Język **ANSI C**

dr inż. Przemysław Śliwiński

7 października 2003

1 Kalkulator

Tematem prac w ramach laboratorium jest skonstruowanie **kalkulatora**.

Ocena	Funkcje
dst	**+ , - , * , / ,** (dla liczb całkowitych)
dst +	**jw. +** obsługa nawiasów ogrągłych, klamrowych etc.
db	**jw. +** działania na liczbach zmiennoprzecinkowych
db +	**jw. +** obsługa funkcji trygonometrycznych
bdb	**jw. +** obsługa zmiennych
cel	**jw. +** obsługa własnych funkcji

Przykłady:

$1+2$	$\Longrightarrow$	3
$(2-2)*5$	$\Longrightarrow$	0
$2.5*2.5$	$\Longrightarrow$	6.25
$\sin(1)*\sin(1)+\cos(1)*\cos(1)$	$\Longrightarrow$	1
$\mathbf{brutto}=2000; \mathbf{ZUS}=20.33\%; \mathbf{Podatek}=19\%;$ $\mathbf{netto}=\mathbf{brutto}*(1-\mathbf{ZUS})*(1-\mathbf{Podatek}); \mathbf{netto};$	$\Longrightarrow$	1290,65
$f(x)=1/x; f(2);$	$\Longrightarrow$	0,5

Pytania – podpowiedzi:[1]

1. **lex/yacc**: Co to są wyrażenia regularne?
2. **lex/yacc**: Do czego służą analizatory leksykalne i składni?
3. **(C/C++):** Czy do generowania szkieletu programu warto użyć narzędzi **lex/yacc** (**flex/bison**)?
4. **(C++):** A do przechowywania zmiennych, czy wygodnie jest skorzystać z szablonu **map** z biblioteki **STL**?

[1] `http://www.google.pl/search?hl=pl&ie=UTF-8&oe=UTF-8&q=yacc+lex+kalkulator&btnG=Szukaj+z+Google&lr=`

2 Kalkulator - szkielet (lex)

```
%{
#include <stdio.h>     /* C declarations used in actions */

void print_result(double);

%}

%union {double a_number;}         /* Yacc definitions */
%start line
%token <a_number> NUMBER
%type  <a_number> exp term factor

%left '+'
%left '*'

%%
/* descriptions of expected inputs     corresponding actions (in C) */

line   : exp '='                       {
                                          print_result($1);
                                          return 0;
                                       }
       ;
exp    : term                          {  $$ = $1;          }
       | exp '+' term                  {  $$ = $1 + $3;     }
       ;
term   : factor                        {  $$ = $1;          }
       | term '*' factor               {  $$ = $1 * $3;     }
       ;
factor : NUMBER                        {  $$ = $1;          }
       ;
%%                     /* C code */
int main()
{
   return yyparse ( );
}
int yyerror(char *s)
{
   fprintf(stderr, ''%s\n'', s);
}
void print_result(double result)
{
   printf ('' %ld\n'', result);
}
```

3 Kalkulator - szkielet (yacc)

```
%{
#include ''y.tab.h''
char error_msg[] = ''Ten kalklulator nie radzi sobie z literami...;('';

%}
%%

[0-9]+          {yylval.a_number = atof(yytext); return NUMBER;      }
[ \t\n]         ;
[+*=]           {                                return yytext[0];   }
.               {printf (''%s\n'', yytext); yyerror (error_msg);}

%%
int yywrap (void)
{
   return 1;
}
```