Computer Programming 1 Lab

2020-10-08

Outline

- decision-making in C
- repetition statements
- Exercise3

decision-making in C

decision-making in C

Boolean data type

- Two states (true or false)
- Logical operators - AND(`&&`), OR(`||`), NOT(`!`)

decision-making in C

examples:

```
if((1+1 == 2) && (1+1 == 3)){  //  returns false
    //  This part will NOT be executed.
}

if((1+1 == 2) || (1+1 == 3)){  //  returns true
    //  This part will be executed.
}

if(!(1+1 == 3)){ //  returns true
    //  This part will be executed.
}
```

decision-making in C

what "else"

- Can only be used with `if()`.
- Executed when the previous `if()` does not execute.

decision-making in C

examples:

```
if(Letter == 'A'){
    //  Do something
}
else if(Letter == 'B'){
    //  Do something
}
else if(Letter == 'C'){
    //  Do something
}
else{
    //  Do something
}
```

decision-making in C

"switch" on

- switch between cases
- `break;` each cases
- Use `default` as the last `else`

decision-making in C

examples:

```
switch(Letter){
    case 'A':
        //  Do something
        break;
    case 'B':
        //  Do something
        break;
    case 'C':
        //  Do something
        break;
    default:
        //  Do something
        break;
}
```

decision-making in C

"switch" Tips:

- Don't forget to break.

example:

```
switch(Letter){
   case 'A':
       printf("The letter is A.\n");
   case 'B':
       printf("The letter is B.\n");
   case 'C':
       printf("The letter is C.\n");
   default:
       printf("None of them above.\n");
}
```

results:

```
darkknive@1091cp1:~$ ./a.out B
The letter is B.
The letter is C.
None of them above.
darkknive@1091cp1:~$
```

repetition statements

repetition statements

Introducing "for"

- Usage: `for(init; condition; increment){}`
- init part will be executed before for loop start.
- condition part will be executed before each looped. Only when return value is true will the next loop be triggered.
- increment will be executed after each loop.

repetition statements

In conclution, this is how for loop works...

init ->
if(condition == true) -> execute { } -> increment ->
if(condition == true) -> execute { } -> increment ->
if(condition == true) -> execute { } -> increment ->
...
if(condition == false) -> leave for()

repetition statements

example:

```
for(int a = 0; a < 5; a++){
    printf("%d\n", a);
}
```

results:

```
darkknive@1091cp1:~$ ./a.out
0
1
2
3
4
darkknive@1091cp1:~$
```

repetition statements

"For" Pro Tips:

1. Declear an int and start with 0, set condition as `index < N;` and increment as `index++`. This for loop will run N times with index = 0, 1, 2, 3.....N-2, N-1.
2. If you get a segmentation fault during runtime, it may because your for loop messed up. For example, `for(int index = N-1; index >= 0; index++)`.
3. You may declare multiple variables in init part by using `int a = 0, b = 0, ...;`. Please note that they should be the same data type.

repetition statements

do "while"

- Usage: `while(condition){statement(s)}`.
- While (condition == true), do statement(s), then do the whole loop again.

In conclution, this is how it works...

if(condition == true) -> execute { } ->
if(condition == true) -> execute { } ->
...
if(condition == false) -> leave while()

repetition statements

example:

```
int total = 100;
while(total != 0){
    printf("%d ", total);
    total /= 2;
}
printf("\n");
```

results

```
darkknive@1091cp1:~$ ./a.out
100 50 25 12 6 3 1
darkknive@1091cp1:~$
```

repetition statements

do "while"

- Another form of while loop is `do{statement(s)}while(condition);`
- Do statements first, then check condition.
- Stops while (condition == false).

In conclution, this is how it works...

execute { } -> if(condition == true) ->
execute { } -> if(condition == true) ->
...
execute { } -> if(condition == false) -> leave while()

repetition statements

example:

```c
int total = 100;
do{
    printf("%d ", total);
    total /= 2;
}
while(total != 0);
printf("\n");
```

results

```
darkknive@1091cp1:~$ ./a.out
100 50 25 12 6 3 1
darkknive@1091cp1:~$
```

repetition statements

"While" Pro Tips:

1. If your runtime is stucked, it is very possible that you have an infinite while loop. For example, `while(a > 1){printf("%d ", a);}`. The value of `a` won't be changed in the loop, so if you enters this while loop, it's gonna run FOREVER.

Notes:

- Format your code!
- control your input smartly with scanf().
- Every argument has its reason of existence.
- Think as a program.

Exercise3

Any Question?

Course? Assignment? Exercise? TA?