

Problem Solving using Conditionals

Rupesh Nasre.

Mentor TAs: Ahmed, Vivek, Vimala, Akash,
Kankan, Rahul, Swati, Akshay, Ashok, Keshav

IIT Madras
May 2022

Assignments are not enough!

- We need to make a decision: this way or the other.
- Finding the larger number
 - if x is less than y then y , otherwise x
- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes
- Will I pass this course?
 - If marks are equal to or above 40 then yes, otherwise GBY.

Conditionals

- Finding the larger number
 - if x is less than y then y, otherwise x

```
#include <stdio.h>

int main() {
    int x, y;
    scanf("%d%d", &x, &y);
    if (x < y)
        printf("%d is larger\n", y);
    else
        printf("%d is larger\n", x);
}
```

```
#include <stdio.h>
int main() {
    int x, y;
    scanf("%d%d", &x, &y);
    if (x < y) {
        printf("%d", y);
    } else {
        printf("%d", x);
    }
    printf(" is larger\n");
}
```

```
printf("%d is larger\n", (x < y ? y : x));
```

Problem: Match blood groups

Four blood groups: A, B, O, C (assume AB as C)

What happens if I mistakenly use `(bgone == bgtwo)`?

```
scanf("%c%c", &bgone, &bgtwo);  
if bgone is same as bgtwo then  
    printf("Blood groups match.\n");  
otherwise  
    printf("It is a mismatch.\n");
```

```
scanf("%c%c", &bgone, &bgtwo);  
if (bgone == bgtwo) {  
    printf("Blood groups match.\n");  
} else {  
    printf("It is a mismatch.\n");  
}
```

If O is a universal donor and C is a universal recipient, check if bgone can donate blood to bgtwo.

Allow blood group AB (instead of C). This can be done using char arrays.

Problem: Find your Bro!

- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes

```
scanf("%c%c%d%c%d",
      &first, &second, &year,
      &prog, &classroll);
if (first == 'C') {
    if (second == 'S') {
        printf("Hi Bro!\n");
    } else {
        printf("Excuse me?\n");
    }
}
```

```
scanf("%c%c%d%c%d",
      &first, &second, &year,
      &prog, &classroll);
if (first == 'C') {
    if (second == 'S') {
        printf("Hi Bro!\n");
    } else {
        printf("Excuse me?\n");
    }
}
```

This **else** corresponds to which **if**?
The ambiguity is clearer if we remove braces.

Compile-time error

Problem: Find your Bro!

- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes

```
scanf("%c%c%d%c%d",
      &first, &second, &year,
      &prog, &classroll);
if (first == 'C') {
    if (second == 'S') {
        printf("Hi Bro!\n");
    } else {
        printf("Excuse me?\n");
    }
}
```

Removing these braces would make it equivalent to the left code.

```
scanf("%c%c%d%c%d",
      &first, &second, &year,
      &prog, &classroll);
if (first == 'C') {
    if (second == 'S')
        printf("Hi Bro!\n");
} else
    printf("Excuse me?\n");
```

else corresponds to the **second** if.

else corresponds to the **first** if.

How do we modify the code so that the else corresponds to the first if?

And both the codes are incomplete.

Silent on EE, MS, ...

Silent on CE, CH, ...

Problem: Find your Bro!

- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes

```
scanf("%c%c%d%c%d",
      &first, &second, &year,
      &prog, &classroll);
if (first == 'C')
    if (second == 'S')
        printf("Hi Bro!\n");
    else
        printf("Excuse me?\n");
else
    printf("Excuse me?\n");
```

CS21B010
Hi Bro!

CE21B010
Excuse me?

MS21B010
Excuse me?

If first if condition is false,
the second if condition
is not evaluated.

We need both the elses.

- It would be nice if this repeated `printf("Excuse me?\n");` is avoided.
- Can we make this work for small-case roll numbers also (`cs21b010`)?

Problem: Find your Bro!

- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes

```
scanf("%c%c%d%c%d",
      &first, &second, &year,
      &prog, &classroll);
if (first == 'C') {
    if (second == 'S')
        printf("Hi Bro!\n");
    else
        printf("Excuse me?\n");
} else
    printf("Excuse me?\n");
```

If first condition is false,
the second condition
is not evaluated.

```
scanf("%c%c%d%c%d",
      &first, &second, &year,
      &prog, &classroll);
if (first == 'C' && second == 'S')
    printf("Hi Bro!\n");
else
    printf("Excuse me?\n");
```

- It would be nice if this repeated `printf("Excuse me?\n");` is avoided.
- Can we make this work for small-case roll numbers also (`cs21b010`)?

Problem: Find your Bro!

- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes

If first `||` condition is **true**, the second `||` condition is not evaluated.

If first `&&` condition is **false**, the second `&&` condition is not evaluated.

```
if ((first == 'C' || first == 'c') &&
    (second == 'S' || second == 's'))
    printf("Hi Bro!\n");
else
    printf("Excuse me?\n");
```

This is called **short-circuiting**.

Extend this for only your batch: CS21B

```
if ((first == 'C' && second == 'S') ||
    (first == 'c' && second == 's'))
    printf("Hi Bro!\n");
else
    printf("Excuse me?\n");
```

Are these two codes equivalent?

- It would be nice if this repeated `printf("Excuse me?\n");` is avoided.
- Can we make this work for small-case roll numbers also (`cs21b010`)?

Problem: Find your Bro!

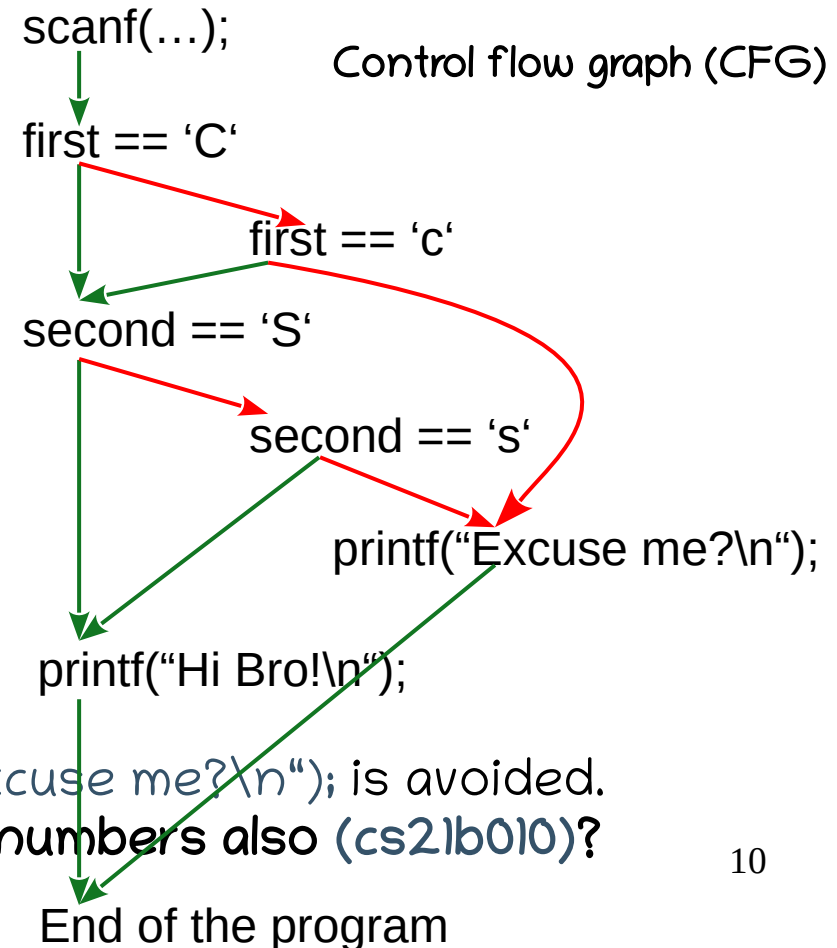
- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes

If first `||` condition is **true**, the second `||` condition is not evaluated.

If first `&&` condition is **false**, the second `&&` condition is not evaluated.

```
if ((first == 'C' || first == 'c') &&  
    (second == 'S' || second == 's'))  
    printf("Hi Bro!\n");  
else  
    printf("Excuse me?\n");
```

This is called **short-circuiting**.



- It would be nice if this repeated `printf("Excuse me?\n");` is avoided.
- Can we make this work for small-case roll numbers also (`cs21b010`)?

Problem: Find your Bro!

- Finding if a roll number is from our department
 - if the first two letters are 'C' and 'S', then yes

```
if ((first == 'C' && second == 'S') ||  
    (first == 'c' && second == 's'))  
    printf("Hi Bro!\n");  
else  
    printf("Excuse me?\n");
```

Draw control flow graph
(CFG) for this code.

- It would be nice if this repeated `printf("Excuse me?\n");` is avoided.
- Can we make this work for small-case roll numbers also (`cs21b010`)?

Problem: Lucky Cards!

- Four suites: Club C, Diamond D, Spade S, Heart H
- Each suite has 13 cards: A, 2, 3, ..., 9, T, J, Q, K
 - Lucky cards: SA, any H, DQ, DK, any 7

```
if ((suite == 'S' && card == 'A') ||
    (suite == 'H') ||
    (suite == 'D' && (card == 'Q' || card == 'K')) ||
    (card == '7'))
    printf("Lucky you!\n");
else
    printf("Better luck next time.\n");
```

if suite is invalid or card is invalid
printf("Invalid card\n");
else

```
if ((suite == 'S' && card == 'A') ||
    (suite == 'H') ||
    (suite == 'D' && (card == 'Q' || card == 'K')) ||
    (card == '7'))
    printf("Lucky you!\n");
else
    printf("Better luck next time.\n");
```

- For an input H0 or (*, the code should print "Invalid card".
- Support D10 (instead of DT). Can be done without char arrays.

Problem: Lucky Cards!

- Four suites: Club C, Diamond D, Spade S, Heart H
- Each suite has 13 cards: A, 2, 3, ..., 9, T, J, Q, K
 - Lucky cards: SA, any H, DQ, DK, any 7

```
if ((suite != 'C' && suite != 'D' &&
    suite != 'S' && suite != 'H' ) ||
    (card != 'A' && card != 'T' && card != 'J' &&
    card != 'Q' && card != 'K' &&
    ((int)card < '2' || (int)card > '9'))
    printf("Invalid card\n");
else
```

```
if suite is invalid or card is invalid
    printf("Invalid card\n");
```

```
else
```

```
if ((suite == 'S' && card == 'A') ||
    (suite == 'H') ||
    (suite == 'D' && (card == 'Q' || card == 'K')) ||
    (card == '7'))
    printf("Lucky you!\n");
else
    printf("Better luck next time.\n");
```

- For an input H0 or (*, the code should print "Invalid card".
- Support D10 (instead of DT). Can be done without char arrays.

Problem: Lucky Cards!

- Four suites: Club C, Diamond D, Spade S, Heart H
- Each suite has 13 cards: A, 2, 3, ..., 9, T, J, Q, K
 - Lucky cards: SA, any H, DQ, DK, any 7

```
if (!(suite == 'C' || suite == 'D' ||
    suite == 'S' || suite == 'H') &&
    (card == 'A' || card == 'T' || card == 'J' ||
    card == 'Q' || card == 'K') &&
    ((int)card >= '2' && (int)card <= '9'))
    printf("Invalid card\n");
else
```

De Morgan's Law
 $!(x \ \&\& \ y) == (!x \ || \ !y)$

```
if (suite is valid and card is valid)
    printf("Invalid card\n");
else
    if ((suite == 'S' && card == 'A') ||
        (suite == 'H') ||
        (suite == 'D' && (card == 'Q' || card == 'K')) ||
        (card == '7'))
        printf("Lucky you!\n");
    else
        printf("Better luck next time.\n");
```

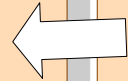
- For an input H0 or (*, the code should print "Invalid card".
- Support D10 (instead of DT). Can be done without char arrays.

Problem: ISD Code to Country

When conditions use equality over constant integers, C provides an alternative: switch-case.

```
scanf("%d", &isd);  
if (isd == 91) printf("India\n");  
else if (isd == 1) printf("US-Canada\n");  
else if (isd == 61) printf("Australia\n");  
else if (...) ...  
else  
    printf("Unknown ISD code\n");
```

is equivalent to



```
scanf("%d", &isd);  
if (isd == 91) printf("India\n");  
else if (isd == 1) printf("US-Canada\n");  
else if (isd == 61) printf("Australia\n");  
else if (...) ...  
else  
    printf("Unknown ISD code\n");
```

Correctness-wise, the order of conditions does not matter.
Performance-wise, the more frequent conditions should appear first.
(e.g., if most queries are about India, that should be the first condition)

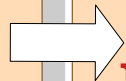
- Given **ISD code**, find the country.

91: India, 1: US, 93: Afghanistan, 61: Australia, 55: Brazil, 1:
Canada, 86: China, 20: Egypt, 49: Germany, 39: Italy, 218:
Libya, 850: N Korea, 92: Pakistan, 7: Russia, 82: S Korea, 94:
Sri Lanka, 380: Ukraine

Problem: ISD Code to Country

```
scanf("%d", &isd);
if (isd == 91) printf("India\n");
else if (isd == 1) printf("US-Canada\n");
else if (isd == 61) printf("Australia\n");
else if (...) ...
else
    printf("Unknown ISD code\n");
```

is equivalent to



```
scanf("%d", &isd);
switch (isd) {
    case 91: printf("India\n"); break;
    case 1: printf("US-Canada\n"); break;
    case 61: printf("Australia\n"); break;
    case ...: ...
    default: printf("Unknown ISD code\n");
}
```

Future Connect:
break is also used to
come out of loops.

When condition

91
India

over constant integers, C

91
India

switch-case.

- Given **ISD code**, find the country

91: India, 1: US, 93: Afghanistan, 61: Australia, 1:
Canada, 86: China, 20: Egypt, 49: Germany, 8:
Libya, 850: N Korea, 92: Pakistan, 7: Russia, 1:
Sri Lanka, 380: Ukraine

Problem: ISD Code to Country

Why this complication?
Is there a use of such
a behavior?

```
1
US-Canada
Australia

91
India
```

```
scanf("%d", &isd);
switch (isd) {
    case 91: printf("India\n"); break;
    case 1: printf("US-Canada\n"); break;
    case 61: printf("Australia\n"); break;
    case ...: ...
    default: printf("Unknown ISD code\n");
}
```

When conditions use equality over constant integers, C provides an alternative: switch-case.

- Given **ISD code**, find the country.

91: India, 1: US, 93: Afghanistan, 61: Australia, 55: Brazil, 1: Canada, 86: China, 20: Egypt, 49: Germany, 39: Italy, 218: Libya, 850: N Korea, 92: Pakistan, 7: Russia, 82: S Korea, 94: Sri Lanka, 380: Ukraine

Problem: Seating Arrangement

Why this complication?
Is there a use of such
a behavior?

21
CRC
19
CRC
18
Raman
17
Ramanujan

Avoids repetition of
common processing.

```
scanf("%d", &batch);  
switch (batch) {  
  case 19:  
  case 21: printf("CRC\n"); break;  
  
  case 18: printf("Raman\n"); break;  
  
  case 20:  
  default: printf("Ramanujan\n"); break;  
}
```

Problem: Year-wise Courses

```
scanf("%d", &batch);  
printf("You should have completed ");  
switch(batch) {  
    case 18: printf("Prof. Ethics, Internship ");  
    case 19: printf("Compilers, PoP, OS, ");  
    case 20: printf("LMC, FCSD, COA, DAA, OOAIA, ");  
    case 21: printf("CS1111, DM\n");  
}
```

19

You should have completed Compilers, PoP, OS, LMC, FCSD, COA, DAA, OOAIA, CS1111, DM

- Given a year of admission, print the core courses that should have been completed.

Two uses of fallthrough

```
scanf("%d", &batch);  
switch (batch) {  
    case 19:  
    case 21: printf("CRC\n"); break;  
  
    case 18: printf("Raman\n"); break;  
  
    case 20:  
    default: printf("Ramanujan\n"); break;  
}
```

When there is many-to-one mapping

Multiple cases map to one processing.

Examples

- Case-insensitive processing
- Mobile number to name mapping
- Issued book to roll number mapping
- Problem to algorithm mapping (DAA)

```
scanf("%d", &batch);  
printf("You should have completed ");  
switch (batch) {  
    case 18: printf("Prof. Ethics, Internship");  
    case 19: printf("Compilers, PoP, OS, ");  
    case 20: printf("LMC, FCSD, COA, OOAIA, ");  
    case 21: printf("CS1111, DM\n");  
}
```

When there is subsumption of processing

A case subsumes another's processing.

Examples

- Eligibility of quarters based on seniority
- Finding all taller than a given student
- Constraints in normal forms (Databases)
- Properties of various languages (LMC)
- [Duff's device](#)

Problem: Month Days

```
scanf("%d", &month);  
switch (month) {  
    case 1: case 3: case 5: case 7: case 8:  
    case 10: case 12: printf("31\n"); break;  
    case 4: case 6: case 9: case 11: printf("30\n"); break;  
    case 2: printf("28 or 29\n"); break;  
    default: printf("Invalid month %d\n", month);  
}
```

```
if (month == 1 || month == 3 || month == 5 || month == 7 ||  
    month == 8 || month == 10 || month == 12)  
    printf("31\n");  
else if (month == 4 || month == 6 || month == 9 || month == 11)  
    printf("30\n");  
else if (month == 2)  
    printf("28 or 29\n");  
else printf("Invalid month %d\n", month);
```

Given a month 1..12, print the number of days in that month.

Old Problem: Lucky Cards!

- Four suites: Club C, Diamond D, Spade S, Heart H
- Each suite has 13 cards: A, 2, 3, ..., 9, T, J, Q, K
 - Lucky cards: SA, any H, DQ, DK, any 7

Nested switch

```
if (!(suite == 'C' || suite == 'D' ||
    suite == 'S' || suite == 'H') &&
    (card == 'A' || card == 'T' || card == 'J' ||
    card == 'Q' || card == 'K') &&
    ((int)card >= '2' && (int)card <= '9'))
    printf("Invalid card\n");
else
    // check luck.
```

```
switch (suite) {
    case 'C': case 'D': case 'S': case 'H':
        switch (card) {
            case 'A': case 'T': case 'J': case 'Q':
            case 'K': case '2': case '3': case '4':
            case '5': case '6': case '7': case '8':
            case '9': // check luck.
            default: printf("Invalid card\n");
        }
    default: printf("Invalid card\n");
}
```

Restrictions on cases

```
scanf("%f", &weight);  
switch (weight) {  
  case 19.2: ...  
  case 21.5: ...  
  default: ...  
}
```



```
int final = 18, prefinal = 19,  
    sophomore = 20, freshies = 21;  
scanf("%d", &batch);  
switch (batch) {  
  case final: ...  
  case prefinal: ...  
  case sophomore: ...  
  case freshies: ...  
  default: ...  
}
```



Wh

```
#define final 18  
#define prefinal 19  
#define sophomore 20  
#define freshies 21  
scanf("%d", &batch);  
switch (batch) {  
  case final: ...  
  case prefinal: ...  
  case sophomore: ...  
  case freshies: ...  
  default: ...  
}
```



integers, C provides an alternative: switch-case.

```
enum {final = 18, prefinal = 19,  
    sophomore = 20, freshies = 21};  
scanf("%d", &batch);  
switch (batch) {  
  case final: ...  
  case prefinal: ...  
  case sophomore: ...  
  case freshies: ...  
  default: ...  
}
```



#define

To see preprocessed output: `gcc -E file.c`

```
#include <stdio.h>
int main() {
#define FSTR "%d"
    int yearofbirth;
    scanf(FSTR, &yearofbirth);
    int age = (2022 - yearofbirth);
    printf(FSTR, age);
}
```

```
#include <stdio.h>
#define output printf
#define input scanf

int main() {
#define FSTR "%d"
    int yearofbirth;
    input(FSTR, &yearofbirth);
    int age = (2022 - yearofbirth);
    output(FSTR, age);
}
```

```
#include <stdio.h>
#define allmain \
int main() { \
    int yearofbirth; \
    scanf("%d", &yearofbirth); \
    int age = (2022 - yearofbirth); \
    printf("%d", age); \
}
```

allmain

```
#include <stdio.h>
#define _main
#define __
#define BEGIN (
#define END )
#define _a int
#define __a "%d"
_a _BEGIN END {
    _a a;
    scanf BEGIN __a, &a END;
    _a a_ = BEGIN 2022 - a END;
    printf BEGIN __a, a_ END;
}
```

The interested
may take a look
at the LOCCC.

Uses of #define

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

```
#define PLUS(x, y)    x + y
```

```
int main() {  
    printf("%d + %d = %d\n", 5, 6, PLUS(5, 6));  
    printf("%d + %d = %d\n", 3, 7, PLUS(3, 7));  
}
```

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

```
#define DEBUG 0
```

```
#define dprintf if (DEBUG) printf
```

```
int main() {  
    dprintf("Start of main\n");  
    int n = 5;  
    n = n - 1;  
    printf("n = %d\n", n);  
    dprintf("End of main\n");  
}
```

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

```
#define SQ(x)    x*x
```

```
int main() {  
    if (SQ(3) == 9)  
        printf("(2+3)*(2+3) = %d\n", SQ(2+3));  
}
```

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

```
#define log(x) printf(x)
```

```
#define ERROR(x)    {  
    if (x == 0) ;  
    else if (x == 1) log("Warning\n");  
    else if (x == 2) log("Error\n");  
    else log("Serious error\n");  
}
```

```
int main() {  
    // some code  
    if (x < y) ERROR(1);  
}
```

Problem: Mini Calculator

```
scanf("%d%c%d", &x, &op, &y);  
switch (op) {  
    case '+': printf("%d\n", x + y); break;  
    case '-': printf("%d\n", x - y); break;  
    case '*': printf("%d\n", x * y); break;  
    case '/': if (y != 0) printf("%d\n", x / y);  
              else printf("Division by zero error\n");  
              break;  
    default: printf("Invalid operator %c\n", op);  
}
```

Future Connect:

To support full-fledged expressions (such as $3 - 4 * 5 / (6 + 7)$), we will need a stack.

Given an expression num#num, print its value. # is +, -, *, /.

Summary

- if, if else, nested if
- Logical expressions with `&&` and `||`
- switch-case
- `#define`
- Problem Solving with conditionals