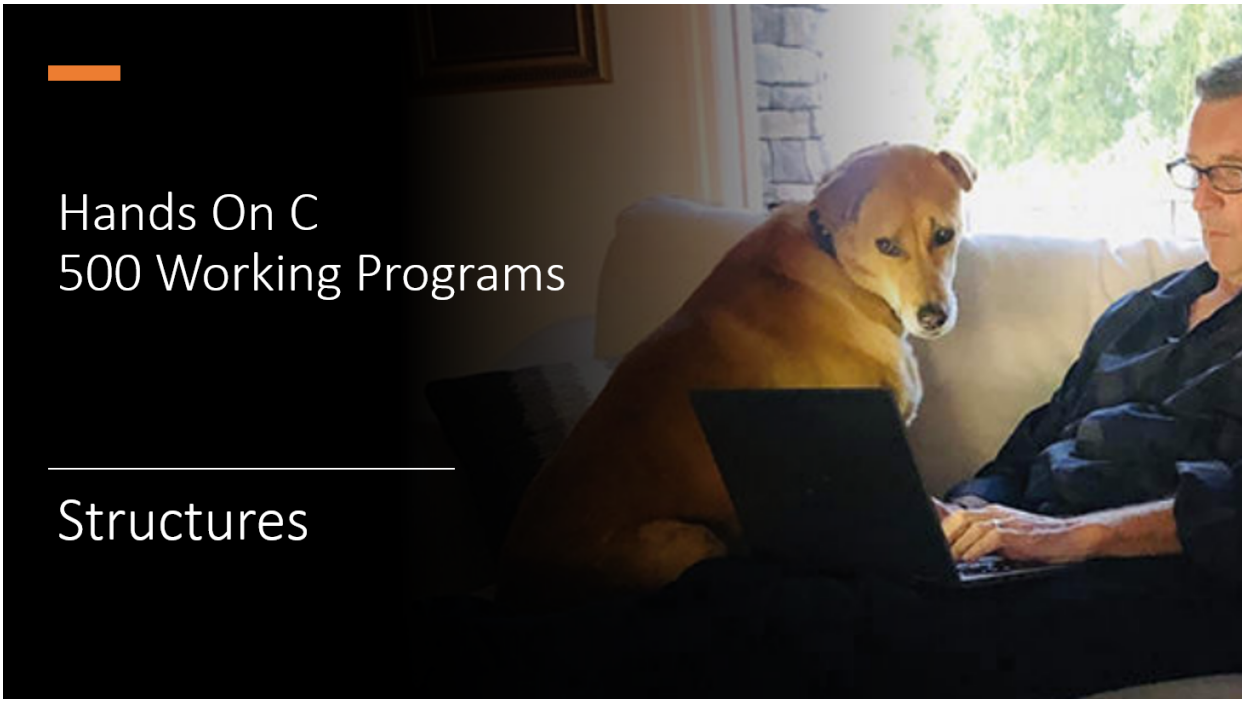

Understanding Structures

In [1]:
```
#include <stdio.h>

void showEmployee(int EmployeeID, char EmployeeName[], char JobTitle[], float Sal
{
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s\n",
          EmployeeID, EmployeeName, JobTitle, Salary, Phone);
}

int main(void)
{
    int EmployeeID = 1;
    char EmployeeName[256] = "Kris Jamsa";
    char JobTitle[256] = "Programmer";
    float Salary = 100000.0;
    char Phone[32] = "928-555-1212";

    showEmployee(EmployeeID, EmployeeName, JobTitle, Salary, Phone);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212
```

```
struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};
```

In [2]:
```
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};

int main(void)
{
  struct Employee worker;
  printf("sizeof worker %ld bytes\n", sizeof(worker));
}
```

```
sizeof worker 552 bytes
```

In [3]:
```c
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};

int main(void)
{
  struct Employee worker;

  worker.EmployeeID = 1;
  strcpy(worker.EmployeeName, "Kris Jamsa");
  strcpy(worker.JobTitle, "Programmer");
  worker.Salary = 100000;
  strcpy(worker.Phone, "928-555-1212");
}
```

Initializing a Structure at Declaration

In [4]:
```c
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTItle[256];
  float Salary;
  char Phone[32];
};

int main(void)
{
  struct Employee programmer = { 1, "Kris Jamsa", "Programmer", 100000, "928-555-

  printf("Name: %s\n", programmer.EmployeeName);
}
```

```
Name: Kris Jamsa
```

Passing a Structure to a Function

In [5]:
```
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};


void ShowEmployee(struct Employee worker)
{
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s\n",
          worker.EmployeeID, worker.EmployeeName, worker.JobTitle, worker.Salary,
}


int main(void)
{
  struct Employee coder = { 1, "Kris Jamsa", "Programmer", 100000, "928-555-1212"
  struct Employee manager = { 1, "Debbie Jamsa", "Manager", 125000, "928-555-1213

  ShowEmployee(coder);
  ShowEmployee(manager);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212
ID 1 Name: Debbie Jamsa Title Manager Salary 125000.00 Phone 928-555-1213
```

Revisiting Integer Division with div

In [6]:
```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  div_t result;

  result = div(20, 7);

  printf("20 divided by 7 is %d remainder %d\n", result.quot, result.rem);
}
```

```
20 divided by 7 is 2 remainder 6
```

Creating an Array of Structures

In [7]:
```c
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};

void ShowEmployee(struct Employee worker)
{
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s\n",
          worker.EmployeeID, worker.EmployeeName, worker.JobTitle, worker.Salary,
}

int main(void)
{
  struct Employee staff[2] = {
                                  { 1, "Kris Jamsa", "Programmer", 100000, "928-555
                                  { 2, "Debbie Jamsa", "Manager", 125000, "928-555-
                               };

  ShowEmployee(staff[0]);
  ShowEmployee(staff[1]);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212
ID 2 Name: Debbie Jamsa Title Manager Salary 125000.00 Phone 928-555-1213
```

Passing an Array of Structures to a Function

In [8]:
```
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};

void ShowStaff(struct Employee Employees[], int employeeCount)
{
  for (int i = 0; i < employeeCount; i++)
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s\n",
          Employees[i].EmployeeID, Employees[i].EmployeeName, Employees[i].JobTit
          Employees[i].Salary, Employees[i].Phone);
}

int main(void)
{
  struct Employee staff[2] = {
                                { 1, "Kris Jamsa", "Programmer", 100000, "928-555
                                { 1, "Debbie Jamsa", "Manager", 125000, "928-555-
                              };

  ShowStaff(staff, 2);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212
ID 1 Name: Debbie Jamsa Title Manager Salary 125000.00 Phone 928-555-1213
```

Changing a Structure's Value in a Function

In [9]:
```
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};


void PayRaise(struct Employee *worker, float amount)
{
    (*worker).Salary += amount;
}


void ShowEmployee(struct Employee worker)
{
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s\n",
          worker.EmployeeID, worker.EmployeeName, worker.JobTitle, worker.Salary,
}


int main(void)
{
  struct Employee coder = { 1, "Kris Jamsa", "Programmer", 100000, "928-555-1212"

  ShowEmployee(coder);
  PayRaise(&coder, 25000);
  ShowEmployee(coder);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212
ID 1 Name: Kris Jamsa Title Programmer Salary 125000.00 Phone 928-555-1212
```

Using -> to Reference a Structure Member Passed by Reference

In [10]:
```
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
};


void PayRaise(struct Employee *worker, float amount)
{
    worker->Salary += amount;
}


void ShowEmployee(struct Employee worker)
{
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s\n",
          worker.EmployeeID, worker.EmployeeName, worker.JobTitle, worker.Salary,
}


int main(void)
{
  struct Employee coder = { 1, "Kris Jamsa", "Programmer", 100000, "928-555-1212"

  ShowEmployee(coder);
  PayRaise(&coder, 25000);
  ShowEmployee(coder);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212
ID 1 Name: Kris Jamsa Title Programmer Salary 125000.00 Phone 928-555-1212
```

Revisiting System Time

```
struct tm
{
  int tm_sec;
  int tm_min;
  int tm_hour;
  int tm_mday;
  int tm_mon;   // 0 thru 11
  int tm_year;  // year - 1900
  int tm_wday;  // Sun 0 - Sat 6
  int tm_yday;  // 1 - 365 day of year
  int tm_isdst; // 1 if daylight savings 0 otherwise
}
```

In [11]:
```c
#include <stdio.h>
#include <time.h>

int main(void)
{
  struct tm *currentDate;
  time_t seconds;

  time(&seconds);

  currentDate = localtime(&seconds);
  printf("Date: %d-%d-%d\n", currentDate->tm_mon+1, currentDate->tm_mday, currentI
  printf("Time: %d:%02d\n", currentDate->tm_hour, currentDate->tm_min);
}
```

```
Date: 2-1-2021
Time: 13:06
```

Using a Nested Structure

In [12]:
```
#include <stdio.h>
#include <string.h>

struct Employee {
  int EmployeeID;
  char EmployeeName[256];
  char JobTitle[256];
  float Salary;
  char Phone[32];
  struct tm {
    int Month;
    int Day;
    int Year;
  } HireDate;
};

void ShowEmployee(struct Employee worker)
{
    printf("ID %d Name: %s Title %s Salary %6.2f Phone %s Hire Date: %d-%2d-%d\n"
          worker.EmployeeID, worker.EmployeeName, worker.JobTitle, worker.Salary,
          worker.HireDate.Month, worker.HireDate.Day, worker.HireDate.Year);
}


int main(void)
{
  struct Employee coder = { 1, "Kris Jamsa", "Programmer", 100000, "928-555-1212"
  struct Employee manager = { 2, "Debbie Jamsa", "Manager", 125000, "928-555-1213

  ShowEmployee(coder);
  ShowEmployee(manager);
}
```

```
ID 1 Name: Kris Jamsa Title Programmer Salary 100000.00 Phone 928-555-1212 Hire
Date: 9-30-2021
ID 2 Name: Debbie Jamsa Title Manager Salary 125000.00 Phone 928-555-1213 Hire
Date: 12- 8-2020
```

Understanding Bit Fields

In [13]:
```c
#include <stdio.h>

struct Date {
  unsigned int month: 5;
  unsigned int day: 5;
  unsigned int year: 12;
} ;

int main(void)
{
   struct Date today;

   today.month = 1;  // January
   today.day = 25;
   today.year = 2021;

   printf("Date: %d-%d-%d\n", today.month, today.day, today.year);
   printf("size in bytes %ld\n", sizeof(today));
}
```

```
Date: 1-25-2021
size in bytes 4
```

Understanding Unions

In [14]:
```c
#include <stdio.h>

struct WordRegisters {
  unsigned int AX, BX, CX, DX, SI, DI, FLAGS;
};

struct ByteRegisters {
  unsigned char al, ah, bl, bh, cl, ch, dl, dh;
};

union REGISTERS {
  struct WordRegisters w;
  struct ByteRegisters b;
};

int main(void)
{
  union REGISTERS regs;

  regs.b.al = 5;

  printf("AL contains %u\n", regs.b.al);
  printf("AX contains %u\n", regs.w.AX);
}
```

```
AL contains 5
AX contains 5
```

What You will Learn Next

To store data from one user session to the next, C programs often use files.

```
#include <stdio.h>

int main(void)
{
   FILE *fp = fopen("Alphabet.txt", "r");
   char letter;

   while ((letter = fgetc(fp)) != EOF)
     putchar(letter);

   fclose(fp);
}
```