vladaf@matf.bg.ac.rs

Funkcije i struktura programa

vladaf@matf.bg.ac.rs

4.1 Osnovne funkcije

Ah Love! could you and I with Fate conspire
To grasp this sorry Scheme of Things entire,
Would not we shatter it to bits -- and then
Re-mould it nearer to the Heart's Desire!

Ah Love! could you and I with Fate conspire
Would not we shatter it to bits -- and then
Re-mould it nearer to the Heart's Desire!

```
while (postoji jos jedan red)
    if (taj red sadrzi obrazac)
        prikazati ga
```

```
tip-rezultata ime-funkcije (deklaracije argumenata)
{
    deklaracije i naredbe
}
```

```
#include <stdio.h>
#define MAXLINE 1000 /* maximum input line length */
int getline(char line[], int max);
int strindex(char source[], char searchfor[]);

char pattern[] = "ould";   /* pattern to search for */
/* find all lines matching pattern */
main()
{
    char line[MAXLINE];
    int found = 0;

    while (getline(line, MAXLINE) > 0)
        if (strindex(line, pattern) >= 0) {
            printf("%s", line);
            found++;
        }
    return found;
}
/* getline:  get line into s, return length */
int getline(char s[], int lim)
{
    int c, i;

    i = 0;
    while (--lim > 0 && (c=getchar()) != EOF && c != '\n')
        s[i++] = c;
    if (c == '\n')
        s[i++] = c;
    s[i] = '\0';
    return i;
}
/* strindex:  return index of t in s, -1 if none */
int strindex(char s[], char t[])
{
    int i, j, k;
    for (i = 0; s[i] != '\0'; i++) {
        for (j=i, k=0; t[k]!='\0' && s[j]==t[k]; j++, k++)
            ;
        if (k > 0 && t[k] == '\0')
            return i;
    }
    return -1;
}
```

vladaf@matf.bg.ac.rs

dummy() {}

return izraz;

cc main.c getline.c strindex.c

cc main.c getline.o strindex.o

vladaf@matf.bg.ac.rs

4.2 Funkcije koje ne vraćaju celobrojne vrednosti

```
#include <ctype.h>

/* atof:  convert string s to double */
double atof(char s[ ])
{
    double val, power;
    int i, sign;

    for (i = 0; isspace(s[i]); i++)  /* skip white space */
        ;
    sign = (s[i] == '-') ? -1 : 1;
    if (s[i] == '+' || s[i] == '-')
        i++;
    for (val = 0.0; isdigit(s[i]); i++)
        val = 10.0 * val + (s[i] - '0');
    if (s[i] == '.')
        i++;
    for (power = 1.0; isdigit(s[i]); i++) {
        val = 10.0 * val + (s[i] - '0');
        power *= 10.0;
    }
    return sign * val / power;
}
```

```
#include <stdio.h>

#define MAXLINE 100

/* rudimentary calculator */
main()
{
    double sum, atof(char []);
    char line[MAXLINE];
    int getline(char line[], int max);

    sum = 0;
    while (getline(line, MAXLINE) > 0)
        printf("\t%g\n", sum += atof(line));
    return 0;
}
```

```
/* atoi:  convert string s to integer using atof */
int atoi(char s[])
{
    double atof(char s[]);
    return (int) atof(s);
}
```

```
double sum, atof(char []);

sum += atof(line)

double atof();
```

vladaf@matf.bg.ac.rs

4.3 Spoljašnje promenljive

(1 – 2) * (4 + 5)

1 2 – 4 5 + *

```
while (naredni operator ili operand nije indikator kraja datoteke)
    if (broj)
        dodati ga na stek
    else if (operator)
        ukloniti operande sa steka
        izvršiti operaciju
        dodati rezultat na stek
    else if (novi red)
        ukloniti i prikazati vrednost na vrhu steka
    else
        greška
```

```
#includes
#defines

function declarations for main
main() { … }

external variables for push and pop

void push(double f) { … }
double pop(void) { … }

int getop(char s[]) { … }

routines called by getop
```

vladaf@matf.bg.ac.rs

```
#include <stdio.h>
#include <stdlib.h>  /* for  atof() */

#define MAXOP   100  /* max size of operand or operator */
#define NUMBER  '0'  /* signal that a number was found */

int getop(char []);
void push(double);
double pop(void);
```

```
/* reverse Polish calculator */
   main()
   {
      int type;
      double op2;
      char s[MAXOP];

      while ((type = getop(s)) != EOF) {
         switch (type) {
         case NUMBER:
            push(atof(s));
            break;
         case '+':
            push(pop() + pop());
            break;
         case '*':
            push(pop() * pop());
            break;
         case '-':
            op2 = pop();
            push(pop() - op2);
            break;
         case '/':
            op2 = pop();
            if (op2 != 0.0)
               push(pop() / op2);
            else
               printf("error: zero divisor\n");
            break;
         case '\n':
            printf("\t%.8g\n", pop());
            break;
         default:
            printf("error: unknown command %s\n", s);
            break;
         }
      }
      return 0;
   }
```

```
push(pop() – pop());   /* POGREŠNO */
```

```
#define MAXVAL  100          /* maximum depth of val stack */

int sp = 0;                  /* next free stack position */
double val[MAXVAL];          /* value stack */

/* push:  push f onto value stack */
void push(double f)
{
    if (sp < MAXVAL)
        val[sp++] = f;
    else
        printf("error: stack full, can't push %g\n", f);
}


/* pop:  pop and return top value from stack */
double pop(void)
{
    if (sp > 0)
        return val[--sp];
    else {
        printf("error: stack empty\n");
        return 0.0;
    }
}
```

vladaf@matf.bg.ac.rs

```
#include <ctype.h>

int getch(void);
void ungetch(int);

/* getop:  get next character or numeric operand */
int getop(char s[ ])
{
    int i, c;

    while ((s[0] = c = getch()) == ' ' || c == '\t')
        ;
    s[1] = '\0';
    if (!isdigit(c) && c != '.')
        return c;      /* not a number */
    i = 0;
    if (isdigit(c))    /* collect integer part */
        while (isdigit(s[++i] = c = getch()))
            ;
    if (c == '.')      /* collect fraction part */
        while (isdigit(s[++i] = c = getch()))
            ;
    s[i] = '\0';
    if (c != EOF)
        ungetch(c);
    return NUMBER;
}
```

vladaf@matf.bg.ac.rs

```
#define BUFSIZE 100

char buf[BUFSIZE];              /* buffer for ungetch */
int bufp = 0;                   /* next free position in buf */

int getch(void)  /* get a (possibly pushed-back) character */
{
    return (bufp > 0) ? buf[--bufp] : getchar();
}

void ungetch(int c)   /* push character back on input */
{
    if (bufp >= BUFSIZE)
        printf("ungetch: too many characters\n");
    else
        buf[bufp++] = c;
}
```

vladaf@matf.bg.ac.rs

4.4 Pravila dometa

```
main() { … }

int sp = 0;
double val[MAXVAL];

void push(double f) { … }

double pop(void) { … }
```

```
int sp;
double val[MAXVAL];
```

```
extern int sp;
extern double val[];
```

u datoteci 1

```
extern int sp;
extern double val[];

void push(double f) { … }

double pop(void) { … }
```

u datoteci 2

```
int sp = 0;
double val[MAXVAL];
```

vladaf@matf.bg.ac.rs

4.5 Datoteke zaglavlja

calc.h:

```
#define NUMBER '0'
void push(double);
double pop(void);
int getop(char [ ]);
int getch(void);
void ungetch(int);
```

main.c:

```
#include <stdio.h>
#include <stlib.h>
#include "calc.h"
#define MAXOP 100
main() {
        ...
}
```

getop.c:

```
#include <stdio.h>
#include <ctype.h>
#include "calc.h"
getop() {
        ...
}
```

getch.c:

```
#include <stdio.h>
#define BUFSIZE 100
char buf[BUFSIZE];
int bufp = 0;
int getchar(void) {
      ...
}
void ungetch(int) {
      ...
}
```

stack.c:

```
# include <stdio.h>
#include "calc.h"
#define MAXVAL 100
int sp = 0;
double val[MAXVAL];
void push(double) {
        ...
}
double pop(void) {
        ...
}
```

vladaf@matf.bg.ac.rs

4.6 Statičke promenljive

```
static char buf[BUFSIZE];          /* buffer for ungetch */
static int bufp = 0;               /* next free position in buf */

int getch(void) { … }

void ungetch(int c) { … }
```

vladaf@matf.bg.ac.rs

4.7 Registarske promenljive

```
register int x;
register char c;


f(register unsigned m, register long n)
{
        register int i;
        …
}
```

vladaf@matf.bg.ac.rs

4.8 Blokovska struktura

```
if (n > 0) {
    int i;              /* deklarisanje nove promenljive i */
    for (i = 0; i < n; i++)
            …
}
```

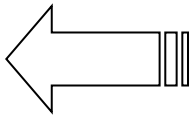
```
int x;
int y;

f(double x)
{
        double y;
        …
}
```

vladaf@matf.bg.ac.rs

4.9 Inicijalizacija

```
int x = 1;
char squote =' \ ' ';
long day = 1000L * 60L * 60L * 24L; /* broj milisekundi u danu */
```

```
int binsearch(int x, int v[ ], int n)
{
        int low = 0;
        int high = n – 1;
        int mid;
        …
}
```



```
int low, high, mid;
low = 0;
high = n – 1;
```

```
int days[ ] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

```
char pattern[ ] = "ould";
```

```
char pattern[ ] = {'o', 'u', 'l', 'd', '\0'};
```

4.10 Rekurzija

```
#include <stdio.h>

/* printd:  print n in decimal */
void printd(int n)
{
    if (n < 0) {
        putchar('-');
        n = -n;
    }
    if (n / 10)
        printd(n / 10);
    putchar(n % 10 + '0');
}
```



```
/*
 * rfakt.c                                  vladaf@matf.bg.ac.rs
 * Izracunavanje faktorijela nekog broja primenom rekurzije.
 */
#include <stdio.h>
#include <stdlib.h>
#define BRCIF 10
long faktorijel(short);
int main (void)
{
    char ulaz[BRCIF+1];                     /* ulazni bafer */
    short broj;                             /* polazni broj */
    long rezultat;                          /* rezultat */
    /* pitaj korisnika za ulazne podatke i ucitaj ih */
    printf("Otkucajte nenegativan ceo broj + RETURN: ");
    gets(ulaz);
    broj = (short) atoi(ulaz);
    /* rekurzijom izracunaj faktorijel i prikazi ga */
    rezultat = faktorijel(broj);
    printf("Faktorijel broja %hd je %ld\n", broj, rezultat);
    return EXIT_SUCCESS;
}
long faktorijel(short n)                    /* rekurzivna verzija */
{
                if (n <= 1)
                        return 1L;
                else
                        return (n * faktorijel(n-1));
}
```

vladaf@matf.bg.ac.rs

```
/* qsort:  sort v[left]...v[right] into increasing order */
void qsort(int v[], int left, int right)
{
    int i, last;
    void swap(int v[], int i, int j);

    if (left >= right)       /* do nothing if array contains */
        return;              /* fewer than two elements */
    swap(v, left, (left + right)/2);     /* move partition elem */
    last = left;                                    /* to v[0] */
    for (i = left + 1; i <= right; i++)  /* partition */
        if (v[i] < v[left])
            swap(v, ++last, i);
    swap(v, left, last);                            /* restore partition  elem */
    qsort(v, left, last-1);
    qsort(v, last+1, right);
}
```

```
/* swap:  interchange v[i] and v[j] */
void swap(int v[], int i, int j)
{
    int temp;

    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}
```

vladaf@matf.bg.ac.rs

4.11 C pretprocesor

#include

#define

uslovno kompajliranje

makroi sa argumentima

vladaf@matf.bg.ac.rs

4.11.1 Uključivanje datoteka

```
#include "imedatoteke"

#include <imedatoteke>
```

vladaf@matf.bg.ac.rs

4.11.2 Zamena makroa

#define *ime tekst zamene*

```
#define  forever  for (;;)          /* beskonačna petlja */

#define  max(A, B)  ((A) > (B) ? (A) : (B))

x = max(p+q, r+s);

x = ((p+q) > (r+s) ? (p+q) : (r+s));

max(i++, j++);   /* POGREŠNO */


#define  square(x)  x * x   /* POGREŠNO */


#undef getchar
int getchar(void) { … }
```

vladaf@matf.bg.ac.rs

```
#define dprint(expr)  printf(#expr " = %g\n", expr)

dprint(x/y);

printf("x/y" " = %g\n", x/y);

printf("x/y = %g\n", x/y);
```

```
#define paste(front, back) front ## back

 paste(name, 1)          kreira     name1
```

vladaf@matf.bg.ac.rs

4.11.3 Uslovno uključivanje datoteka

```
#if !defined(HDR)
#define HDR

/* sadržaj datoteke hdr.h ide ovde - ukljucuje se samo jednom */

#endif
```

```
#if SYSTEM == SYSV
        #define HDR "sysv.h"
#elif SYSTEM == BSD
        #define HDR "bsd.h"
#elif SYSTEM == MSDOS
        #define HDR "msdos.h"
#else
        #define HDR "default.h"
#endif
#include HDR
```

```
#ifndef HDR
#define HDR

/* sadržaj datoteke hdr.h ide ovde */

#endif
```