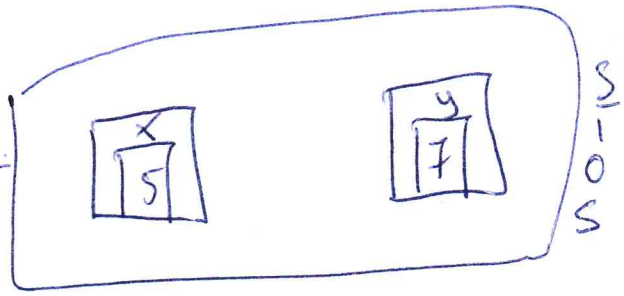
3.2.1

do funkcji trafiają wartości zapisane pod konkretnymi adresami pamięci

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int funkcja(int*a, int*b)
5   {
6       if (*a<*b)
7           return *a;
8       else
9           return *b;
10  }
11
12  int main()
13  {
14      int x=5,y=7;
15      printf("%d",funkcja(&x,&y));
16      return 0;
17  }
18
```



przechowujemy adresy gdzie w pamięci są zmienne x i y

&x - przechowujemy adres

*a - pobieramy wartość pod wskazanym adresem

3.2.2.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int funkcja(int*a, int*b)
5   {
6       if (*a<*b)
7           return a;
8       else
9           return b;
10  }
11
12  int main()
13  {
14      int x=5,y=7;
15      printf("%#010x",funkcja(&x,&y));
16      return 0;
17  }
18
```

argumenty są wskaźnikowe

zwracamy wskaźnik na mniejszą z liczb

format 0x...

przekazujemy adresy

3.2.3. – Zadanie pomocnicze

(potrzeba po co są potrzebne adresy!!)

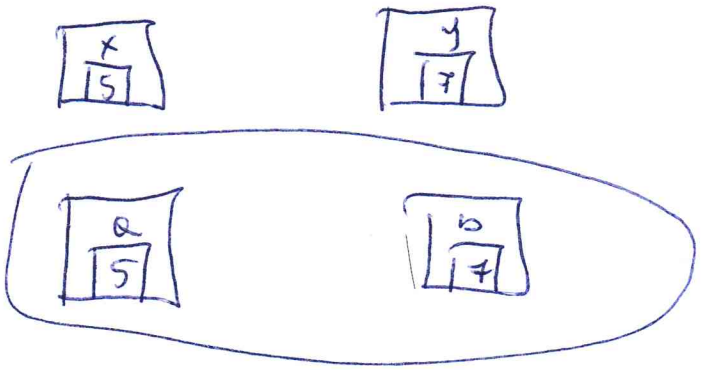
```
1   #include <stdio.h>
2
3   void zamiana(int a, int b)
4   {
5       int temp;
6       temp=b;
7       b=a;
8       a=temp;
9   }
10
11  int main()
12  {
13      int x=5, y=7;
14      printf("%d %d \n",x,y);
15      zamiana(x,y);
16      printf("%d %d",x,y);
17      return 0;
18  }
19
20
```

na wyjściu jest 5 7
5 7

Dlaczego? w tym miejscu kopiowane są wartości.

linijka 13

linijka 15, po linijce 3.



← to kolejna kopia wartości

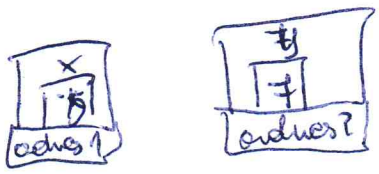
(na nich wykonywane jest wszystkie funkcje)

3.23.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void zamiana(int*a, int*b)
5   {
6       int temp;
7       temp=*b;
8       *b=*a;
9       *a=temp;
10  }
11
12  int main()
13  {
14      int x=5, y=7;
15      printf("%d %d \n",x,y);
16      zamiana(&x,&y);
17      printf("%d %d",x,y);
18      return 0;
19  }
20
```

↑ odczytujemy wartości na wskazanych adresach

⇦ przekazujemy adresy



zamiana

3.2.4.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void funkcja(int *a,int *b)
5   {
6       if(*a>*b)
7       {
8           int temp=*a;
9           *a=*b;
10          *b=temp;
11      }
12  }
13
14  int main()
15  {
16      int x=7,y=8;
17      funkcja(&x, &y);
18      printf("%d %d \n",x,y);
19      int s=7,t=6;
20      funkcja(&s, &t);
21      printf("%d %d \n",s,t);
22      return 0;
23  }
24
```

pobranie wartości pod podanym adresem

operacje na wartościach pod wskazanymi adresami

przekazanie adresów

przekazanie adres.

B.2.5

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4
5   int funkcja(const int *a,const int *b)
6   {
7       return *a+*b;
8   }
9
10
11  int main()
12  {
13      int x=5,y=7;
14      printf("%d",funkcja(&x,&y));
15      return 0;
16  }
17
```

nie można zmodyfikować
gdy będziemy to wyrażali
*a=5;
będzie błąd
kompilacji

3.2.6

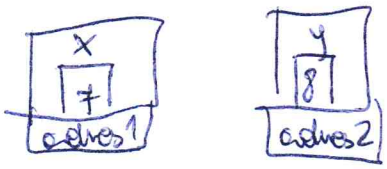
```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void funkcja(int n, int *w)
5   {
6       *w=n;
7   }
8
9   int main()
10  {
11      int x=7,y=8;
12      funkcja(x, &y);
13      printf("%d",y);
14      return 0;
15  }
16
```

↑ wskaźnik

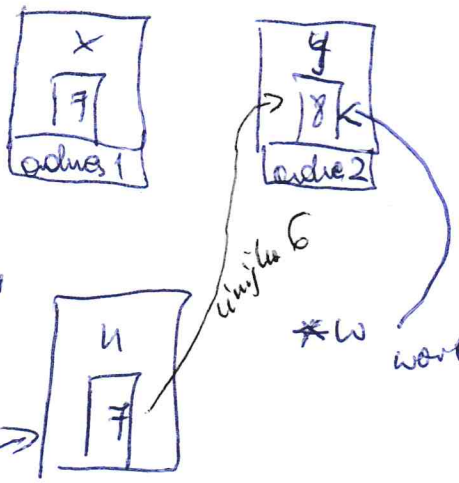
↑ przypisujemy wartość n do zmiennej wskazanej przez w

← dajemy adres

Linijka 11



Linijka 12

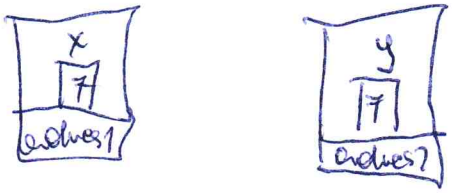


w funkcji

*w wartość pod adresem &y

skopiowano wartość 7 do zmiennej lokalnej n

po wykonaniu funkcji



3.2.9

potrzebna biblioteka

```
1   #include <stdio.h>
2   #include <stdlib.h>
3      *
4   int funkcja()
5   {
6       return malloc(sizeof(int));
7   }
8
9   int main()
10  {
11      printf("%#010x",funkcja());
12      return 0;
13  }
14
```

← tu można ustawić gwiazdkę (choć nic nie zmieni)

malloc - przydzielenie zadanej pamięci

jeśli się powiedzie, to zwracany jest adres

alternatywna opcja -

```
int funkcja()
{
    int a;
    return 0;
}
```

3.2.10

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int* funkcja()
5   {
6       return malloc(sizeof(double));
7   }
8
9   int main()
10  {
11      printf("%#010x",funkcja());
12      return 0;
13  }
14
```

potrzebna biblioteka do malloc

zwracamy to jako wskaźnik

typ zwracanej zmiennej musi być int bo adresy są całkowite

[illegible]

3.2.11.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int* funkcja(int n)
5   {
6       return malloc(n*sizeof(int));
7   }
8
9   int main()
10  {
11      printf("%#010x",funkcja(3));
12      return 0;
13  }
14
```

3.2.12

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int* funkcja(int n)
5   {
6       return malloc(n*sizeof(double));
7   }
8
9   int main()
10  {
11      printf("%#010x",funkcja(4));
12      return 0;
13  }
14
```

dynamicznie może być pamięta

typ zwracany to int do adresy są całkowite

3.2.13

```
1   #include<stdlib.h>
2   #include<stdio.h>
3   double fun(int x)
4   {
5       return x+1;
6   }
7
8   double funkcja(double (*fun)(int arg),int a)
9   {
10      return fun(a);
11  }
12  main()
13  {
14      int a=1,arg,x;
15      printf("%lf",funkcja((&fun),a));
16      return 0;
17  }
18
```

wskaźnik na funkcję

&fun i fun to jest to samo w C!

3.2.14

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int fun1(int n)
5   {
6       return n;
7   }
8
9   int fun2(int n)
10  {
11      return n%5;
12  }
13
14  int fun3(int n)
15  {
16      return (n*n+2*n)/(n+2);
17  }
18
19  int funkcja(int (* f1)(int), int (* f2)(int), unsigned int
    n)
20  {
21      for(int i = 0; i <= n; i++)
22      {
23          if(f1(i) != f2(i))
24              return 0;
25      }
26      return 1;
27  }
28
29  int main()
30  {
31      printf("%d \n", funkcja(&fun1,&fun2,5));
32      printf("%d \n", funkcja(&fun1,fun3,7));
33      printf("%d \n", funkcja(fun2,fun3,9));
34      return 0;
35  }
36
```

wskaźniki na funkcje

&fun1 : fun1

znaczą w C to samo!

3.2.15

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void funkcja(int const *a , int* b)
5   {
6       *b=*a;
7   }
8
9   int main()
10  {
11      int a=2, b=3;
12      printf("%d %d \n",a,b);
13      funkcja(&a,&b);
14      printf("%d %d \n",a,b);
15      return 0;
16  }
17
```

nie można zmodyfikować zmiennej wskazywanej przez podany wskaźnik

wskaźnik na stałą wartość

3.2.16

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void funkcja(int const*a , int*const b)
5   {
6       *b=*a;
7   }
8
9   int main()
10  {
11      int a=2, b=3;
12      printf("%d %d \n",a,b);
13      funkcja(&a,&b);
14      printf("%d %d \n",a,b);
15      return 0;
16  }
17
```

stały wskaźnik:
wskaźnika nie można
przestawić na
inny adres

PRZYKŁAD: RÓŻNICA MIĘDZY WSKAŹNIKIEM NA STAŁĄ A STAŁYM WSKAŹNIKIEM

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main()
5   {
6       int i=0;
7       int const *a=&i; //wskaznik na stala wartosc
8       int * const b=&i; //stały wskaznik
9       *a = 1;  /* kompilator zaprotestuje */
10      *b = 2;  /* ok */
11      a = b;   /* ok */
12      b = a;  /* kompilator zaprotestuje */
13      return 0;
14  }
15
```