Lecture 12
Structures

Simple data types

- Up to now we used simple, built-in data types, such as int, double, etc.
- Those could be grouped using static or dynamic arrays.
- Example: Write a program simulating motion of a couple of "robots"
  - Each robot has a name, position, instant velocity and acceleration
  - Each robot moves, according to its velocity
  - Our program will get rather messy for larger number of robots
- If our program becomes larger it will become difficult to handle
- Need a method to organize it better
- We know how to separate functionality, by dividing what program does in to different functions
- Structures are used to group data in to larger constructs, allowing a better overview of it

Structure

User defined data type

Structure is a user defined data type available in C that allows the programmer to combine data items of different kinds.

Syntax:

```
struct structure_name {
   member_type member_mane;
   member_type member_mane;
   ...
   member_type member_mane;
} one or more structure variables;
```

E.g.:

```
struct Robot {
   char  name[50]; //String identifying the robot, its name
   double x,y,vx,vy,ax,ay; //position, velocity, acceleration
   ...
} r1, r2;
```

Structure

User defined data type

Access to *member* elements through '.', and $->$

Usage:

```
struct Robot {
    char  name[50];
    double x,y,vx,vy,ax,ay;
    ...
} r1, r2;

int main(){
  //r1 and r2 are allready defined and global
  r1.x = 8.0;

  struct Robot r3;

  struct Robot *p = &r3;
  p->y=5.0;
}
```

Write an example, define objects of type Robot, and acces data through '.' and '$->$'

Structure

Use with functions

Structures can be passed to functions, by value (copy) and with a pointer.

C language does not allow structures to 'posses' functions (in C++ this is allowed), but we still can add a pointer to a function ...

Write an example of a list data colloection using structures. What is a list?