In the C language structures are used to group together different types of variables under the same name. For example you could create a structure “telephone”: which is made up of a string (that is used to hold the name of the person) and an integer (that is used to hold the telephone number).

Take a look at the example:

```c
struct telephone
{
    char *name;
    int number;
};
```

**Note:** the ; behind the last curly bracket.

With the declaration of the structure you have created a new type, called telephone. Before you can use the type telephone you have to create a variable of the type telephone. Take a look at the following example:
#include<stdio.h>

struct telephone
{
    char *name;
    int number;
};

int main()
{
    struct telephone index;

    return 0;
}

Note: index is now a variable of the type telephone.
To access the members of the structure telephone, you must use a dot between the structure name and the variable name (variables: name or number.) Take a look at the next example:

```c
#include<stdio.h>

struct telephone
{
    char *name;
    int number;
};

int main()
{
    struct telephone index;

    index.name = "Jane Doe";
    index.number = 12345;
    printf("Name: %s\n", index.name);
    printf("Telephone number: %d\n", index.number);
    return 0;
}
```
Type definitions and structures

Type definitions make it possible to create your own variable types. In the following example we will create a type definition called “intptr” (a pointer to an integer):

```c
#include<stdio.h>

typedef int *int_ptr;

int main()
{
    int_ptr myvar;
    return 0;
}
```
It is also possible to use type definitions with structures. The name of the type definition of a structure is usually in uppercase letters. Take a look at the example:

```c
#include<stdio.h>

typedef struct telephone
{
    char *name;
    int number;
}TELEPHONE;

int main()
{
    TELEPHONE index;

    index.name = "Jane Doe";
    index.number = 12345;
    printf("Name: %s\n", index.name);
    printf("Telephone number: %d\n", index.number);

    return 0;
}

Note: The word struct is not needed before TELEPHONE index;
```
**Pointer to structures**

If you want a pointer to a structure you have to use the -> (infix operator) instead of a dot. Take a look at the following example:

```c
#include<stdio.h>
typedef struct telephone
{
    char *name;
    int number;
}TELEPHONE;

int main()
{
    TELEPHONE index;
    TELEPHONE *ptr_myindex;

    ptr_myindex = &index;

    ptr_myindex->name = "Jane Doe";
    ptr_myindex->number = 12345;
    printf("Name: %s\n", ptr_myindex->name);
    printf("Telephone number: %d\n", ptr_myindex->number);
    return 0;
}
```

**Note:** The -> (infix operator) is also used in the printf statement.
// STRUCTURE WITH BOTH POINTERS AND ARRAY ON THE SAME VARIABLE
// STRUCT CARS EXAMPLE WHERE THE BRAND, MAKE AND PRODUCTION YEAR
// ARE ENTERED
#include<stdio.h>
#define SIZE 2

struct car { 
    int year;
    char *brand[12];
    char *make[12];
};

int main() {  
    struct car Car[SIZE]; /* new variable Car of type struct car*/
    int i;

    for(i=0; i<SIZE; i++)
    {
        printf("Enter Brand: 
"); scanf("%s", Car[i].brand);
        printf("Enter Make: 
"); scanf("%s", Car[i].make);
        printf("Enter age: 
"); scanf("%d", &Car[i].year);
    }
    for(i=0 ; i<SIZE ; i++)
    {
        printf("Car Brand: %s, Make: %s of year %d
", Car[i].brand, Car[i].make, Car[i].year);
    }
    return 0;
}