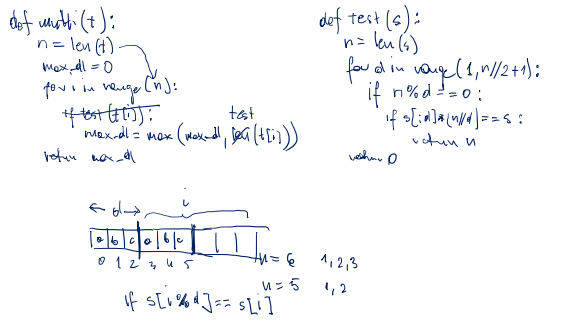
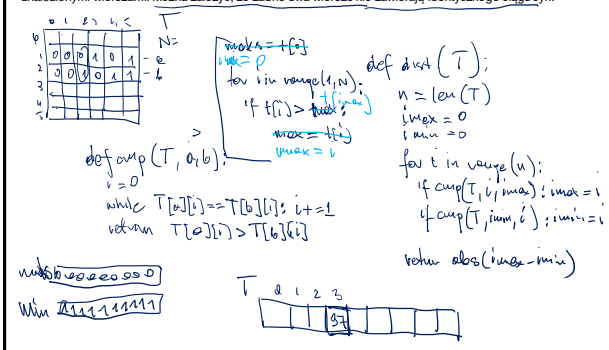
Napis nazywamy wielokrotnym, jeżeli powstał przez n-krotne (n >1) powtórzenie innego napisu o długości co najmniej 1. Przykłady napisów wielokrotnych: ABCABCABC, AAAA, ABAABA. Dana jest tablica T[N] zawierająca napisy. Proszę napisać funkcję `multi(T)`, która zwraca długość najdłuższego napisu wielokrotnego występującego w tablicy T lub wartość 0, jeżeli takiego napisu nie ma w tablicy.



1

Dana jest tablica T[N][N] wypełniona wartościami 0,1. Każdy wiersz tablicy traktujemy jako liczbę zapisaną w systemie dwójkowym o długości N bitów. Stała N jest rzędu 1000. Proszę zaimplementować funkcję `distance(T)`, która dla takiej tablicy wyznaczy dwa wiersze, dla których różnica zawartych w wierszach liczb jest największa. Do funkcji należy przekazać tablicę, funkcja powinna zwrócić odległość pomiędzy znalezionymi wierszami. Można założyć, że żadne dwa wiersze nie zawierają identycznego ciągu cyfr.



2

Wykład 7

- Zmienna i jej aspekty
- Zmienna wskaźnikowa
- Przydział pamięci dla zmiennych
- Działania na zmiennych wskaźnikowych
- Zastosowanie typu wskaźnikowego
- Przykłady

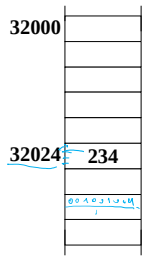
3

Zmienna

Aspekty zmiennej:

- nazwa
- adres (lokacja)
- wartość
- typ
- rozmiar

```
int x;
int main() {
...
  x=234;
  cout << &x;
}
```



4

Instrukcja podstawienia

<zmienna> = <wyrażenie>

z1 = z2 = z3 = wyrażenie (podstawienie wielokrotne)
z1:=: z2 (podstawienie symetryczne)

z ***op*** = wyr ⟺ z = z ***op*** wyr

5

Czas istnienia nazwy

Program w C/C++ —kompilacja→ Program wykonywalny

nazwa zmiennej → adres w pamięci

```
procedure main()
  i:=5;  j:=7;  k:=11
  nazwa:=read()          #j
  m:=variable(nazwa)
  write(m)               #7
  variable(nazwa):=3
  write(j)               #3
end
```

Język Icon

6

Funkcje transformujące

	Pascal	Język C
zmienna ⇒ adres (dostarcza adres zmiennej)	**addr(x)** **@x**	**&x**
adres ⇒ zmienna	**a^**	***a**

```
var
   x:real; absolute adres;
```

Język Turbo Pascal

7

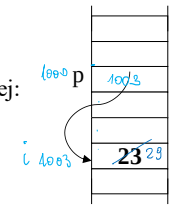
Zmienna wskaźnikowa

Zmienna wskaźnikowa - zmienna, która przechowuje adres innej zmiennej.

Zmienna wskazywana - zmienna, na którą wskazuje zmienna wskaźnikowa.

Deklaracja zmiennej wskaźnikowej:

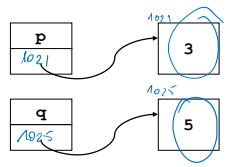
```
int i=23;
int *p;

p = &i;
*p = 29;
```



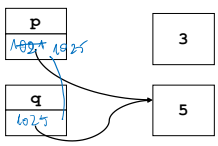
8

Zmienna wskaźnikowa i wskazywana

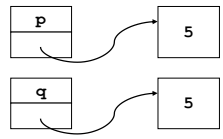
```
{
  int *p;
  int *q;
  ...
  *p = 3;
  *q = 5;
  ...
}
```



p = q



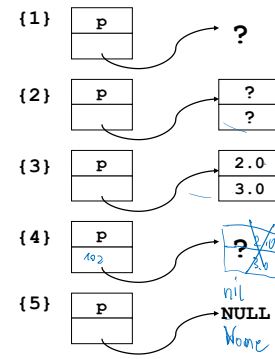
*p = *q



9

Alokacje i zwalnianie pamięci

```
struct punkt {
  double x,y;
};

punkt *p;          {1}


p = new punkt;     {2}
   ...
p->x = 2.0;        {3}
p->y = 3.0;        {3}
   ...
delete p;          {4}
   ...
p = NULL;          {5}
```



10

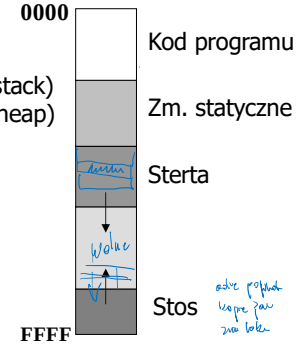
Operacje na zmiennych wskaźnikowych

deklarowanie:	typ *p;
alokacja zmiennej:	p = new typ;
zwalnianie pamięci:	delete p;
przypisanie:	=
operacje logiczne:	== !=
wartość „pusta”:	NULL
wypisanie wartości:	*cout << p;*

11

Przydział pamięci

- statyczny
- dynamiczny ze stosu (stack)
- dynamiczny ze sterty (heap)



12

Zastosowania typu wskaźnikowego

Nieregularne struktury danych:

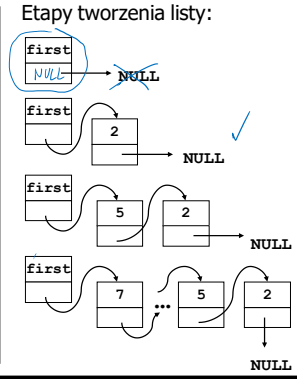
- stos, kolejka, talia, lista
- struktura drzewiasta
- struktura grafowa

13

Tworzenie łańcucha odsyłaczowego (listy)

```
struct node {
  int val;
  node *next;
};

node *first;
node *p;
int s;

first=NULL;
for (int i=1; i<n; i++){
  cin >> s;
  p = new node;
  p->next=first;
  p->val=s;
  first=p;
}
```

Etapy tworzenia listy:



14

Wypisanie wartości listy

Iteracyjnie

```
void wypisz1(node *p)
{
  while (p!=NULL)
  {
    cout << p->val;
    p=p->next;
  }
}
```

Rekurencyjnie

```
void wypisz2(node *p)
{
  if (p!=NULL)
  {
    cout << p->val;
    wypisz2(p->next);
  }
}
```

15

Programowanie obiektowe

- Obiekt – połączenie danych i operacji na nich wykonywanych, unikatowy egzemplarz danych zdefiniowanych w jego klasie
- Metoda – funkcja określona w definicji klasy
- Klasa – szablon, projekt, prototyp obiektu
- Dziedziczenie – przekazywanie charakterystyki klasy do innych klas

16

Programowanie obiektowe

```
class osoba:
  pass
# end

os1 = osoba()
os1.imie = 'Ala'
os1.nazw = 'Nowak'

class user:
  def __init__(self, imie, nazw)
    self.imie = imie
    self.nazw = nazw
  # end
# end
u1 = user('Ola', 'Kowalska')
```

17

Lista

```
class Node:
  def __init__(self):
    self.val = None      # przechowywana liczba całkowita
    self.next = None     # odnośnik do następnego elementu

first = None
for i in range(4):
    s = int(input('>>'))
    p = Node()
    p.val = s
    p.next = first
    first = p

wypisz1(first)
```

18

Wypisanie wartości listy

Iteracyjnie

```
def wypisz1(p):
  while p!=None:
    print(p.val)
    p = p.next
  end
end
```

Rekurencyjnie

```
def wypisz2(p):
  if p!=None:
    print(p.val)
    wypisz2(p.next)
  end
end
```

19