Example 19

1. Write the program which will enter a short text (1 line) from the keyboard, reverse the order of the letters of the text and print the reversed text on the screen.
2. Separate the function in the program which reverses the text in the place.


```
#include <stdio.h>
#include <string.h>
/* program reversing the text in the place */
int main(void)
{
   int  c, i, j, n;
   char s[81];
   gets(s);
   n=strlen(s); //returns the length of the text without '\0'
   for (i=0, j=n-1; i<j; i++, j--)
   {
        c=s[i];   s[i]=s[j];   s[j]=c;
   }
   printf(" n= %d\n",n);
   printf(" %s\n",s);
   return 0;
}
```

Function „*reverse*”

```
# include <string.h>
/* function reversing the text in the place */
void reverse(char s[ ])
{
   int c, i, j;
   for (i=0, j=strlen(s)-1; i<j; i++, j--)
   {
        c=s[i];   s[i]=s[j];   s[j]=c;
   }
}
```

134

```
/* program reversing the text in the place */
#include <stdio.h>
#include <string.h>

void reverse(char s[ ]); // function prototype - declaration

int main(void)
{
   char stab[81];
   scanf("%80s", &stab);
   fflush(stdin);
   reverse(stab);
   printf("%s\n",stab);
   return 0;
}
```

formal parameter

actual parameter - argument

```
/* function reversing the text in the place */
void reverse(char s[ ]) // function definition
{
   int c, i, j;
   for (i=0, j=strlen(s)-1; i<j; i++, j--)
   {
        c=s[i];   s[i]=s[j];   s[j]=c;
   }
}
```

Important!
While the **reverse** function will finish, **the content of the table s will change !!!**

136

Example 20

Problem:
Write a program entering measured data sets into an array and finding the maximum data in the each of 10 series containing 50 measurements.

Discussion:
Input data: 10 series of 50 measurements
Output data: maximum in each set

Algorithm:
Divide the program into modules. Write a procedure entering the data into an array and a function calculating the maximum value in each measurement set.

137

```
#include <stdio.h>
#define NSETS 10
#define NMEAS 50
typedef float series[NMEAS];  //measurements in one series
typedef series all_series[NSETS];  //all series
void ReadSeries(all_series tab);  //ReadSeries function declaration
float Maximum(series tab); //Maximum function declaration
int main() { //main part of the program
 int nrs;
 float maxpom[NSETS];
 all_series measurements;
 ReadSeries(measurements);
 printf("\n\n");
 for(nrs=0; nrs<NSETS; nrs++) {
  maxpom[nrs]=Maximum(measurements[nrs]);
  printf("The highest measurement in the series %d is %8.3f\n", nrs,
    maxpom[nrs]);
 }
 getch();
 return 0;
} //the end of the main part of the program
```

```
 void ReadSeries(all_series tab) {
 /*Function which enters the data series into the array*/
  int nrs, nrm;
  for(nrs=0; nrs<NSETS; nrs++) { //read successive series
   printf("Series %d\n", nrs+1);
   for(nrm=0; nrm<NMEAS; nrm++) { //read successive measurement in series
     printf("Give the data %d: ", nrm);
     scanf("%f", &tab[nrs][nrm]);
   }
  }
 }
 float Maximum(series tab) {
  /*Function for finding the maximum value in the series*/
  int n;
  float max;
  max=tab[0];  // maximum calculation algorithm
  for(n=1; n<NMEAS; n++) {
   if(tab[n]>max)
    max=tab[n];
  }
 return max;
 }
```

Example 21

Problem:

Write a program for tabulation of the power function: $f(x) = x^n$, for real x, n and $x > 0$ in the range $< x_p , x_k >$ for evenly tabulated 200 function values.

Discussion:

Input data: 200 values of x

Output data: 200 pairs of x values and corresponding f(x)

Algorithm:

Because $x > 0$, the algorithm is based on the equation:

$$x^n = e^{\ln(x^n)} = e^{n\ln(x)}$$

140

```
/*Program for function discritization*/

#include <stdio.h>
#include <math.h>

/*constants definition*/
#define LP 200
#define N 1.5

/*definition of the new data type*/
typedef float tab[LP][2];

/*functions declarations*/
float f(float x, const float n);
void tabulation(float xpocz, float xkon, float krok, tab t);
```

141

```c
int main() {                        /*the main part of the program*/
 int i;
 tab fx;
 float xp, xk, dx;
 printf("Program for tabulation of the function \n");
 printf("Give in the sequence the beginning and the end of the
     range of x argument (x1, x2, for x > 0 and x1<x2)\n ");
scanf("%f %f", &xp, &xk);
 if((xp>0)&&(xk>0)&&(xk>xp)) {
  dx=(xk-xp)/(LP-1);
  tabulation(xp, xk, dx, fx);          // call of function for tabulation
  printf("\n\nFunction table f(x):\n");
  printf("x: \t\t f(x):\n");
  for (i=0; i<LP; i++)
   printf("%f \t\t %5.8f\n", fx[i][0], fx[i][1]);
 }
 else printf("Incorrect range of data\n");
 getch();
 return 0;
}
```

42

```c
float f(float x, const float n) {
 float y;
 y=exp(n*log(x));
 return y;
}
void tabulation(float xpocz, float xkon, float krok, tab t) {
 int i;
 float x;
 i = 0;
 x = xpocz;
 while(x<=xkon) {
  t[i][0]=x;
  t[i][1]=f(x,N);
  x=xpocz+krok*i;
  i=i+1;
 }
 if (x<(xkon+krok)) {
  t[i-1][0]=xkon;
  t[i-1][1]=f(xkon,N);
  }
}
```

/*Function definition*/

143

Control statements

- The C language includes the whole set of the control instructions, which enables writing the programs in accordance with the idea of structural programming.
- Their operation enables to switch the control between the instructions which occurs in the complex program and all of them allow to write the concise and logically constructed code.

144

Control statements

- Conditional Instruction **if**
- Instruction **while**
- Instruction **do … while**
- Instruction **for**
- Instruction **switch**
- Instruction **break**
- Instruction **continue**
- Instruction **goto**

145

The if instruction – ambiguity of the syntax

if (expression) statement_1 [else statement_ 2]

Examples:

1.
```
if (a!=0 && b!=0)
    if (a>b)
      x = 4*a;
    else
      x = 4 *b;
```

2.
```
if (a!=0 && b!=0) {
    if (a>b)
    x = 4*a;
  }
  else
    x = 4 *b;
```

else option always applies to the last if !

146

else if construction

```
if (expression) statement_1
else  if (expression) statement_2
      else if (expression) statement_3
      …
            else statement_n
```

else if construction enables the multi-option choice.

147

Example:

```
if (dd == 1)
    printf (" Monday\n");
else if (dd == 2)
    printf (" Tuesday\n");
else if (dd == 3)
    printf (" Wednesday\n");
else if (dd == 4)
    printf („Thursday\n");
else if (dd == 5)
    printf („Friday\n");
else if (dd == 6)
    printf („Saturday\n");
else if (dd == 7)
    printf („Sunday\n");
else printf („ERROR\n");
```

148

The switch statement

```
switch (expression)
{
   case fixed- expression : statements
   …
   case fixed- expression : statements
   default: statements
}
```

149

Example:

```
switch (dd)
{
   case 1: printf (" Monday\n");
           break;
   case 2: printf (" Tuesday\n");
           break;
   case 3: printf (" Wednesday\n");
           break;
   case 4: printf (" Thursday\n");
           break;
   case 5: printf (" Friday\n");
           break;
   case 6: printf (" Saturday\n");
           break;
   case 7: printf (" Sunday\n");
           break;
   default: printf (" ERROR\n");
           break;
}
```

150

Example 22

Write program, which calculates the number of the occurrence of each number, of each white character and all the other characters for the data entered from the keyboard.

Apply *switch* instruction.

151

Program 22

```
# include <stdio.h>
/* program which counts numbers,
white characters and others*/

int main (void)
{
   int c, i, nwhite = 0, nother = 0;
   int ndigit [10] = {0};    // array initialization
```

152

```
   while ((c = getchar ()) != EOF) {
       switch (c)
       { case '0': case '1': case '2': case '3': case '4':
         case '5': case '6': case '7': case '8': case '9':
              ndigit [c-'0'] ++;
              break;
         case ' ': case '\n': case '\t':
              nwhite ++;
              break;
         default:  nother ++;
                   break;
       }
   }
```

153

```
printf(”The number of occurrence of numbers:\n”);
for (i =0; i <10; ++i)
        printf(”%d - %d,\n”, i, ndigit[i]);
printf(”white character = %d\n other = d\n”, nwhite,
   nother);
return 0;
}
```

154

Example 23

Problem:

Check if the entered character is vowel, polish consonant or one of the foreign letter: q, x, v. Revise checking the letters as long as the user requires this.

Discussion:

Input data: letter

Output data: letter and the information about the letter

A chain of multiple choices - switch-case statement is optional.

Algorithm:

1. Read a character and print it on a screen
2. Check, if it is vowel, consonant or the other character
3. Print the result
4. Repeat steps (1-3) if the user press 'y' or 'Y'

155

```c
#include <stdio.h>                                  /*Program for recognising the letter*/
int main() {
   char letter;                                    //letter
   printf("Program for recognising the letters\n");
   do {                                            //beginning of the do-while loop
    printf("\n Enter a letter: ");
    letter = getch();
    printf("\n You have entered the letter: %c", letter);
    switch(letter) {                               //beginning of the switch statement
     case 'a': case 'e': case 'i': case 'o': case 'u': case 'y':
         printf("\n vowel\n");
         break;
     case 'b': case 'c': case 'd': case 'f': case 'g': case 'h': case 'j': case 'k': case 'l':
     case 'm': case 'n': case 'p': case 'r': case 's': case 't': case 'w': case 'z':
         printf("\n consonant\n");
         break;
     case 'q': case 'v': case 'x':
         printf("\n foreign vowel \n");
         break;
     default:
         printf("\n it was not a letter\n");
    }                                              //end of the switch instruction
    printf("\n If to continue (t/n)? ");
    letter=getchar();
   }while((letter=='T')||(letter=='t'));           //end of the do-while loop
   return 0;
}
```

The break statement

Break instruction is used to break the activity of the loop and the *switch* instruction.

Example:

```c
int i, m;
int length_of_the_line= 5;
for (i=0; i < 4; i++) {
    for (m=0; m<10; m++) {
          printf("*");
          if (m > length_of_the_line) break;
    }
    printf("\nContinuation for the external loop for i = %d\n", i);
}
```

157

goto instruction and labels

goto name_of_the_label;

goto instruction is used for breaking repeatedly nested loops or jumping to the pointed out program statement.

goto instruction is not recommended !!!

Example:

```
for (…)
    for (…) {
            …
            if (error)
                    goto error; // go to the operation of errors
    }
…
error: printf("Blad");
```

158

Example 24

Write the program which checks if the entered number from the keyboard is the prime number.

159

```c
/* program for recognising prime numbers */
#include <stdio.h>
#include <math.h>

int prime (int);

int main (void)
{
        int n,p;
        do {
        printf ("Enter natural number, n>0, n= ");
        scanf ("%d", &n);
        } while (n<= 0);
        p = prime (n);
        if (p) printf("\nNumber %d is not the prime number\n", n);
        else printf("\nNumber %d is the prime number\n", n);
return 0;
}
```

160

```c
int prime (int n)
{
        int i;
        int p=0;

        for (i=2; i <= sqrt (n); i++)
                if (n % i == 0)
                {
                        p=1;
                        break;
                }
        return p;
}
```

161

The continue statement

The continue statement causes an immediate return to the head of a loop, e.g. if a certain condition is implemented.

Example:

```
for (i=0; i<n; i++) {
   if (a[i] < 0) // exclude the negative element
        continue;
   …  // convert the non-negative element
}
```

162