

PPK

# Pamięć

- Kod
- Dane (zmienne globalne)
- Stos (zmienne lokalne procedur / funkcji, adresy powrotu)
- Sterta (zmienne dynamiczne)

# Zmienne dynamiczne

- Zmienne tworzone w trakcie działania programu
- Kontrolowane przez programistę
  - Tworzenie
  - Zwalnianie !!!

# Zmienna wskaźnikowa (wskaźnik)

- Zmienna przechowująca adres pamięci komputera
- Umożliwia odwoływanie się do obszaru wskazanego pamiętanym adresem
  - Pobranie wartości
  - Zapis wartości

- type
- `typ_wskaznikowy = ^typ_wskazywany;`
- Typ\_wskazywany -> określony typ

type

tpi = ^integer;

var

pi : tpi;

- var
- pi : ^integer;

rozmiar wskaźnika jest niezależny od zmiennej  
na którą wskazuje

Kompilator 64-bitowy: 8 bajtów

❑ kompilator 32-bitowy: 4 bajty adres

❑



Operacja na wskaźniku

Operacje na elemencie wskazywanym  
zależne od typu na który wskazuje wskaźnik

- Inicjalizacja wskaźnika
  - NIL
  - Obszar pamięci
    - Istniejąca zmienna
    - Zmienna powołana dynamicznie
- Jaka wartość na początku ?

- Alokacja obszaru pamięci
- var
- pi : ^integer;
- begin
- new(pi); //alokacja obszaru
- ...
- dispose(pi); //zwolnienie obszaru
- end.

- Alokacja obszaru pamięci
- var
- pi : ^integer;
- begin
- new(pi); //alokacja obszaru  
pi^ := 30;  
pi^ := pi^+20;  
Write(pi^);
- dispose(pi); //zwolnienie obszaru
- end.

- Alokacja obszaru pamięci
- var
- pi : ^integer;
- Begin
- Pi^:=10; //??
- new(pi); //alokacja obszaru
- ...
- dispose(pi); //zwolnienie obszaru
- Pi^=30; //??
- end.

# NIL

- Specjalna wartość
- Można przypisać do zmiennej wskaźnikowej dowolnego typu
- Określa nieistniejący adres
- Wykorzystywane w porównaniach i testach
  - Informacja iż zmienna nie wskazuje na żaden blok pamięci

- var
- pi : ^integer;
- begin
- pi:=nil;
- ...
- if pr=nil then
- new(pi);
- ...
- dispose(pi);
- end.

# Zwalnianie pamięci

- Należy zwalniać pamięć zaalokowaną dynamicznie
  - Możliwość wyczerpania pamięci
  - Zajmowanie zasobów które mogą być dostępne dla innych aplikacji
- Nie można zwalniać już
  - zwolnionego obszaru pamięci
  - Nie zarezerwowanego obszaru pamięci
  - Zmiennych powołanych przez kompilator
- Pamięć zwolniona może zostać ponownie zaalokowana
- Gdy program się kończy system automatycznie zwalnia pamięć



- var
- pi : ^integer;
- begin
- dispose(pi); //zwolnienie obszaru
- end.

- var
- pi : ^integer;
- begin
- new(pi); //alokacja obszaru
- ...
- dispose(pi); //zwolnienie obszaru
- Dispose(pi); //błąd
- end.

# Błędy

- var
- pi : ^integer;
- begin
- new(pi); //alokacja obszaru
- Pi^=10;
- new(pi); //obszar pamięci nie został zwolniony, pod pi  
//znajduje się nowo zaalokowany obszar  
//nastąpiła utrata fragmentu dostępnej pamięci
- pi^=pi^\*2;
- Dispose(pi);
- end.

- `mark(wskaźnik)` – zapamiętuje bieżącą "wysokość" sterty w zmiennej wskaźnik.
- `release(wskaźnik)` – zwalnia cały obszar sterty leżącego powyżej wskaźnika
- Wykorzystywane w programowaniu niskiego poziomu do zwalniania pamięci przydzielonej na stercie

# Alokacja tablic

```
var
tab: array [1..1000] of ^integer;
begin
  new(tab[1]);

  for i:=2 to 20 do
    new(tab[i]);

  dispose(tab[1]);

  for i:=2 to 20 do
    dispose(tab[i]);
end.
```

- Var
- W\_sterty:^ pointer;
- tab: array [1..1000] of ^integer;
- begin
- new(tab[1]);
- mark(W\_Sterty);
- for i:=2 to 200 do
  - new(tab[i]);
- release(w\_sterty);
- dispose(tab[1]);
- end.

- Var
- W\_sterty:^ pointer;
- tab: array [1..1000] of ^integer;
- begin
- mark(W\_Sterty);
- new(tab[1]);
- for i:=2 to 200 do
  - new(tab[i]);
- release(w\_sterty);
- end.

type

TabInteger = array[1..200] of ^Integer;

var

tab : TabInteger ;

begin

new(tab[10]);

tab[10]^ := 12; ...

end.



- Pointer - typ wskaźnikowy
  - nie związany z konkretnym typem bazowym
  - Zgodny, pod względem przypisania, z innymi typami wskaźnikowymi

# Tablice

type

TabInteger = array[1..200] of Integer;

PTabInteger = ^TabInteger;

var

tab : array[1..200] of PTabInteger;

begin

new(tab[10]);

tab[10]^[30] := 12;

...

end.

type

TabInteger = array[1..200] of Integer;

PTabInteger = ^TabInteger;

var

tab : array[1..200] of PTabInteger;

begin

new(tab[10]);

tab[1]^ [30] := 12; //??

...

end.

# Dostępność pamięci

- Rozmiar tworzonej zmiennej dynamicznej nie może być większy od dostępnego ciągłego obszaru pamięci na stercie
- Sterta ulega defragmentacji
- Brak automatycznego porządkowania sterty
- MemAvail – sumaryczny rozmiar wolnej pamięci na stercie
- MaxAvail – rozmiar największego wolnego bloku na stercie

# Przypisanie adresu istniejącej zmiennej

```
var
```

```
pi : ^integer;
```

```
x : integer;
```

```
begin
```

```
x:=3;
```

```
pi:= @x;
```

```
writeln(pi^:0:1);
```

```
end.
```

# Błędy



```
var  
pi : ^integer;  
x  : integer;  
begin  
x:=3;  
pl:= @x;  
writeln(pl^:0:1);  
Dispose(pl);  
end.
```

- Operator @ zwraca Pointer
- Można odwoływać się do tego samego obszaru pamięci poprzez zmienne wskaźnikowe różnego typu

- var
- pi : ^integer;
- tab: array [0..7] of byte;
- r:integer;
- begin
- Pi:= @tab;
- pi^:=1;
- for i:=0 to 7 do
  - writeln(tab[i]);
- end.



# Tablice dynamiczne

- GetMem
  - Alokuje pamięć na ciąg bajtów o zadanej długości.
- 
- FreeMem
  - zwalnia pamięć przydzieloną za pomocą procedury GetMem
-  Wykorzystują wskaźnik pointer

- type
- ttab= array [0..7] of byte;
- var
- pl : ^integer;
- tab: ^ ttab;
- p : pointer;
- begin
- GetMem(p,8);
- pl:=p;
- pl^:=4;
- tab:=p;
- tab^[0]:=5;
- writeln(pr^);
- FreeMem(p,8);
- end.

- GetMem
  - Zwalniać FreeMem
  - Nie można zwolnić dispose
- New
  - Zwalniać dispose
  - Nie można zwolnić FreeMem

# Absolute

- Miejsce pamięci w którym należy umieścić wartości zmiennej
- var
- a: integer;
- b: integer absolute a;
- begin
- a:=3;
- b:=4;
- writeln(a, ' ', b);
- end;