Паскал vs. C++

Краток историјат

Паскал е еден од постарите програмски јазици и според некои луѓе еден од "најпопуларните".

Паскал е креиран од страна на швајцарскиот професор Никлаус Вирд во седумдесетите години, а го именувал во чест на фрацускиот математичар и физичар Блеиз Паскал, кој пак ја има конструирано првата машина за собирање.

Во 1980 година е конструиран објектниот јазик С наречен "С со класа", кој работи на многу ниско ниво и е наменет за креирање на оперативни системи и компајлери. Токму на таа основа во 1983 година се раѓа C++ варијанта на С со една голема разлика во својата стабилност во работата.

Како за почеток најголемата разлика помеѓу паскал и C++ е тоа што паскал е структурен (некаде се дефинира како процедурален) јазик а C++ е објектно-ориентиран јазик. Паскал ја доживува својата експанзија токму поради тоа што тој бил првиот јазик кој овозможува структурно програмирање.

Краток опис

Структурното програмирање (прецедуралното програмирање) на паскал му диктира пат на работа со модули, кои пак се посебни независни делови а секој од нив реализира различна функција. Модулот мора да им оперделен почеток и крај. Модулите можат да се менуваат и тестираат независно еден од друг а се повикуваат според пропишана редослед. Недостатокот на процедуралното програмирање е тоа што фукциите и податоците се раздвоени.

Објектно-ориентираното програмирање е пак од друга страна по дефиниција значи: манипулација со објекти низ програмирањето. Па токму оваа дефиниција го носи терминот објектно-ориентирано програмирање. Кај OOJ имплементиран е апстрактен тип на податоци преку концептот наречен "КЛАСИ". Таа апстракција всушност е комбинирање на податоците со функциите, со користење на класи. Огромна предност пред структурните јазици е тоа што може да се издвојат делови од програмите како објекти и да се користат во други програми, било да се надополнуваат или не. ОО-јазиците овозможуваат и лесна модификација на постоечките програми без ништо да се менува. Ова се прави со помош на:

- Наследување и
- Полиморфизам

"Преклопувањето на оператори" е уште една од многуте добри собини на ОО-јазиците.

СЕ ШТО МОЖЕ ДА СЕ НАПРАВИ ВО ПАСКАЛ МОЖЕ ДА СЕ НАПРАВИ И ВО C++, ОБРАТНОТО ТВРДЕЊЕ НЕ ВАЖИ.

Почетни синтаксички разлики

Како прво и основно паскал не е “case-sensitive“ додека C++ е “case-sensitive“.

Кога разгледуваме програма во паскал и програма направена во C++ уште на самиот почеток имаме видливи синтаксички разлики како што се заглавје на програми, дефинирање на променливи, функции и процедури наспроти класи, запишување коментари, структурни наредби, наредби за печатење, циклуси за повторување, аритметички операции итн. Сега со мала демонстрација на програмски код и од двата програмски јазици ќе ја утврдиме разликата.

На самиот врв на програмата во паксал дефинираме име на програмата, од друга страна во C++ тоа не е потребно тоа се изведува на поинаков начин што ќе го покажеме со следниов мал и наједноставен пример:

Паскал код:

```
program PechatenjeIme
 begin
   writeln (' Marjan ');
   {Запишување на коментар}
 end.
```

C++ код:

```
#include
 int main()
 {
   cout << "Marjan" << endl;
   // запишување на коментар
 }
```

Двата кода извршуваат потполно иста работа разликата е само во синтаксата (овие две програмчиња кога би се извршиле на монитор ќе се испише Marjan). Да ги воочиме разликите кај паскал наредбите се пишуваат помеѓу

```
begin
    ;

 end;
```

а кај C++ тоа се прави помеѓу

```
{
    ;
    ;
 }
```

Симболот “;“ е задолжителен после секоја наредба освен пред end во паскал, а во C++ не смее никаде да се изостави.

Аритметички операции

Паскал код:

```
:=   { симбол за доделување на вредност }
+
-
*
/    { "real" делење }
div  { "integer" делење }
mod  { делење по модул}
```

C++ код:

```
=  // симбол за доделување на вредност
+
-
*
/
%  // делење по модул
```

Релациски оператори

Паскал код:

```
=   { еднакво }
   { различно }
<
<=
>
>=
```

C++ код:

```
==   // еднакво
!=   // различно
<
<=
>
```

Логички оператори

Паскал код:

```
and {и}
or  {или}
not {не}
```

C++ код:

```
&& //и
|| //или
!  //не
```

Декларација на променливи, константи и низи: Типови на променливи

Паскал код:

```
·         char
·         integer
·         real
·         boolean
```

C++ код:

```
·         char
·         int
·         float
·         bool
```

Константи:

Паскал код:

```
const
   Pi = 3.14;
   Rate = 0.05;
      { .05 не е дозволува}
   Chash = 3600;
   Denar = 'den';
   Pozdrav  = 'Zdravo';
```

C++ код:

```
 const double Pi =
3.14,
       Rate = .05; //
or 0.05;
 const int Chas = 3600;
 const char Denar =
'den';
 const char Pozdrav[] =
"Zdravo";
 //  Исто така се
дозволува и употреба на
следниов начин
 const double R = 5.,
Pi = 3.14,
    Area = Pi * R * R;
```

Променливи:

Кај паскал променливите и во главната програма и во функциите и процедурите променливите мора да се наведени по пишување на зборот var. Не е дозволено иницијализирање на променливите во нивната декларација.

Паскал код:

```
...
   var
     r : real;
     i, j : integer;
     star : char;
     match : boolean;
...
```

C++ код:

```
  double r = 5.;
   int i = 0, j = i+1;
   char star = '*';
```

Во C++ декларирањето на променливи може да се направи било каде во кодот и тие може да се иницијализираат при нивната декларација. Глобалните променливи може да се декларираат надвор од секоја класа.

Низи

Паскал код:

```
  var
   str : packed array [1..80] of char;
   grid : array [1..32, 1..25] of integer;
```

Резервираниот збор packed е препорачлив за користење на текстуални низи (но ева е отфрлено во пракса бидејќи денешните компјутери располагаат со доволно голема меморија). Првиот член во низата може да почне од било кој број но вообичаен индекс е еден (1).

C++ код:

```
char str[80];
 int grid[32][25];
```

Исто така низите во C++ можат да бидат иницијализирани при нивната декларација:

```
  int fiboBroevi[5] = {1,1,2,3,5};
  char tekst[80] = "Значи, да а";
```

Во C++ индексот на првиот елемент е нула (0), а индексот на последниот елемент е n-1.

Операторот sizeof(. . .)

Паскал код:

```
   Во паскал не постои такво нешто
```

C++ код:

```
Неговата улога е да врати големина во бајти на константи, променливи на пример
sizeof(char)
врќа вредност еден (1).
```

Множества (SETS)

Паскал код:

Пример:

```
if ch in ['0'..'9'] then
   writeln (ch, ' e broj.');
 if ch in ['A'..'Z', 'a'..'z']
  then
    writeln (ch, ' e bukva.');
     ...
```

Компајлерот може да ја лимитира големината на множеството, вообичаено е големината да биде 256.

C++ код:

Не постои такво нешто во C++.

Готови математички функции

Паскал код:

```
abs(x)
sqrt(x)
sin(x)
cos(x)
exp(x)
ln(x)
sqr(x)
arctan(x)
```

C++ код:

```
int abs(int x);
double fabs(double x);
double sqrt(double x);
double sin(double x);
double cos(double x);
double exp(double x);
double log(double x); // Природен log
double pow(double base, double exponent);
double atan(double x);
```

Комбинирани аритметички операции (Инкремент, Декремент)

Паскал код:

Во паскал не постои такво нешто, во паскал овие работи можат да се решат на следниот начин:

```
a := a + 1;
b := a; a := a + 1;
a = :a + 1; b := a;
a := a - 1;
b := a; a := a - 1;
a := a - 1; b := a;
a := a + b;
a := a - b;
a := a * b;
a := a / b;
a := a mod b;
```

C++ код:

Соодветните од оние кај паскал во C++ би ги запишале вака:

```
a++;      //  a := a + 1
b = a++; //  b := a; a := a + 1
b = ++a; //  a := a + 1; b := a
a--;      //  a := a - 1;
b = a--; //  b := a; a := a - 1
b = --a; //  a := a - 1; b := a
a += b;  //  a := a + b
a -= b;  //  a := a - b
a *= b;  //  a := a * b
a /= b;  //  a := a / b
a %= b;  //  a := a % b
```

Влез и Излез

Паскал код:

```
write (x, y, ...);
writeln (x, y);
writeln;
read (x, y, ...);
readln (x, y);
readln;
```

C++ код:

```
cout << x << ' ' << y << ...;
cout << x << ' ' << y << endl;
cout << endl;
cin >> x >> y >> ...;
```

```
cin >> x >> y;
```

Излезни карактери:

Паскал код:

Во паскал не постојат “излезни карактери“ се што се пишува за печатење се пишува помеѓу два наводника

```
writeln ('Prim''er!');
```

C++ код:

Во C++ постојат “излезни карактери“ тие се пишуваат на следниов начин и нивната акција е следна:

```
'\n'  Нова Линија
'\''  Единечен Наводник
'\"'  Наводница
'\\'  Коса црта
'\a'  Аларм
'\t'  Таб
'\r'  Враќање назад
```

Наредби за разгранување(if-else, case - switch)

Паскал код:

```
if  then
   ;
 if  then
    {употреба на ; не е дозволено}
 else
   ;
 if  then begin
   ;
   ;
 end
 else begin
   ;
   ;
 end;
```

C++ код:

```
if (  )
   ;
 if (  )
   ; // задолжителна уптреба на ;
 else
   ;
 if (  ) {
   ;
   ;
 }
 else {
   ;
```

```
   ;
 }
```

case – switch

Паскал код:

```
case  of
   :
     ;
   ...
   :
     ;
 end;
```

Примери:

```
 case ch of
   'A': Add();
   'D': Delete();
   'M': Modify();
   'Q': ; { do nothing }
 end;
 case d of
   1,2: ...;
    99:   ...;
   100:   ...;
 end;
 case age >= 65 of
   true: ...;
   false: ...;
 end;
```

C++ код:

```
switch (  ) {
   case :
     ; break;
   ...
   case :
     ; break;
   default:  // optional
     ; break;
 }
```

Наредби за повторување (итеративни наредби)

Паскал код:

```
   { while }
 while  do begin
   ;
   ...
 end;
  { for }
 for i := n1 to n2 do begin
```

```
  ...       { зголемуваме за еден }
end;
for i := n2 downto n1 do begin
  ...       {намалуваме за еден }
for ch := ltr1 to ltr2 do
  ...
  { repeat - until }
repeat
  ;
  ...
until ;
```

Кај наредбата repeat, наредбите во циклусот се извршуаат барем еднаш бидејќи излезот е на крајот од циклусот.

C++ код:

```
 // while
while (  ) {
  ;
  ...
}
 // for
for ( ;; )
{
  ...
}
```

Пример за for:

```
for (i = 0;   i < n;   i++)
  ...
 // do - while
do {
  ;
  ...
} while ( );
```

Наредби за прекин (*break* и *continue*)

Паскал код:

Во паскал не постојат наредба за прекин или пак за продолжување.

C++ код:

```
for (i = 0;   i < n;   i++) {
   if (a[i] < 0) break;
     // Излез од for-наредбата
   if (a[i] == 0) continue;
     // Продолжи со следното i
   // Ако се наоѓаме тука тогаш a[i] е >0
   b = a[i];
     ...
}
```

Процедури и Функции

Паскал код:

```
procedure Primer
     (n : integer; ch : char);
 ...
   begin
     ...
   end;
 function PrimerFunkcija(m, n : integer) : real;
 ...
   begin
     ...
     PrimerFunkcija:= ;
   end;
```

Процедурите и функциите примаат аргументи. Процедурите не враќаат вредност, наспроти функциите кои
 враќаат вредност на дефиниран тип на податок т.е вредност на аргументот.

C++ код:

```
void Primer (int n, char ch)
 {
     ...
 }
 double Primer1(int m, int n)
 {
     ...
     return ;
 }
```

Во C++ не постојат процедури, се е функција. функциите примаат аргументи и враќаат вредности. Функциите кои не враќаат експлицитна вредност се дефинираат како void функции.

Функциите кои не се void враќаат вредност со помош на наредбата return, можно е повеќекратно враќање на вредности со тоа што ставаме услов за разгранување if.

Место на процедурите и функциите во програмот

Паскал код:

```
program ...
 ...
 procedure Primer (...);
 ...
   begin
     ...
   end;
 ...
 begin { main }
   ...
   Primer(...);
   ...
 end.
```

Во Паскал вообичаено е дефинирањето на функциите и процедурите да биде пред тие да бидат повикани. Дозволено е вгнездување на функциите и процедурите.

C++ код:

```
// Function prototype:
 double MyFunc(int arg1, int arg2);
 int main()
 {
   double x;
   ...
   x = MyFunc(1999, 3);
   ...
 }
 ...
 // Function definition:
 double MyFunc(int arg1, int arg2)
 {
   ...
 }
```

Функцијата мора да биде декларирана пред првото нејзино повикување, но можеме телото или кодот кој таа ќе го извршува да биде напишан по нејзиното повикување. Во C++ вгнездените функции не се дозволени.

Предавање на параметри (аргументи) според адреса

Паскал код:

```
procedure Primer (var x, y : integer);

procedure Primer1 (a, b, c : real; var x1, x2 : real);
```

C++ код:

```
void Primer (int &x, int &y);
 void Primer1 (double a,double b, double c, double &x1, double &x2);
```

Во паскал за да го направиме ова потребно ни е користење на резервираниот збор var, додека тоа кај C++ се прави со симболот “&“.

Покажувачи

Еден прост пример за креирање, бришење на покажувач како и негово печатење т.е на вредноста на која што покажува.

Паскал код:

```
type
   RealArray = array [1..100] of real;
 var
   i : integer;
   pi1, pi2 : ^integer; { pointers to integer }
   pa : ^RealArray;     { pointer to an array of reals }
 begin
   i := 99;
   new (pi1);  { allocates one integer }
   if pi1 = nil then
     writeln ('Greshka vo memoriska alokacija');
   pi1^ := i;
   pi2 := pi1;
   writeln (pi2^);   { output: 99 }
   dispose (pi1);
   new (pa);  { allocate an array of RealArray type }
   pa^[1] := 1.23;
   pa^[2] := pa^[1];
   writeln (pa^[2]); { output: 1.23 }
   dispose (pa);
 end.
```

C++ код:

```
int main()
 {
   int i, *pi1, *pi2; // pi1, pi2 are pointers to int.
   double *pa;        // pointer to a double
   i = 99;
   pi1 = new int;
   if (!pi1)      // или со други зборови if (pi == 0)
     cout << " Greshka vo memoriska alokacija " << endl;
   *pi1 = i;
   pi2 = pi1;
   cout << *pi2 << endl;  // Излез: 99
   delete pi1;
   pa = new double[100]; // allocate an array of 100 doubles
   pa[0] = 1.23;
   pa[1] = pa[0];
   cout << pa[1] << endl; // Излез: 1.23
   delete [] pa;
   return 0;
 }
```

Во C++ со користење на операторот & покажувачот можеме да го наместиме да покажува на променлива која е од истиот тип како и покажувачот.

Пример:

```
int a, *pa = &a;  // Покажувачот pa е поставен на адресата на a.
```

Покажувачи и низи

Паскал код:

Во паскал нема директна врска помеѓу покажувачите и низите. Најчесто покажувачите се користат со поврзани листи, дрва и други поврзани структури.

C++ код:

Во C++ постои силна врска помеѓу покажувачите и низите. Пример:

```
int a[100];

a[0] е исто со *a.

a[i] е исто што и *(a+i).
```

Во употребата со поврзани листи и други поврзани структури операторот “new” подржува алокација на низи со променлива должина.

Пример:

```
int n;
 cin >> n; // Enter n
 // Allocate an array of n integers:
 int *a = new int[n];
 a[0] = ...;
```

Адресни променливи (Reference Variables)

Паскал код:

Во паскал не постои такво нешто.

C++ код:

Во суштина тие се скоро исти со покажувачите меѓутоа користат различна синтакса.

Пример:

```
int main()
 {
   int i = 1, &ri = i;
   // ri becomes an alias for i
   i = 99;
   cout << ri << endl;
   // Излез: 99
   ...
 }
```

Адресните променливи се користат најчесто за предавање на аргументи на функции по адреса и за да вратат вредности од тие функции или на некој “пренаполнети оператори“ (overloaded operators).

Класи

Во C++ класите ги комбинираат елементите и поврзаните функции во еден ентитет. Класите се многу згодни за имплементација и главниот чекор до ОО-програмирање. Кај нив ги среќаваме термините како што се “private“, “public“,“constructor“,“destructor“, “полиморфизам” и слично.