Lesson 4

Character strings, and data structures

In this lesson, you will learn:

- What character strings are
- How to declare character arrays
- How to read character strings into memory using the `scanf()`, and `gets()`
- About the `%s` format specifier
- About the string functions `strlen()`, `strcmp()`, `strcpy()`
- What data structures are
- How to declare data structures

Important Notes:

- **You should type in all the programs in this handout, and run them more than once with different data**
- **You should read, and understand everything in this handout, the material in it forms the basis of the quizzes**
- **If you don't understand something; ask me to explain.**

Characters

When we declared a character variable in a program, we have used the line of code:

```
char the_character ' ';   // declare a character variable
```

This line of code sets up an area of memory, referred to by the variable name `the_character`, and able to hold a single ASCII character, such as: (`'a', '*', '2', '+'`).

Character strings (character arrays)

Character strings are the same kind of thing that we usually call words.

Character strings are just a bunch of characters grouped together. To use character strings in a C program, we have to declare variables of a new data type.

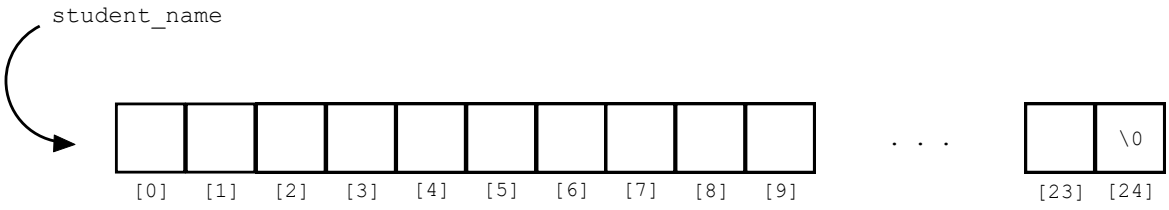
This new data type is called a character array, and to declare a variable of this type requires a line of code like:

```
char student_name[25] = "";   // declare a character array variable
```

NOTE:

- The `char` tells you that the variable's *base type* is character
- The square brackets `[]` tells you that the variable is an array type
- The `25` tells you how many characters the variable can hold

A pictorial representation of this variable is below:



NOTE:

- The first element of the character array is numbered 0
- The last element of the character array is numbered 24
- The last element of the character array contains the *null character* `'\0'`

Program (student1.c)

```
/*
  Program "student1.c"
  Written by: Joe Dorward
  Date: 03/30/00

  This program demonstrates a simple use of character strings
*/

#include <stdio.h>   /* list header files */
#include <string.h>

void main(void)
{
char student_name[25] = "";


  printf("\n Enter your name: ");
  scanf("%s",student_name);

  printf("\n Hello %s \n",student_name);
}
```

Program (student2.c)

```
/*
  Program "student2.c"
  Written by: Joe Dorward
  Date: 03/30/00

  This program demonstrates a simple use of character strings
*/

#include <stdio.h>   /* list header files */
#include <string.h>

void main(void)
{
char student_name[25] = "";


  printf("\n Enter your name: ");
  gets(student_name);

  printf("\n Hello %s \n",student_name);
}
```

Program (`string2.c`)

This program reads in a string of characters, then it prints out the string one character at a time.

```
/*
  Program "string2.c"
  Written by: Joe Dorward
  Date: 03/09/00

  This program uses a while loop to write out the characters
  of a string to the screen.
*/

#include <stdio.h>
#include <string.h>

void main(void)
{
int pause = 0;   // used for pausing while() loop

unsigned int string_element = 0;

char the_string[50] = "";

  printf("Input string:\n");

  printf("\t Please type a lowercase string: ");
  gets(the_string);

  printf("\nOutput string:\n");
  printf("\t ");

  while (string_element < strlen(the_string))
  {
    printf("%c",the_string[string_element]);   // print current character
    string_element++;

    for (pause = 1; pause < 20000000; pause++);   // count to twenty million
  }
  printf("\n\n");
}
```

Program (student3.c)

```
/*
  Program "student3.c"
  Written by: Joe Dorward
  Date: 03/30/00

  This program demonstrates a simple use of character strings
*/

#include <stdio.h>   /* list header files */
#include <string.h>

void main(void)
{
char student_name[25] = "",
     student_address[50] = "",
     student_date_of_birth[10] = "",
     student_program[25] = "";


  printf(" Enter the student\'s name: ");
  gets(student_name);

  printf(" Enter the student\'s address: ");
  gets(student_address);

  printf(" Enter the student\'s date of birth: ");
  gets(student_date_of_birth);

  printf(" Enter the student\'s program: ");
  gets(student_program);


  printf("\n Student: %s \n",student_name);
  printf(" Address: %s \n",student_address);
  printf(" Date of birth: %s \n",student_date_of_birth);
  printf(" Program: %s \n",student_program);
}
```

The standard function `strlen()`

The standard function `strlen()` is the function that returns the number of characters in a character string. The function returns a value of data type *unsigned integer*, which means a positive whole number.

```
/*
  Program "strlen.c"
  written by: Joe Dorward
  Date: 04/18/00

  This program demonstrates a call of the standard function strlen();
*/

#include <stdio.h>
#include <string.h>

void main(void)
{
unsigned int string_length = 0;   // strlen() returns an unsigned integer

char the_string[25] = "";

  printf(" Please enter a string: ");
  scanf("%s",the_string);

  string_length = strlen(the_string);   /* function call */

  printf("\n The string: %s,",the_string);
  printf(" contains: %u characters. \n",string_length);
}
```

The standard function `strcmp()`

This standard function is used to compare the contents of string variables in a similar way that the operators: ==, >, < are used to compare the contents of number variables.

In the case of strings, less-than, and greater-than means alphabetically, as in “apples” is less-than “oranges”.

The function returns a value of type integer

```
if ( strcmp(first_string,second_string) < 0 )     // first is first

if ( strcmp(first_string,second_string) == 0 )   // both strings are the same

if ( strcmp(first_string,second_string) > 0 )     // second is first
```

Program (strcmp.c)

```
/*
  Program "strcmp.c"
  written by: Joe Dorward
  Date: 04/18/00

  This program demonstrates a call of the standard function strcmp();
*/

#include <stdio.h>
#include <string.h>

void main(void)
{
char first_string[25] = "",
     second_string[25] = "";

  printf(" Please enter the first string: ");
  scanf("%s",first_string);

  printf(" Please enter the second string: ");
  scanf("%s",second_string);

  // the two strings are the same
  if ( strcmp(first_string,second_string) == 0 )
  {
    printf(" ** First if. \n");
    printf("\n The strings: %s,",first_string);
    printf(" and: %s are the same. \n",second_string);
  }


  // first is first
  if ( strcmp(first_string,second_string) < 0 )
  {
    printf(" ** Second if. \n");
    printf("\n The string: %s, comes",first_string);
    printf(" before the string: %s. \n",second_string);
  }


  // second is first
  if ( strcmp(first_string,second_string) > 0 )
  {
    printf(" ** Third if. \n");
    printf("\n The string: %s, comes",second_string);
    printf(" before the string: %s. \n",first_string);
  }
}
```

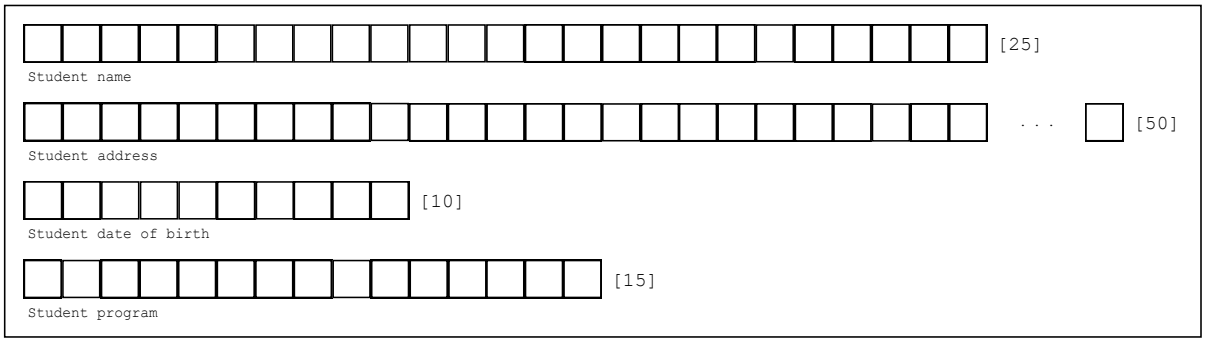
Data structures

In the program `student3.c` you wrote a program which asked the user for a student's personal details, then printed them out.

In a way, that program is the beginning of a computerized database.

A proper database program in C will use Data Structures, and the reserved word `struct` for this kind of task.

The Data Structure is just a way of packaging data together, and is very like a blank file card. The blank file card for the program `student4.c` is shown below.



In structures, each element of the structure is called a data field.

Next, is how that blank file card is defined in a C `struct` as a new data type:

```
struct a_student_record
{
  char student_name_field[25];
  char student_address_field[50];
  char student_date_of_birth_field[10];
  char student_program_field[15];
};
```

The name of this new data type is: `a_customer_record`

To use one of these structures, you have to declared it with the line of code:

```
struct a_student_record first_student_record;
```

NOTE:

- `struct` is a C reserved word
- `a_student_record` is the new data type
- `first_student_record` is a variable of type `a_student_record`

Program (student4.c)

```
/*
  Program "student4.c"
  Written by: Joe Dorward
  Date: 03/30/00

  This program demonstrates a simple use of a data structure
  and also how to use the standard function strcpy()
*/

#include <stdio.h>   // list header files
#include <string.h>

void main(void)
{
// declare character arrays
char temp_name[25] = "";            // temporary name variable
char temp_address[50] = "";         // temporary address variable
char temp_date_of_birth[10] = "";   // temporary date of birth variable
char temp_program[15] = "";         // temporary  variable

// define a data structure
struct a_student_record
{
  char student_name_field[25];
  char student_address_field[50];
  char student_date_of_birth_field[10];
  char student_program_field[15];
};

// declare a data structure
struct a_student_record first_student_record;

  // ================================================================
  /* get the new data to put into the data structure */
  printf(" Enter the student\'s name: ");
  gets(temp_name);
  strcpy(first_student_record.student_name_field,temp_name);

// add the code for entering and copying other data values here

  // ================================================================
  /* Now print data from the structure */
  printf("\n +-----------------------------------+");

  printf("\n The student\'s name is: ");
  printf("%s \n\n",first_student_record.student_name_field);

// add the code for printing other data fields here
}
```