

person-record.h

Page 1

```
1 #ifndef person_record_included
2 #define person_record_included
3
4 /*! \file
5 *
6 * This file contains three versions of a PersonRecord data structure to
7 * illustrate the way C lays out memory for structs, pointers, and arrays.
8 *
9 */
10
11 /**
12 * PersonRecord Version 1 has name, id, address, and age fields. The name and
13 * address are char*, the id and age int.
14 *
15 * Consider the following code segment:
16 *
17 *     PersonRecordV1 prv1;
18 *     prv1.name = "Jane Doe";
19 *     prv1.id = 123456789;
20 *     prv1.address = "1 Main St."
21 *     prv1.age = 25;
22 *
23 * Here is a picture of the memory layout of variable prv1:
24 *
25 *     
26 *
27 * Consider now the following code segment, which dynamically allocates a value
28 * of type PersonRecordV1:
29 *
30 *     PersonRecordV1 prvlp = new(PersonRecordV1);
31 *     prvlp->name = "Jane Doe";
32 *     prvlp->id = 123456789;
33 *     prvlp->address = "1 Main St."
34 *     prvlp->age = 25;
35 *
36 * Here is a picture of the memory layout of variable prvlp:
37 *
38 *     
39 *
40 */
41
42 typedef struct {
43     char* name;           /**< name is a variable-length string */
44     int id;               /**< id is an int */
45     char* address;        /**< address is a variable-length string */
46     int age;              /**< age is an int */
47 } PersonRecordV1;
48
49 /**
50 * PersonRecord Version 2 has the same fields as a Person Record Version 1,
51 * except the name and address fields are fixed character arrays instead of
52 * char*. These fields are 30 and 50 chars, respectively.
53 *
54 * Consider the following code segment:
55 *     PersonRecordV2 prv2;
56
57 *         strcpy(prv2.name, "Jane Doe");
58 *         prv2.id = 123456789;
59 *         strcpy(prv2.address, "1 Main St.");
60 *         prv2.age = 25;
61 *
62 * Here is a picture of the memory layout of variable prv1:
63 *     
64 *
65 */
66
67 typedef struct {
68     char name[20];          /**< name is a string of max 20 chars */
69     int id;                /**< id is an int */
70     char address[25];       /**< address is a string of max 25 chars */
71     int age;                /**< age is an int */
72 } PersonRecordV2;
73
74 /**
75 * Name is an extended version of the name field in a person record. It's used
76 * in PersonRecord Version 3. Instead of the the simple char* name field in a
77 * PersonRecordV1, the Name struct has three char* fields for first, middle,
78 * and last names.
79 */
80 typedef struct {
81     char* first;            /**< first name is a string */
82     char middle_initial;   /**< middle initial is just one char */
83     char* last;             /**< last name is a string */
84 } Name;
85
86 /**
87 * Address is an extended version of the address field in a person record. It's used
88 * in PersonRecord Version 3. Instead of the the simple char* address field in
89 * a PersonRecordV1, the Address struct has six separate fields for number,
90 * street, etc.
91 */
92 typedef struct {
93     int number;             /**< street number is an int */
94     char* street;           /**< street name is a string */
95     char* city;             /**< city name is a string */
96     char state[2];          /**< state is limited to a 2-char string */
97     char* country;          /**< country is a string */
98     int zip;
99 } Address;
100
101 /**
102 * PersonRecord Version 3 has the same fields as Versions 1 and 2, except here
103 * in V3, the name and address fields are structs, no just strings. The reader
104 * is invited to draw a picture.
105 */
106 typedef struct {
107     Name name;              /**< name is defined by the Name type */
108     int id;                /**< id is an int */
109     Address address;        /**< address is defined by the Address type */
110     int age;
111 } PersonRecordV3;
112
```

```
113 /**
114  * Print out a PersonRecordV1.
115 */
116 void printPersonRecordV1(PersonRecordV1 prv1);
117
118 /**
119  * Print out a PersonRecordV1*.
120 */
121 void printPersonRecordV1p(PersonRecordV1* prv1p);
122
123 /**
124  * Print out a PersonRecordV2.
125 */
126 void printPersonRecordV2(PersonRecordV2 prv2);
127
128 /**
129  * Print out a PersonRecordV3.
130 */
131 void printPersonRecordV3(PersonRecordV3 prv3);
132
133 #endif
```