetition: **`for`** loop
  - Flow chart:



  - Example:

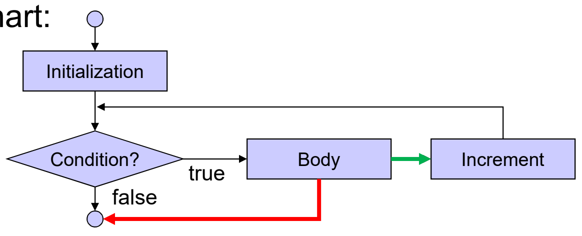
```
for(i = 0; i < 10; i++)
   { printf("i = %d\n", i);
    } /* rof */
```

  - Syntax:
    - **for(*initialization*; *condition*; *increment*)**
      { *body* }

EECS10: Computational Methods in ECE, Lecture 8   (c) 2019 R. Doemer   15

Structured Programming

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



  - Control flow:
    - → control flow with **`break`**
    - → control flow with **`continue`**
  - Syntax:
    - **for(*initialization*; *condition*; *increment*)**
      { *body* }

EECS10: Computational Methods in ECE, Lecture 8   (c) 2019 R. Doemer   16

Arbitrary Control Flow

- Arbitrary jumps: **`goto`** statement
  - **`goto`** statement jumps to the specified *labeled* statement (within the same function)
  - Example:

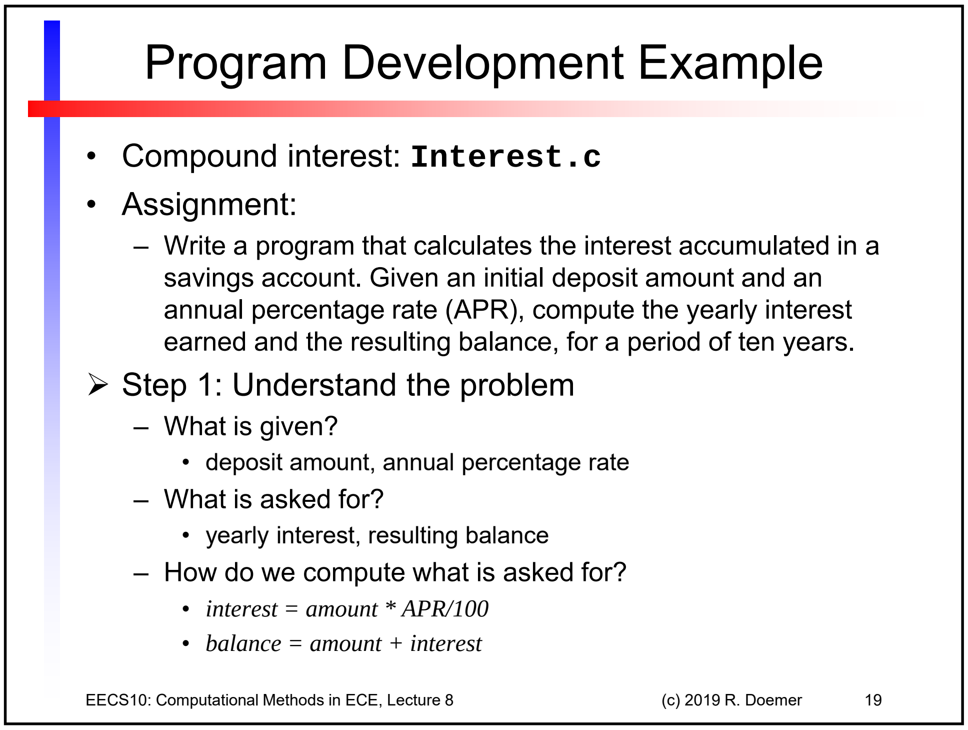
```
begin:  count = 0;
        goto next;
repeat: if (count > 100)
          { goto end; }
next:   count++;
        if (count == 77)
          { goto next; }
        goto repeat;
end:    printf("%d", count);
```

  - Warning:
    - **`goto`** statement allows *un-structured* programming!
    - **`goto`** statement should be avoided whenever possible!

Program Development Example

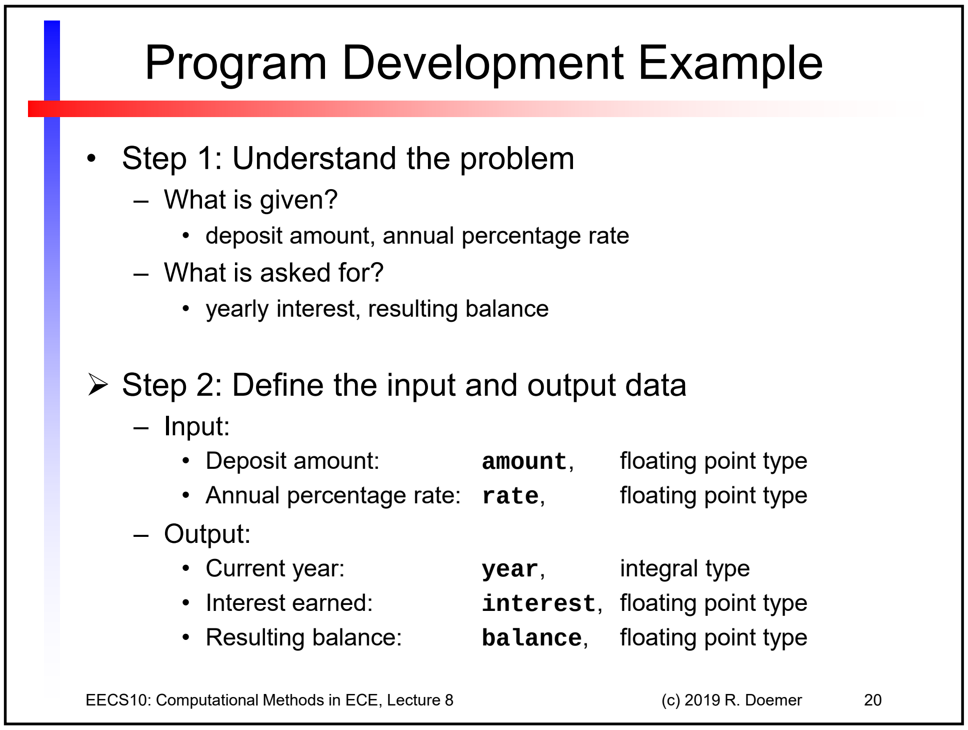
- Compound interest: **`Interest.c`**
- Assignment:
  - Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount and an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
  - For example, for $1000 in savings at 4.5% APR, the annual interest should be tabulated as follows:

```
Interest for year  1 is $   45.00, total balance is $ 1045.00.
Interest for year  2 is $   47.02, total balance is $ 1092.03.
Interest for year  3 is $   49.14, total balance is $ 1141.17.
...
```



Program Development Example

- Compound interest: **`Interest.c`**
- Assignment:
  - Write a program that calculates the interest accumulated in a savings account. Given an initial deposit amount and an annual percentage rate (APR), compute the yearly interest earned and the resulting balance, for a period of ten years.
- ➢ Step 1: Understand the problem
  - What is given?
    - deposit amount, annual percentage rate
  - What is asked for?
    - yearly interest, resulting balance
  - How do we compute what is asked for?
    - *interest = amount * APR/100*
    - *balance = amount + interest*



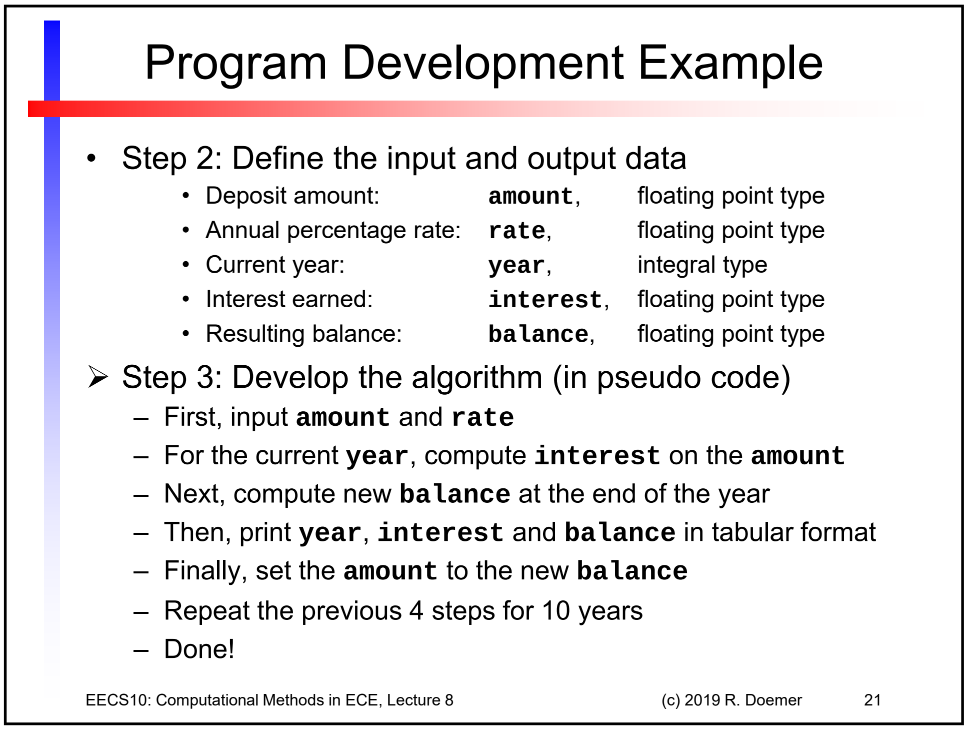
Program Development Example

- Step 1: Understand the problem
  - What is given?
    - deposit amount, annual percentage rate
  - What is asked for?
    - yearly interest, resulting balance
- ➢ Step 2: Define the input and output data
  - Input:

• Deposit amount:	**`amount`**,	floating point type
• Annual percentage rate:	**`rate`**,	floating point type

  - Output:

• Current year:	**`year`**,	integral type
• Interest earned:	**`interest`**,	floating point type
• Resulting balance:	**`balance`**,	floating point type

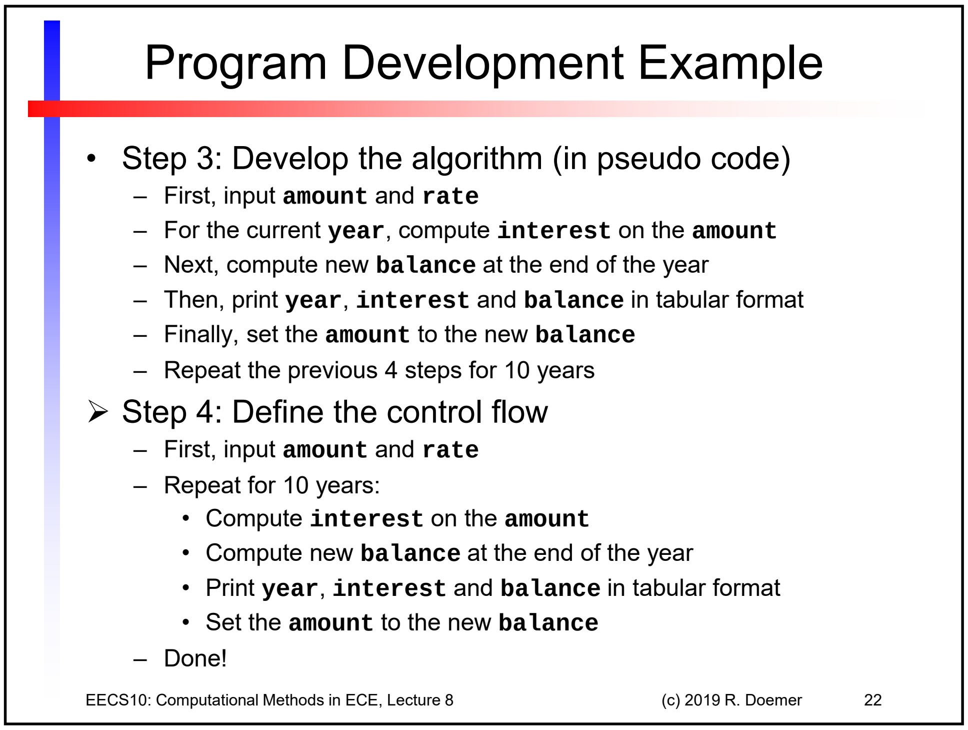


Program Development Example

- Step 2: Define the input and output data

• Deposit amount:	**`amount`**,	floating point type
• Annual percentage rate:	**`rate`**,	floating point type
• Current year:	**`year`**,	integral type
• Interest earned:	**`interest`**,	floating point type
• Resulting balance:	**`balance`**,	floating point type

➢ Step 3: Develop the algorithm (in pseudo code)
  - First, input **`amount`** and **`rate`**
  - For the current **`year`**, compute **`interest`** on the **`amount`**
  - Next, compute new **`balance`** at the end of the year
  - Then, print **`year`**, **`interest`** and **`balance`** in tabular format
  - Finally, set the **`amount`** to the new **`balance`**
  - Repeat the previous 4 steps for 10 years
  - Done!



Program Development Example

- Step 3: Develop the algorithm (in pseudo code)
  - First, input **`amount`** and **`rate`**
  - For the current **`year`**, compute **`interest`** on the **`amount`**
  - Next, compute new **`balance`** at the end of the year
  - Then, print **`year`**, **`interest`** and **`balance`** in tabular format
  - Finally, set the **`amount`** to the new **`balance`**
  - Repeat the previous 4 steps for 10 years

➢ Step 4: Define the control flow
  - First, input **`amount`** and **`rate`**
  - Repeat for 10 years:
    - Compute **`interest`** on the **`amount`**
    - Compute new **`balance`** at the end of the year
    - Print **`year`**, **`interest`** and **`balance`** in tabular format
    - Set the **`amount`** to the new **`balance`**
  - Done!

Program Development Example

- Step 4: Define the control flow
  - First, input **`amount`** and **`rate`**
  - Repeat for 10 years:
    - Compute **`interest`** on the **`amount`**
    - Compute new **`balance`** at the end of the year
    - Print **`year`**, **`interest`** and **`balance`** in tabular format
    - Set the **`amount`** to the new **`balance`**
- ➢ Step 5: Write the program in programming language

```
double amount;
```
```
double rate;
```
```
int year;
```
```
double interest;
```
```
double balance;
```
```
printf("Please enter the initial amount in $: ");
scanf("%lf", &amount);
```
```
printf("Please enter the interest rate in %% : ");
scanf("%lf", &rate);
```

etc.

Example Program

- Compound interest: **`Interest.c`** (part 1/2)

```
/* Interest.c: compound interest on savings account     */
/* author: Rainer Doemer                                */
/* modifications:                                       */
/* 10/18/06 RD  distinguish amount and balance          */
/* 10/19/04 RD  initial version                         */

#include <stdio.h>

/* main function */

int main(void)
{
   /* variable definitions */
   double amount, balance, rate, interest;
   int    year;

   /* input section */
   printf("Please enter the initial amount in $: ");
   scanf("%lf", &amount);
   printf("Please enter the interest rate in %% : ");
   scanf("%lf", &rate);
...
```

Example Program

- Compound interest: **`Interest.c`** (part 2/2)

```
...

   /* computation and output section */
   for(year = 1; year <= 10; year++)
      { interest = amount * (rate/100.0);
        balance = amount + interest;
        printf("Interest for year %2d is $%8.2f,"
                " total balance is $%8.2f.\n",
                year, interest, balance);
        amount = balance;
       } /* rof */

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

/* EOF */
```

EECS10: Computational Methods in ECE, Lecture 8 (c) 2019 R. Doemer 25

Program Development Example

- Step 5: Write the program in programming language
- ➢ Step 6: Compile, test (and debug) the program

```
% vi Interest.c
% gcc Interest.c -o Interest -Wall -ansi
% ./Interest
Please enter the initial amount in $: 1500
Please enter the interest rate in % : 1.5
Interest for year  1 is $   22.50, total balance is $ 1522.50.
Interest for year  2 is $   22.84, total balance is $ 1545.34.
Interest for year  3 is $   23.18, total balance is $ 1568.52.
Interest for year  4 is $   23.53, total balance is $ 1592.05.
Interest for year  5 is $   23.88, total balance is $ 1615.93.
Interest for year  6 is $   24.24, total balance is $ 1640.16.
Interest for year  7 is $   24.60, total balance is $ 1664.77.
Interest for year  8 is $   24.97, total balance is $ 1689.74.
Interest for year  9 is $   25.35, total balance is $ 1715.08.
Interest for year 10 is $   25.73, total balance is $ 1740.81.
%
```

EECS10: Computational Methods in ECE, Lecture 8 (c) 2019 R. Doemer 26

Debugging

- Source-level Debugger **gdb**
  - Debugger features
    - run the program under debugger control
    - follow the control flow of the program during execution
    - set breakpoints to stop execution at specified statements
    - inspect (and adjust) the values of variables
    - find the point in the program where the “crash” happens
  - Preparation:
    compile your program with debugging support on
    - Option **`-g`** tells compiler to add debugging information (symbol tables) to the generated executable file

➢ **`gcc Program.c -o Program -Wall -ansi -g`**

➢ **`gdb Program`**

Debugging

- Source-level Debugger **gdb**
  - Basic **gdb** commands
    - **`run`**
      - starts the execution of the program in the debugger
    - **`break`** ***`function_name`*** *(or* ***`line_number`****)*
      - inserts a breakpoint; program execution will stop at the breakpoint
    - **`cont`**
      - continues the execution of the program in the debugger
    - **`list`** ***`from_line_number,to_line_number`***
      - lists the current or specified range of line_numbers
    - **`print`** ***`variable_name`***
      - prints the current value of the variable ***`variable_name`***
    - **`next`**
      - executes the next statement (one statement at a time)
    - **`quit`**
      - exits the debugger (and terminates the program)
    - **`help`**
      - provides helpful details on debugger commands

Debugging Example

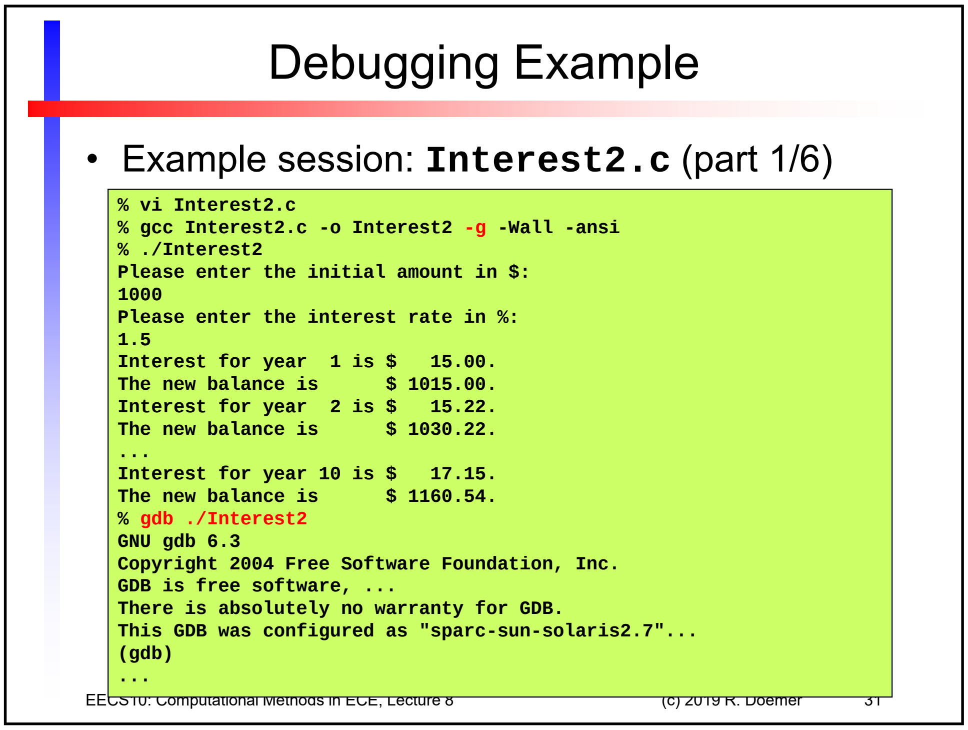
- Compound interest: **`Interest2.c`** (part 1/2)

```
/* Interest2.c: compound interest on savings account    */
/* author: Rainer Doemer                                */
/* modifications:                                       */
/* 10/23/05 RD  version to demonstrate debugging        */
/* 10/19/04 RD  initial version                         */

#include <stdio.h>

/* main function */

int main(void)
{
   /* variable definitions */
   double amount, balance, rate, interest;
   int    year;

   /* input section */
   printf("Please enter the initial amount in $:\n");
   scanf("%lf", &amount);
   printf("Please enter the interest rate in %%:\n");
   scanf("%lf", &rate);
...
```

Debugging Example

- Compound interest: **`Interest2.c`** (part 2/2)

```
...

/* computation and output section */
   for(year = 1; year <= 10; year++)
      { interest = amount * (rate/100.0);
        balance = amount + interest;
        printf("Interest for year%3d is $%8.2f.\n", year,
                                                interest);
        printf("The new balance is      $%8.2f.\n", balance);
        amount = balance;
       } /* rof */

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

/* EOF */
```

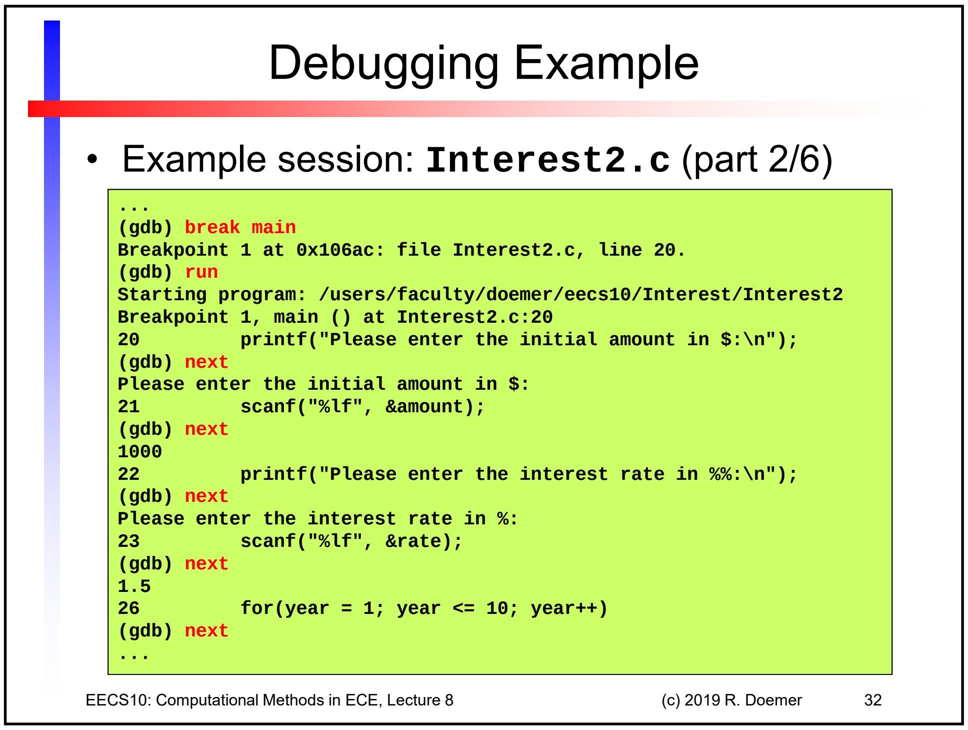


Debugging Example

- Example session: **`Interest2.c`** (part 1/6)

```
% vi Interest2.c
% gcc Interest2.c -o Interest2 -g -Wall -ansi
% ./Interest2
Please enter the initial amount in $:
1000
Please enter the interest rate in %:
1.5
Interest for year  1 is $   15.00.
The new balance is      $ 1015.00.
Interest for year  2 is $   15.22.
The new balance is      $ 1030.22.
...
Interest for year 10 is $   17.15.
The new balance is      $ 1160.54.
% gdb ./Interest2
GNU gdb 6.3
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, ...
There is absolutely no warranty for GDB.
This GDB was configured as "sparc-sun-solaris2.7"...
(gdb)
...
```

EECS10: Computational Methods in ECE, Lecture 8 (c) 2019 R. Doemer 31



Debugging Example

- Example session: **`Interest2.c`** (part 2/6)

```
...
(gdb) break main
Breakpoint 1 at 0x106ac: file Interest2.c, line 20.
(gdb) run
Starting program: /users/faculty/doemer/eecs10/Interest/Interest2
Breakpoint 1, main () at Interest2.c:20
20          printf("Please enter the initial amount in $:\n");
(gdb) next
Please enter the initial amount in $:
21          scanf("%lf", &amount);
(gdb) next
1000
22          printf("Please enter the interest rate in %%:\n");
(gdb) next
Please enter the interest rate in %:
23          scanf("%lf", &rate);
(gdb) next
1.5
26          for(year = 1; year <= 10; year++)
(gdb) next
...
```

EECS10: Computational Methods in ECE, Lecture 8 (c) 2019 R. Doemer 32

Debugging Example

- Example session: **`Interest2.c`** (part 3/6)

```
...
27     { interest = amount * (rate/100.0);
(gdb) next
28       balance = amount + interest;
(gdb) print interest
$1 = 15
(gdb) print amount
$2 = 1000
(gdb) print balance
$3 = -7.3987334479772013e+304
(gdb) next
29       printf("Interest for year%3d is $%8.2f.\n", year, interest);
(gdb) print balance
$4 = 1015
(gdb) next
Interest for year  1 is $   15.00.
30       printf("The new balance is      $%8.2f.\n", balance);
(gdb) next
The new balance is      $ 1015.00.
31       amount = balance;
(gdb) next
...
```

Debugging Example

- Example session: **`Interest2.c`** (part 4/6)

```
...
26 for(year = 1; year <= 10; year++)
(gdb) next
27     { interest = amount * (rate/100.0);
(gdb) print year
$5 = 2
(gdb) list
22 printf("Please enter the interest rate in %%:\n");
23 scanf("%lf", &rate);
24
25 /* computation and output section */
26 for(year = 1; year <= 10; year++)
27     { interest = amount * (rate/100.0);
28       balance = amount + interest;
29       printf("Interest for year%3d is $%8.2f.\n", year, interest);
30       printf("The new balance is      $%8.2f.\n", balance);
31       amount = balance;
(gdb) list 35
30       printf("The new balance is      $%8.2f.\n", balance);
31       amount = balance;
32      } /* rof */
...
```

Debugging Example

- Example session: **`Interest2.c`** (part 5/6)

```
...
33
34 /* exit */
35 return 0;
36 } /* end of main */
37
38 /* EOF */
(gdb) break 35
Breakpoint 2 at 0x1079c: file Interest2.c, line 35.
(gdb) cont
Continuing.
Interest for year  2 is $   15.22.
The new balance is      $ 1030.22.
Interest for year  3 is $   15.45.
The new balance is      $ 1045.68.
...
Interest for year 10 is $   17.15.
The new balance is      $ 1160.54.

Breakpoint 2, main () at Interest2.c:35
35 return 0;
...
```

Debugging Example

- Example session: **`Interest2.c`** (part 6/6)

```
...
(gdb) print balance
$6 = 1160.5408250251503
(gdb) cont
Continuing.

Program exited normally.
(gdb) quit
%
```