Pascal *contra* ***C++***

(* kommentaar *)	/* kommentaar */ // C++ kuni rea lõpuni!
var x: real; n: integer; c: character; rida: string(40); m: array [0..10] of integer;	float x; int n; char c; char rida[41]; int m[11];

begin
k := 1; j := 0;
while k <> 0 do
begin
if k > 7
then j := j-1
else j := j+1;
end;
end;
	{
	k=1; j=0;
	while(k) // while(k != 0)
	{
	if(k>7)
	j--
	else j++;
	};
	};

for j:=1 to 10 do m[j] := m[j] + 3;	for(j=1; j<=10; j++) m[j]+=3;

repeat
j := k mod 7;
until j<>0;
	do {
	j = k % 7; }
	while(j==0); // while(!j);

case rida[2] of
'x': k := k / j;
'y': j := j * k;
else k = k – 1;
end;
	switch(rida[1])
	{ case 'x': k/=j; break;
	case 'y': j*=k; break;
	default: k--;
	};

C:\CPLUS\BIN\TC.EXE failide nimed: ***.CPP**

```
#include  <stdio.h>                    //  standard  input-output

int  main( void )                     //  põhiprogramm
{
      printf(  “Hello, World! \n” );  //  writeln( “……” );
      return  0;
}
```

C

Brian W. Kernighan, Dennis M. Ritchie. The C Programming Language, 1978.

Viktor Leppikson

C++

Bjarne Stroutstrup. The C++ Programming Language, 1986.

Rein Jürgenson, Teodor Luczkowski

Konstandid

“See on string” ‘a’ // see on üks sümbol
‘\n’ // reavahetus – ka stringi sees!

0112 // octal integer 0x3F // hex integer
567389056L // Long decimal integer

Tüübid

char sümbol või sümbolite massiiv (string); interpreteeritakse kui täisarvu -128..127

int 2-baidine täisarv (ehk **short**)
long 4-baidine täisarv

unsigned ***midagi*** (kus midagi on kas char, int, short või long) – märgita naturaalarv, näit. *unsigned char* on vahemikus 0..255

float **double** reaalarvud

enum loendustüüp

void "tüübita" (või ükskõik mis tüüpi) asi

near **long** **huge** viitmuutujate (aadressite) erikujud

sizeof(*midagi*) midagi-tüüpi muutuja pikkus baitides

Kirjeldused

```
const  PII=3.14159,  Nmax=99;

long  x,y;                              // 4-baidised täisarvud

unsigned  char  u,v;    //   0..255

int  vektor[20],  maatriks[7][12];          //  indeksid alati 0..N-1!
char  s6na[9];                              // string  max 8 sümbolit

char  *p;      // viitmuutuja mingile sümbolile  (ka stringi sees!)
```

// puudub Boolean tüüp!!! Avaldis on **tõene**, kui ta on
// nullist erinev!

// saab defineerida uue tüübi

typedef unsigned char uchar; // *uchar* on uus tüüp

typedef int[Nmax] vektor; // nagu ka *vektor*

enum Mast = { Risti, Ruutu, Ärtu, Pada }; // loendustüüp

// muutuja kirjeldamisel võib talle omistada ka *algväärtuse!*

int m=4, n=-1; float pii=3.14159;

char tervitus[] = “Hello, World!”;

double t[3] = { 3.14159, 2.718282, -1e-7 };

// muutujad tuleb kirjeldada enne nende kasutamist, kuid mitte
// tingimatta ploki alguses**!!!**

Relatsioonid avaldistes

< <= **==** **!=** >= >

Operatsioonid avaldistes

( )	funktsiooni väljakutse
[ ]	massiivi element
.	eraldaja liitnimes
- >	osutamine viitmuutuja kaudu struktuuri osale

!	loogiline eitus
~	bithaaval eitus
-	unaarne miinus
++	auto-inkrement
--	auto-dekrement
&	(muutuja) aadressi võtmine
*	viitamine viitmuutuja väärtuse kaudu
(tüüp)	tüübiteisendus
sizeof	objekti pikkus baitides

*	korrutamine
/	jagamine
%	modulo
+	liitmine
-	lahutamine
<<	nihe bithaaval vasakule (korrutamine 2-ga)
>>	nihe bithaaval paremale (mod 2)
&	bithaaval konjuktsioon
^	bithaaval välistav disjunktsioon
|	bithaaval disjunktsioon
&&	konjuktsioon
| |	disjunktsioon

<avaldis> **?** <kui tõene> **:** <kui väär> // IF-avaldis

Omistamine

= *= /= %= += -= <<= >>= &= ^= |=

kus a = b tähendab sama, mis a = a b

NB! Omistamised on *avaldised*, mille väärtuseks on omistatud väärtus.

Operaatorid

{ }	liitoperaator
goto L	märgend kujul L:
break	väljumine lähimast *while*, *do*, *for* või *switch* operaatorist
continue	goto lähima tsükli algusesse (tsükli järgmine kordus)
return	väljumine protseduurist
return *avaldis*	väljumine funktsioonist

Funktsioonid

<väärtuse tüüp> <nimi> **(** <parameetrite loetelu> **)**
{ <kirjeldused ja operaatorid> **}**

Kui <väärtuse tüüp> == *void* siis protseduur
Kui <parameetrite loetelu> == *void* siis parameetreid pole

Parameetrid edastatakse väärtuse järgi. Et edastamine toimuks aadressi järgi, tuleb formaalse parameetri nime ette panna **&** (Pascal'is *var*)

Kui funktsiooni lokaalse muutuja kirjelduses esineb võtmesõna **static**, siis sellise muutuja väärtus säilub järgmise pöördumiseni

```
void  proge( void )   // väljastab tema poole pöördumise jrk.nr.
{
        static int mitmes = 0;
        mitmes++;
        cout  <<  mitmes  <<  ".pöördumine\n";
}
```

Struktuurid

Kirjetüübi kirjeldamine:

struct <tüübi_nimi> **{** <väljade_kirjeldused> **}**

union <tüübi_nimi> **{** <väljade_kirjeldused> **}**

Kirjeldaja *union* korral on kõigil väljadel ühine mälu!

```
union  Bit32
{
        long int  arv;                  //  4-baidine täisarv
        unsigned char bait[ 4 ];        // massiiv neljast baidist
}
```

arv ==	bait[3]	bait[2]	bait[1]	bait[0]

```
Bit32  pikk;
     pikk.arv = 0;     pikk.bait[2] = 0x2A;

Bit32  *viit;
     viit = &pikk;     viit -> bait[1] = 103;
```