# Fundamentals of Programming

## Lecture 8

## C Operations

# Remember: average score

average.c

```c
float score,sum;
int n;

sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);
  if (score < 0)
    break;
  sum = sum + score;
  n++;
}
printf("average=%f\n", sum/n);
```

# Remember: average score

average.c

```c
float score,sum;
int n;

sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);
  if (score < 0)
    break;
  sum = sum + score;
  n++;
}
printf("average=%f\n", sum/n);
```

```
CS@kntu:lecture8$ gcc average.c && ./a.out
12
13
-1
average=12.500000
```

# Remember: average score

average.c

```c
float score,sum;
int n;

sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);
  if (score < 0)
    break;
  sum = sum + score;
  n++;
}
printf("average=%f\n", sum/n);
```

```
CS@kntu:lecture8$ gcc average.c && ./a.out
12
13
-1
average=12.500000
```

```
CS@kntu:lecture8$ gcc average.c && ./a.out
-1
average=-nan
```

# Divide by zero!

average.c

```
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

printf("average=%f\n", sum/n);
```

average2.c

```
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

if (n == 0)
  printf("No score entered!\n");
else
  printf("average=%f\n", sum/n);
```

# Divide by zero!

intdivzero.c

```c
#include <stdio.h>

int main() {
  int a,b;

  a = 10;
  b = 0;

  printf("%d\n", a/b);

  return 0;
}
```

# Divide by zero!

intdivzero.c

```c
#include <stdio.h>

int main() {
  int a,b;

  a = 10;
  b = 0;

  printf("%d\n", a/b);

  return 0;
}
```

```
CS@kntu:lecture8$ gcc intdivzero.c && ./a.out
Floating point exception (core dumped)
```

# Divide by zero!

intdivzero.c

```c
#include <stdio.h>

int main() {
  int a,b;

  a = 10;
  b = 0;

  printf("%d\n", a/b);

  return 0;
}
```

floatdivzero.c

```c
#include <stdio.h>

int main() {
  float a,b;

  a = 10;
  b = 0;

  printf("%f\n", a/b);

  return 0;
}
```

# Divide by zero!

intdivzero.c

```c
#include <stdio.h>

int main() {
  int a,b;

  a = 10;
  b = 0;

  printf("%d\n", a/b);

  return 0;
}
```

floatdivzero.c

```c
#include <stdio.h>

int main() {
  float a,b;

  a = 10;
  b = 0;

  printf("%f\n", a/b);

  return 0;
}
```

```
CS@kntu:lecture8$ gcc floatdivzero.c && ./a.out
inf
```

# printf floating point format

average2.c

```c
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

if (n == 0)
  printf("No score entered!\n");
else
  printf("average=%f\n", sum/n);
```

```
CS@kntu:lecture8$ gcc average.c && ./a.out
12
13
-1
average=12.500000
```

# printf floating point format

average3.c

```
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

if (n == 0)
  printf("No score entered!\n");
else
  printf("average=%.2f\n", sum/n);
```

# printf floating point format

average3.c

```
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

if (n == 0)
  printf("No score entered!\n");
else
  printf("average=%.2f\n", sum/n);
```

```
CS@kntu:lecture8$ gcc average3.c && ./a.out
12
13
-1
average=12.50
```

# printf floating point format

average4.c

```
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

if (n == 0)
  printf("No score entered!\n");
else
  printf("average=%8.2f\n", sum/n);
```

# printf floating point format

average4.c

```c
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

if (n == 0)
  printf("No score entered!\n");
else
  printf("average=%8.2f\n", sum/n);
```

```
CS@kntu:lecture8$ gcc average4.c && ./a.out
12
13
-1
average=   12.50
```

# printf floating point format

average4.c

```c
sum = 0;
n = 0;
while (1) {
  scanf("%f", &score);

  if (score < 0)
    break;

  sum = sum + score;
  n++;
}

if (n == 0)
  printf("No score entered!\n");
else
  printf("average=%8.2f\n", sum/n);
```

https://www.cprogramming.com/tutorial/printf-format-strings.html

https://en.wikipedia.org/wiki/Printf_format_string

```
CS@kntu:lecture8$ gcc average4.c && ./a.out
12
13
-1
average=   12.50
```

# Integer division

intdiv.c

```
  int   a, b, c;
  float f, g, h;

  a = 10;
  b = 4;

  c = a/b;
  f = a/b;

  printf("%d\n",
c);
  printf("%f\n",
f);
```

# Integer division

intdiv.c

```c
  int   a, b, c;
  float f, g, h;

  a = 10;
  b = 4;

  c = a/b;
  f = a/b;

  printf("%d\n",
c);
  printf("%f\n",
f);
```

```
CS@kntu:lecture8$ gcc intdiv.c && ./a.out
2
2.000000
```

# Integer division

intdiv.c

```
  int   a, b, c;
  float f, g, h;

  a = 10;
  b = 4;

  c = a/b;
  f = a/b;

  printf("%d\n",
c);
  printf("%f\n",
f);
```

intdiv2.c

```
  int   a, b, c;
  float f, g, h;

  a = 10;
  b = 4;

  g = a;
  h = b;

  f = g/h;

  printf("%f\n",
f);
```

# Integer division

intdiv.c

```
  int   a, b, c;
  float f, g, h;

  a = 10;
  b = 4;

  c = a/b;
  f = a/b;

  printf("%d\n",
c);
  printf("%f\n",
f);
```

intdiv2.c

```
  int   a, b, c;
  float f, g, h;

  a = 10;
  b = 4;

  g = a;
  h = b;

  f = g/h;

  printf("%f\n",
f);
```

```
CS@kntu:lecture8$ gcc intdiv2.c && ./a.out
2.500000
```

# Integer division

intdiv.c

```c
int   a, b, c;
float f, g, h;

a = 10;
b = 4;

c = a/b;
f = a/b;

printf("%d\n",
c);
printf("%f\n",
f);
```

intdiv2.c

```c
int   a, b, c;
float f, g, h;

a = 10;
b = 4;

g = a;
h = b;

f = g/h;

printf("%f\n",
f);
```

intdiv3.c

```c
int   a, b, c;
float f, g, h;

a = 10;
b = 4;

g = a;
h = b;

f = g/h;
f = a/h;
f = g/b;
f = (float) a / (float)
b;
f = a / (float) b;
f = (float) a / b;
```

# operators - precedence

```
a + b * c
```

# operators - precedence

```
a + b * c
```

```
a + (b * c)
```

# operators - precedence

```
a / b - d * a
```

# operators - precedence

```
a / b - d * a
```

```
(a / b) - (d * a)
```

# operators - precedence

```
a * -b - -c * d
```

# operators - precedence

```
a * -b - -c * d
```

```
a * (-b) - (-c) * d
```

# operators - precedence

```
a * -b - -c * d

(a * (-b)) - ((-c) * d)
```

# operators - precedence

```
a / b / c
```

# operators - precedence

```
(a / b) / c
```

# operators - precedence

```
a - b + c

a + b - c

a * b / c

a / b * c
```

# operators - precedence

```
a - b + c  ⟹  (a - b) + c

a + b - c  ⟹  (a + b) - c

a * b / c  ⟹  (a * b) / c

a / b * c  ⟹  (a / b) * c
```

# Assignment operators

| op | usage | equivalent |
|---|---|---|
| += | a += b | a = a + b |
| -= | a -= b | a = a - b |
| *= | a *= b | a = a * b |
| /= | a /= b | a = a / b |
| %= | a %= b | a = a % b |

# increment and decrement

| op | usage | equivalent |
|---|---|---|
| ++ | a++ | a = a + 1 (*) |
| ++ | ++a | a = a + 1 |
| -- | a-- | a = a - 1 (*) |
| -- | --a | a = a - 1 |

# Assignment as an operator

```c
int a,b;

a = 1;
b = 2;

printf("%d\n", a + b);
printf("%d\n", a = b);
```

# Assignment as an operator

```
int a,b;

a = 1;
b = 2;

printf("%d\n", a + b);
printf("%d\n", a += b);
```

# operators - assignment

```
a = b = c
```

# operators - assignment

```
a = (b = c)
```

# operators associativity

```
a = b = c    ⟹    a = (b = c)

a - b - c    ⟹    (a - b) - c
```

# operators associativity

right to left

```
a = b = c    ⟹    a = (b = c)
```

```
a - b - c    ⟹    (a - b) - c
```

left to right

# operators - assignment

```
a = b = c = d = e;
```

# operators - assignment

```
- - - a
```

# operators - assignment

```
- (- (-a))
```

# operators - assignment

right to left

```
- (- (-a))
```

# increment and decrement

```c
int i;


i = 1;
printf("%d\n", i);
printf("%d\n", ++i);
printf("%d\n", i);
```

```c
int i;


i = 1;
printf("%d\n", i);
printf("%d\n", i++);
printf("%d\n", i);
```

| Category | Operator | Associativity |
|---|---|---|
| Postfix | () [] -> . ++ - - | Left to right |
| Unary | + - ! ~ ++ - - (type)* & sizeof | Right to left |
| Multiplicative | * / % | Left to right |
| Additive | + - | Left to right |
| Shift | << >> | Left to right |
| Relational | < <= > >= | Left to right |
| Equality | == != | Left to right |
| Bitwise AND | & | Left to right |
| Bitwise XOR | ^ | Left to right |
| Bitwise OR | \| | Left to right |
| Logical AND | && | Left to right |
| Logical OR | \|\| | Left to right |
| Conditional | ?: | Right to left |
| Assignment | = += -= *= /= %=>>= <<= &= ^= \|= | Right to left |
| Comma | , | Left to right |

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

# operators - comparison

```
a + b >= c * d
```

# operators - comparison

```
a > b + c && k == d
```

# operators - comparison

```
20 > 16 > 10
```

# operators - comparison

```
(20 > 16) > 10
```

# operators - comparison

```
1 > 10
```

| Category | Operator | Associativity |
|---|---|---|
| Postfix | () [] -> . ++ - - | Left to right |
| Unary | + - ! ~ ++ - - (type)* & sizeof | Right to left |
| Multiplicative | * / % | Left to right |
| Additive | + - | Left to right |
| Shift | << >> | Left to right |
| Relational | < <= > >= | Left to right |
| Equality | == != | Left to right |
| Bitwise AND | & | Left to right |
| Bitwise XOR | ^ | Left to right |
| Bitwise OR | \| | Left to right |
| Logical AND | && | Left to right |
| Logical OR | \|\| | Left to right |
| Conditional | ?: | Right to left |
| Assignment | = += -= *= /= %=>>= <<= &= ^= \|= | Right to left |
| Comma | , | Left to right |

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

# operators - comparison

```
a + b >= c * d
```

# operators - comparison

```
a > b + c && k == d
```

# operators - comparison

```
20 > 16 > 10
```

# operators - comparison

```
(20 > 16) > 10
```

# operators - comparison

```
1 > 10
```

# Count the ratings

```c
int n, rating, bad, average, good;

bad = average = good = 0;
n = 0;
while (1) {
  printf("Enter rating: ");
  scanf("%d", &rating);

  // write your code here!

  n++;
}

printf("Bad: %.1f%% \n",     100*bad/(float)n);
printf("Average: %.1f%% \n", 100*average/(float)n);
printf("Good: %.1f%% \n",    100*good/(float)n);
```

1: Bad ★☆☆

2: Average ★★☆

3: Good ★★★

# Count the ratings

```
bad = average = good = 0;
n = 0;
while (1) {
  printf("Enter rating: ");
  scanf("%d", &rating);

  if (rating == 1)
      bad++;
  else if (rating == 2)
      average++;
  else if (rating == 3)
      good++;
  else
     break
  n++;
}
```

countscores1.c

```
printf("Bad: %.1f%% \n",     100*bad/(float)n);
printf("Average: %.1f%% \n", 100*average/(float)n);
printf("Good: %.1f%% \n",    100*good/(float)n);
```

# Count the ratings

```c
bad = average = good = 0;
n = 0;
while (1) {
  printf("Enter rating: ");
  scanf("%d", &rating);

  if (rating == 1)
      bad++;
  else if (rating == 2)
      average++;
  else if (rating == 3)
      good++;
  else
     break
  n++;
}
```

countscores1.c

```c
bad = average = good = 0;
n = 0; finish = 0;
while (1) {
    printf("Enter rating: ");
    scanf("%d", &rating);

    switch (rating) {
    case 1:
      bad++;
      break;
    case 2:
      average++;
      break;
    case 3:
      good++;
      break;
    default:
      finish = 1;
      break;
    }

    if (finish == 1)
      break;
    n++;
  }
```

countscores2.c

# Count the ratings

```c
bad = average = good = 0;
n = 0;
while (1) {
  printf("Enter rating: ");
  scanf("%d", &rating);

  if (rating == 1)
      bad++;
  else if (rating == 2)
      average++;
  else if (rating == 3)
      good++;
  else
     break
  n++;
}
```

countscores1.c

```c
bad = average = good = 0;
n = 0; finish = 0;
while (1) {
    printf("Enter rating: ");
    scanf("%d", &rating);

    switch (rating) {
    case 1:
      bad++;
      break;
    case 2:
      average++;
      break;
    case 3:
      good++;
      break;
    default:
      finish = 1;
      break;
    }

    if (finish)
      break;
    n++;
  }
```

countscores2.c

```c
int n, rating, finish;
int bad_average, good;
bad_average = good = 0;
n = 0;
finish = 0;

while (1) {
    printf("Enter rating: ");
    scanf("%d", &rating);

    switch (rating) {
    case 1:
    case 2:
      bad_average++;
      break;
    case 3:
      good++;
      break;
    default:
      finish = 1;
      break;
    }
    if (finish)
      break;
    n++;
}
```

countscores3.c

countscores3.c

```c
printf("Bad & Average: %.1f%% \n",100*bad_average/(float)n);
printf("Good: %.1f%% \n", 100*good/(float)n);
```

# logical operators

```
10 < a && a < 16
```

# logical operators

```
if (|a| > 10)
```

# logical operators

```
a > 10 || a < -10
```

# logical operators

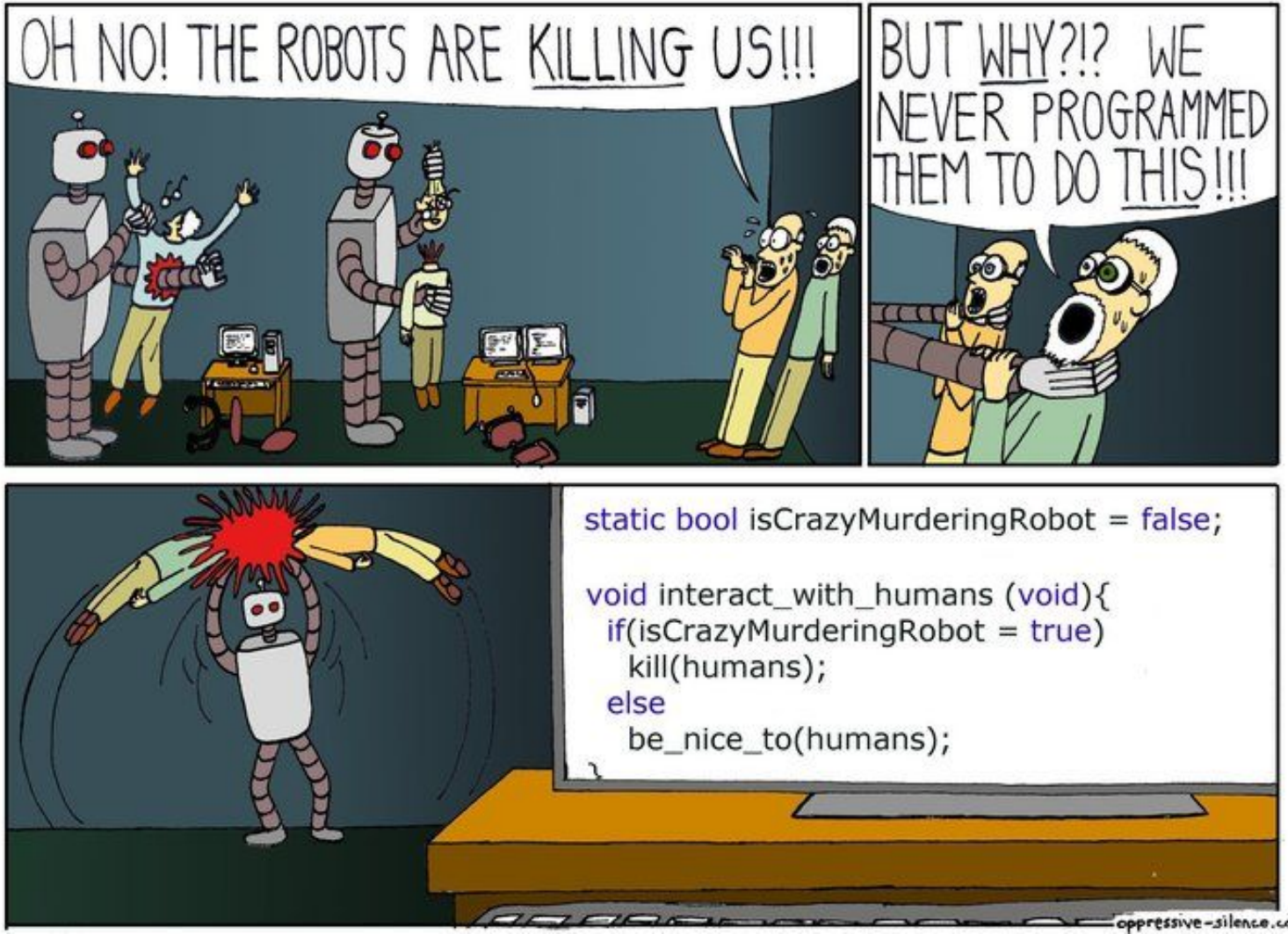
```
!(a <= 10 && a >= -10)
```

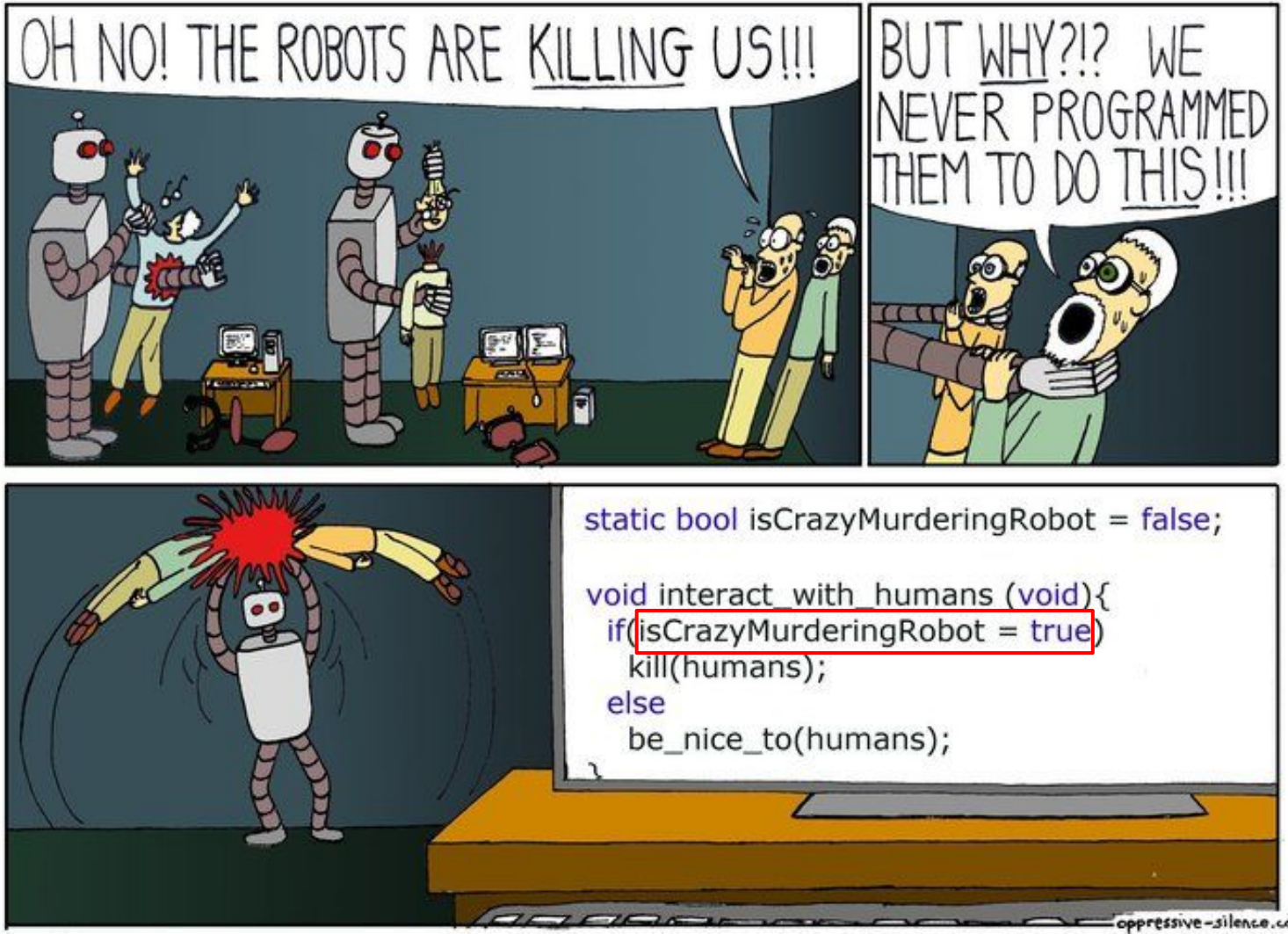
# logical operators

```
!(a == b)   a != b
```

# confusing '==' with '='

```
while (1) {

    safe = check_safe();

    if (safe = 1) {

      demolish_the_building();

      break;

    }

    else
      wait_5_seconds();

  }
```


OH NO! THE ROBOTS ARE KILLING US!!!
BUT WHY?!? WE NEVER PROGRAMMED THEM TO DO THIS!!!
static bool isCrazyMurderingRobot = false;
void interact_with_humans (void){
if(isCrazyMurderingRobot = true)
kill(humans);
else
be_nice_to(humans);
}
oppressive-silence.com


OH NO! THE ROBOTS ARE KILLING US!!!
BUT WHY?!? WE NEVER PROGRAMMED THEM TO DO THIS!!!
static bool isCrazyMurderingRobot = false;
void interact_with_humans (void){
if(isCrazyMurderingRobot = true)
kill(humans);
else
be_nice_to(humans);
}
oppressive-silence.com

passfail1.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || grade < 10;
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

passfail1.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || grade < 10;
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

```
fail = fail || grade < 10
```

passfail1.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || grade < 10;
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

```
fail = fail || (grade < 10)
```

passfail1.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || grade < 10;
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

```
fail = (fail || (grade < 10))
```

passfail1.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || grade < 10;
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

passfail2.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || (grade < 10);
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

passfail1.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || grade < 10;
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

passfail2.c

```c
int fail = 0;
double grade;

while (1) {
    scanf("%lf", &grade);

    if (grade == -1)
      break;

    fail = fail || (grade < 10);
}

if (fail)
    puts("Fail!");
else
    puts("Pass!");
```

more readable

```c
int g,a;

a = 0;
while (1) {
  printf("Enter grade: ");
  scanf("%d", &g);

  if (g== -1)
    break;

  a = a || g < 10;

}


if (a)
  puts("fail");
else
  puts("pass");
```

```c
int g,a;

a = 0;
while (1) {
  printf("Enter grade: ");
  scanf("%d", &g);

  if (g== -1)
    break;

  a = a || g < 10;

}


if (a)
  puts("fail");
else
  puts("pass");
```

```c
int g,a;

a = 0;
while (1) {
  printf("Enter grade: ");
  scanf("%d", &g);

  if (g== -1)
    break;

  a = a || (g < 10);

}


if (a)
  puts("fail");
else
  puts("pass");
```