Introduction to Programming and Computational Physics

Lecture 4

Iterations

Iterative structures in C: for

It happens frequently that the same operation must be repeated several times. If we want to add four times the number 7 to the variable sum we could write:

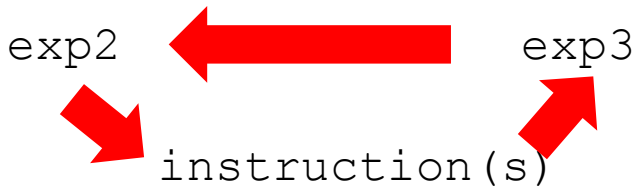
```
sum = sum + 7;

sum = sum + 7;

sum = sum + 7;

sum = sum + 7;
```

The C language offers a vary compact way to perform the same operation:

```
int i; //it is our "counter"

for (i=1; i<=4; i=i+1)

   sum = sum + 7;
```

How the for cycle works

```
for (exp1; exp2; exp3)

  instruction;
```




```
for (exp1; exp2; exp3)

{

  instructions;

}
```

1. At the begin of the for cycle exp1 is **executed**
2. exp2 is **evaluated**. If it is true, the instruction is executed, otherwise the for cycle is terminated.
3. exp3 is **executed**.
4. exp2 is **evaluated again**. If it is true, the instruction is executed again, otherwise the for cycle is terminated.
5. exp3 is **executed again**.
6. …

The sum of the first n numbers

```
#include <stdio.h>

int main()
{
    int sum = 0;
    int n = 200; //n is set to 200
    int i;
    for (i=1; i<=n; i++)   // i++ is like i = i+1
        sum+=i; // the same as sum = sum+i
    printf ("\nThe sum of the first %d numbers is %d \n",n,sum);
    return 0;
}
```

The average of n numbers given by the user

```
#include <stdio.h>

int main()
{
  int n,i;
  float value = 0;
  float sum = 0;
  float ave = 0;
  printf("\n");
  printf("How many numbers? ");
  scanf("%d", &n);
  for (i=1; i<=n; i++)
    {
      printf("Enter a number ");
      scanf("%f", &value);
      sum+=value;
    }
  ave = sum/n;
  printf("\n");
  printf("The average is %.3f\n\n",ave);
  return 0;
}
```

When the instructions in the `for` loop are more than one, brackets are needed

Some other example

In `for(exp1;exp2;exp3)` we may use any expression allowed by the C language

We can write:

```
for(i=5;i>=1;i=i-1)
```

so that i takes the values 5,4,3,2,1,0

```
for(i=7;i>=-8;i=i-3)
```

so that i takes the values 7,4,1,-2,-5,-8,-11

What happens if we write?

```
for(i=5;i>=5;i=i+2)
```

A nested for cycle

Any easy way to print the multiplication table

```c
#include <stdio.h>

int main()
{
  printf("\n");
  int i,j;
  for (i=1; i<=10; i++)
    {
      for (j=1; j<=10; j++)
        {
          printf("%4d ", i*j);
        }
      printf("\n");
    }
  printf("\n");
  return 0;
}
```

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

10 lines only !

Iterative structures in C: while

Another way to perform an iterative structure in C is given by the `while` cycle

```
while (exp)
{
  instruction(s);
}
```

It works in the following way:

1) Exp is evaluated
2) If exp is true, the instruction(s) is(are) executed. Otherwise the cycle is terminated.
3) Exp is evaluated again
4) …

The `while` cycle is very useful when the number of iterations in not known in advance

The sum of n integer numbers given by the user

```
#include <stdio.h>

int main()
{
    int sum = 0;
    int num = 1; // so that num is "true"
    printf ("\nEnter zero to terminate \n");
    while (num) // the same as while (num!=0)
    {
        printf("Enter a number: ");
        scanf("%d", &num);
        sum+=num; // the same as sum = sum + num
    }
    printf("\nThe sum is %d \n\n",sum);
    return 0;
}
```

for VS while

When the number of iteration is well defined, they can be used in an equivalent way:

```
for (exp1; exp2; exp3)
{
  instructions;
}
```

⇔

```
exp1;
while (exp2)
{
  instructions;
  exp3;
}
```

for VS while

...but these two codes are fully equivalent...

```
int num = 1;

while (num)

{

  printf("Enter a
number: ");

  scanf("%d",&num);

  sum+=num;

}
```

```
int num;

for(num=1;num;sum+=num)

{

  printf("Enter a
number:");

  scanf("%d",&num);

}
```

Iterative structures in C: do…while

A third way to perform an iterative structure in C is given by the `do…while` cycle

```
do
{
   instruction(s)
}
while (exp);
```

It works in the following way:

1) Instruction(s) is(are) executed
2) Exp is evaluated
3) If exp is true, the instruction(s) is(are) executed again

It is very similar to the `while` cycle but the instructions are executed at least once

Interruptions: break

We have already seen that the command `break` brings outside a `case` structure. The same command can be used also to interrupt a cycle (`for`, `while`, `do…while`)

```
int i = 1;
while(i)
{
   scanf("%f",&var);
   if(var<0) break;
   var2 = sqrt(var);
}
```

This loop can be stopped only entering a negative number

Interruptions: continue

The instruction `continue` inside a cycle causes the <u>current</u> iteration to be termineted

```
int i;
for (i=1;i<10;i++)
{
   scanf("%f",&var)
   if(var<0) continue;
   var2 = sqrt(var);
}
```

The sum of n positive numbers given by the user

n is not fixed… the user decides when to stop it by entering 0

negative numbers are not taken into account

```
#include <stdio.h>

int main()
{
    int sum = 0;
    int num = 1;
    printf ("\nEnter zero to terminate \n");
    while (num)
    {
        printf("Enter a number: ");
        scanf("%d", &num);
        if (num<0) continue;
        sum+=num;
    }
    printf("\nThe sum is %d \n\n",sum);
    return 0;
}
```