PROGRAMMING LANGUAGES

Laboratory 10

POINTERS

I. FUNDAMENTALS

As already known, a program together with its data is stored in the computer's memory. The RAM memory is a random access memory (RAM) which, at its basic level, is comprised of bits. These bits could be either 0 or 1, while the computer is turned on. Eight bits form a byte, two bytes form a word and four bytes form a long word.

A pointer is a variable which contains the address of another variable, or, in other words, it represents a variable storing the address of a data instead of the data itself.

Pointers are used in order to refer to address-known data.

1. Declaring a pointer

A pointer is declared as any variable, just that its name is preceded by the character *.

Generally, a pointer is declared this way:

type *name_p;

which means that name_p is a pointer which points to a memory zone containing data of the specified type.

Comparing the above declaration with the common one:

type name;

we can consider that type* from this pointer declaration represents the type from an usual declaration.

Therefore, we can say that type* represents a new type, the pointer type. This type is often referred to as the *pointer-to-type* type.

Note:

* -> the indirect operator. The expression following it is a pointer and the result is a lvalue.

& -> the pointer operator. The operand is a lvalue and the result is a pointer.

For the declarations:

```
int x;
int *p;
float y;
```

the assignment:

```
p = &x;
```

is correct, while:

```
p = &y;
```

is incorrect, because *p* can only store addresses associated to int data.

For the declaration:

```
float *q;
```

the following assignment could be used:

```
q = &y;
```

There are cases when one could desire to use a pointer with more than only one data type. For this purpose, when declaring the pointer, the type must not be specified. This is done by using the void word:

```
void *name_p;
```

Example:

```
int x;
float y;
char c;
void *p;
 ...
 p = &x;
  ...
 p = &y;
  ...
 p = &c;
  ...
```

Using the key word void, the p pointer could be assigned adresses for data of various types.

When using void pointers, it is necessary to make explicite cast conversions (an operand's value is converted to the user-specified type using the unary operator (type)). This way, the data type associated to this pointer is specified:

```
(type) operand
```

Therefore, if p is declared as in the previously presented example, then an assignment of the following form:

```
*p = 100;
```

is incorrect, because the data type is undefined. In this case, the value of p must be converted to the type *int**, using the cast expression:

```
(int*)p
```

This way, the assignment from above becomes:

```
*(int*)p = 100;
```

2. Necessary functions for programs working with pointers

2.1. The "malloc" function

This function is declared in the header file malloc.h .The function:

```
malloc(val)
```

adds a number of *val* consecutive bytes from the available memory, returning their starting address:

2.2. The expression "sizeof"

```
sizeof(type)
```

returns the number off necessary bytes for a variable of the type „type”.
For instance, sizeof(int) returns the value 2.

2.3. The "calloc" function

This function is declared in the header file malloc.h .This function has two parameters

```
calloc(nr, dimens)
```

The first parameter specifies for how many objects memory allocation is necessary. The second parameter specifies the bytes dimension of each such object.

Example:

```
int *string;
...
string = (int*)calloc(5, sizeof(int));
```

"string" points to a memory zone that is sufficient for 5 integers (5 x 2 = 10 bytes).
The expression (int*) tells that the adress is going to be a pointer of the int type, representing a cast conversion.

II. ASSIGNMENT WORKFLOW

1. Write a program that uses two integer variables v and p. The last one has the pointer type. The vlue 200 is assigned to v and the address of v is assigned to p. Display the values , the memory location and the pointed value of the two variables.

2. Write a program that uses a single variable of the pointer to integer type called p, for which a memory location is allocated using the malloc function. For a certain specified size, display the value of p and the value pointed by p.

3. Let a and b be two integer variables initialized with two arbitrarily chosen values. Using pointer variables, make a program that swaps the values of the two integer values.

4. According to the principles of procedural programming, write the function:

```
void swap(int *x, int *y)
```

5. Write a program that uses the calloc function in order to allot memory for a pointer variable. Its dimension must be the one of three int variables. First initialize, then display the addresses and the values stored at these memory locations.

6. Write a program where the main function calls the reading function:

```
void read(float *p)
```

of a real number from the keyboard. The value is to be next displayed.

III. SOLUTIONS

1. L10_1.C

```
# include <stdio.h>

void main()
{
  int v, *p;
  p = &v;
  v = 200;
  printf("\nThe memory location of v:  %p\n", &v);
  printf("The value of  v:                %d\n", v);
  printf("The value of  p:                %p\n", p);
  printf("The value pointed by p:        %d\n", *p);
  getch();
}
```

2. L10_2.C

```
# include <stdio.h>
# include <alloc.h>

void main()
{
  int *p;
  p = (int*)malloc(sizeof(int));
  *p = 200;
  printf("\nThe value of p:              %p", p);
  printf("\nThe value pointed by:        %d\n", *p);
  getch();
}
```

3. L10_3.C

```
# include <stdio.h>

void main()

{
   int a=100, b=123, s, *x, * y;
   printf("\nThe start values for a and b are:\na=%d\nb=%d\n",a,b);
   x = &a;
   y = &b;
   s = *x;
   *x = *y;
   *y = s;
   printf("\nThe computed values for a and b are:\na=%d\nb=%d\n",a,b);
   getch();
}
```

4. L10_4.C

```
# include <stdio.h>

void invers(int *x, int *y);

void main()
{
   int a=100, b=123;
   printf("\nThe start values for a and b are:\na=%d\nb=%d\n",a,b);
   invers(&a, &b);
   printf("\nThe computed values for a and b are:\na=%d\nb=%d\n",a,b);
   getch();
}

void invers(int *x, int *y)
{
   int s;
   s = *x;
   *x = *y;
   *y = s;
}
```

5. L10_5.C

```
# include <stdio.h>
# include <alloc.h>

void main()
{
  int i, *sir;
  sir = (int*)calloc(3,sizeof(int));
  *sir = 10;
  *(sir+1) = 20;
  *(sir+2) = 30;
  printf("\n\nAt the adresses:");
  for(i=0; i<3; i++)
        printf("\n%p", (sir+i));
  printf("\nWe have the values:");
  for(i=0; i<3; i++)
        printf("\n%d", *(sir+i));
  getch();
}
```

6. L10_6.C

```
# include <stdio.h>

void citire(float *p);

void main()
{
  float x;
  citire(&x);
  printf("\nThe value assigned to x is: %g",x);
  getch();
}

void citire(float *p)
{
   printf("\n\nPlease introduce a real number: ");
   scanf("%f",p);
}
```