Programowanie

Dariusz Wardecki, wyk. V

Powtórzenie

Co zwraca funkcja i jakie warości zostaną wypisane?

```
#include<iostream>
using namespace std;

void add(int a, int b){
  a += 2;
  b += 2;
}

int main(){
  int a = 10;
  int b = 20;
  add(a, b);
  cout << a << "\t" << b << endl;
  return 0;
}
```

Powtórzenie

Co zwróci funkcja?

```
#include<iostream>
using namespace std;

double suma(int n)
{
  double s = 0;
  double a_n = 1;
  for(int i = 1; i < n; i++)
  {
    a_n /= 2*i; // a_n = 1/n!!
    s += (1/2)*a_n;
  }
  return s;
}
```

Adresowanie pamięci

adres	komórka pamięci	wielkość
`0x7fff5b7359d0`		1 bajt
`0x7fff5b7359d1`		1 bajt
`0x7fff5b7359d2`		1 bajt
`0x7fff5b7359d3`		1 bajt
`0x7fff5b7359d4`		1 bajt
`0x7fff5b7359d5`		1 bajt
`0x7fff5b7359d6`		1 bajt
`0x7fff5b7359d7`		1 bajt
...	...	...

Adresowanie pamięci

adres	komórka pamięci	wielkość
0x7fff5b7359d0		1 bajt
0x7fff5b7359d1		1 bajt
0x7fff5b7359d2		1 bajt
0x7fff5b7359d3		1 bajt
0x7fff5b7359d4		1 bajt
0x7fff5b7359d5		1 bajt
0x7fff5b7359d6		1 bajt
0x7fff5b7359d7		1 bajt
...	...	...

char

Adresowanie pamięci

adres	komórka pamięci	wielkość
0x7fff5b7359d0		1 bajt
0x7fff5b7359d1		1 bajt
0x7fff5b7359d2		1 bajt
0x7fff5b7359d3		1 bajt
0x7fff5b7359d4		1 bajt
0x7fff5b7359d5		1 bajt
0x7fff5b7359d6		1 bajt
0x7fff5b7359d7		1 bajt
...	...	...

char

Adresowanie pamięci

adres	komórka pamięci	wielkość
0x7fff5b7359d0		1 bajt
0x7fff5b7359d1		1 bajt
0x7fff5b7359d2		1 bajt
0x7fff5b7359d3		1 bajt
0x7fff5b7359d4		1 bajt
0x7fff5b7359d5		1 bajt
0x7fff5b7359d6		1 bajt
0x7fff5b7359d7		1 bajt
...	...	...

char

int, float

Adresowanie pamięci

adres	komórka pamięci	wielkość
0x7fff5b7359d0		1 bajt
0x7fff5b7359d1		1 bajt
0x7fff5b7359d2		1 bajt
0x7fff5b7359d3		1 bajt
0x7fff5b7359d4		1 bajt
0x7fff5b7359d5		1 bajt
0x7fff5b7359d6		1 bajt
0x7fff5b7359d7		1 bajt
...	...	...

char

int, float

Adresowanie pamięci

adres	komórka pamięci	wielkość
0x7fff5b7359d0		1 bajt
0x7fff5b7359d1		1 bajt
0x7fff5b7359d2		1 bajt
0x7fff5b7359d3		1 bajt
0x7fff5b7359d4		1 bajt
0x7fff5b7359d5		1 bajt
0x7fff5b7359d6		1 bajt
0x7fff5b7359d7		1 bajt
...	...	...

char

int, float

double

Adresowanie pamięci

adres	komórka pamięci	wielkość
0x7fff5b7359d0		1 bajt
0x7fff5b7359d1		1 bajt
0x7fff5b7359d2		1 bajt
0x7fff5b7359d3		1 bajt
0x7fff5b7359d4		1 bajt
0x7fff5b7359d5		1 bajt
0x7fff5b7359d6		1 bajt
0x7fff5b7359d7		1 bajt
...	...	...

char

int, float

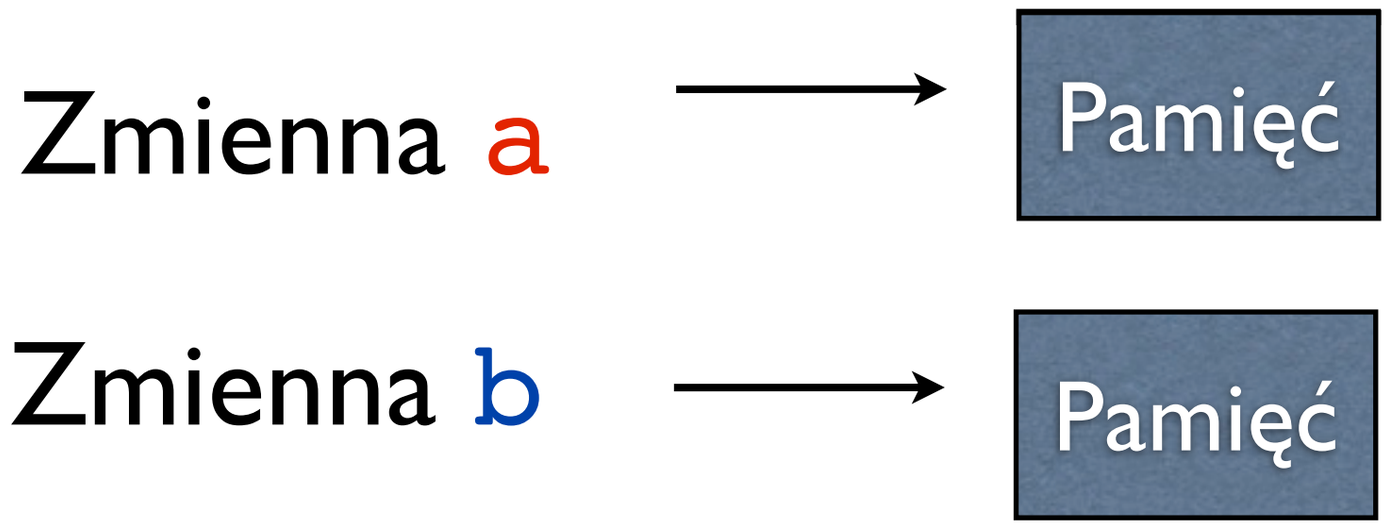
double

Jak odczytać adres?

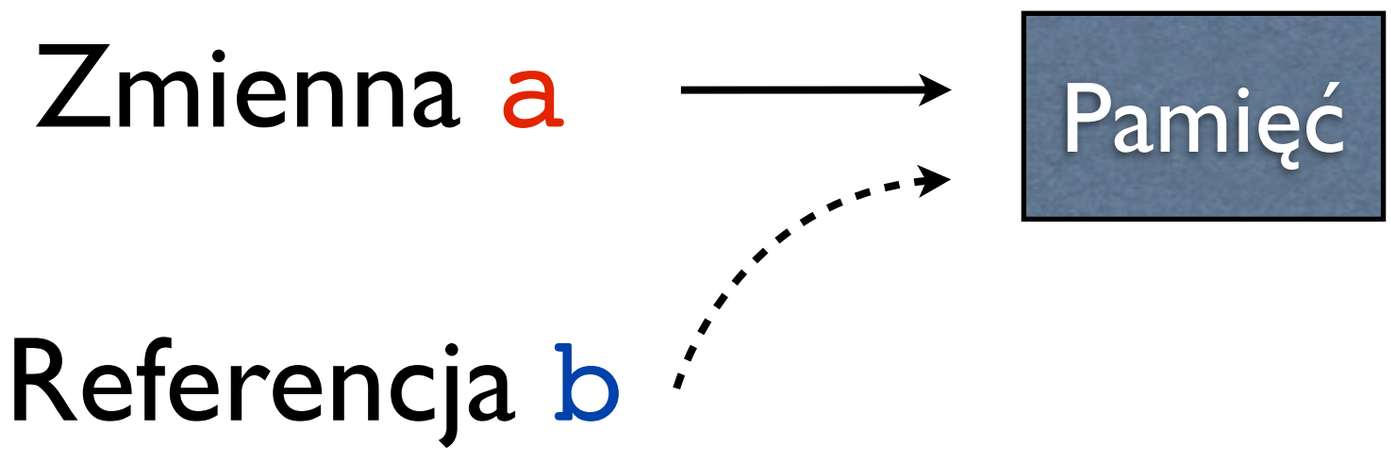
`operator&` – zwraca adres zmiennej

```
#include<iostream>
using namespace std;

int main()
{
  double s = 66;
  cout << "Zmienna s jest pod adresem " << &s
  << " i ma wartosc " << s << endl;
  return 0;
}
```

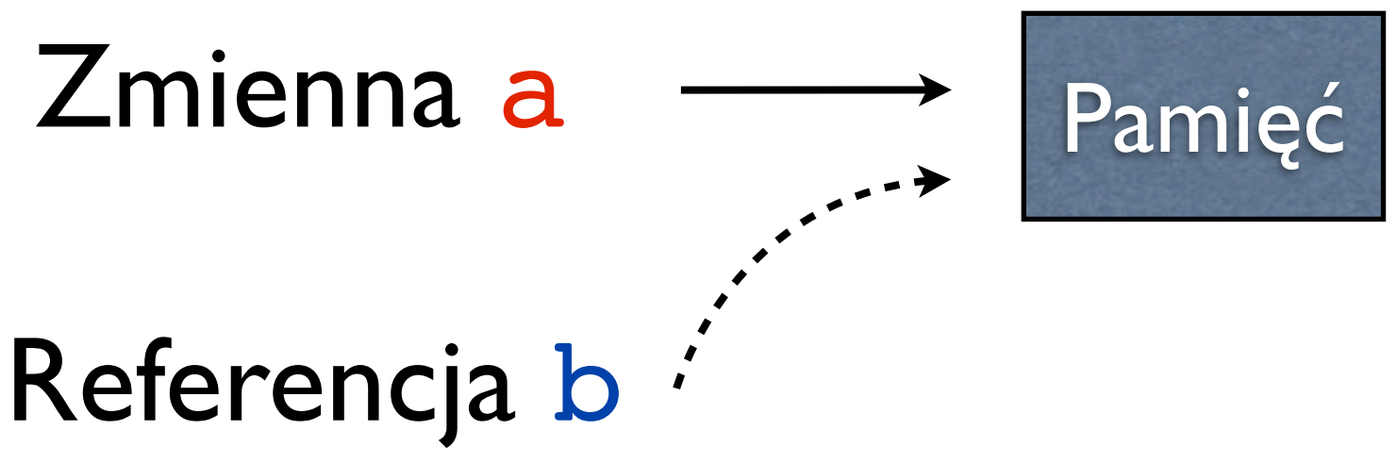
Referencje



Referencje

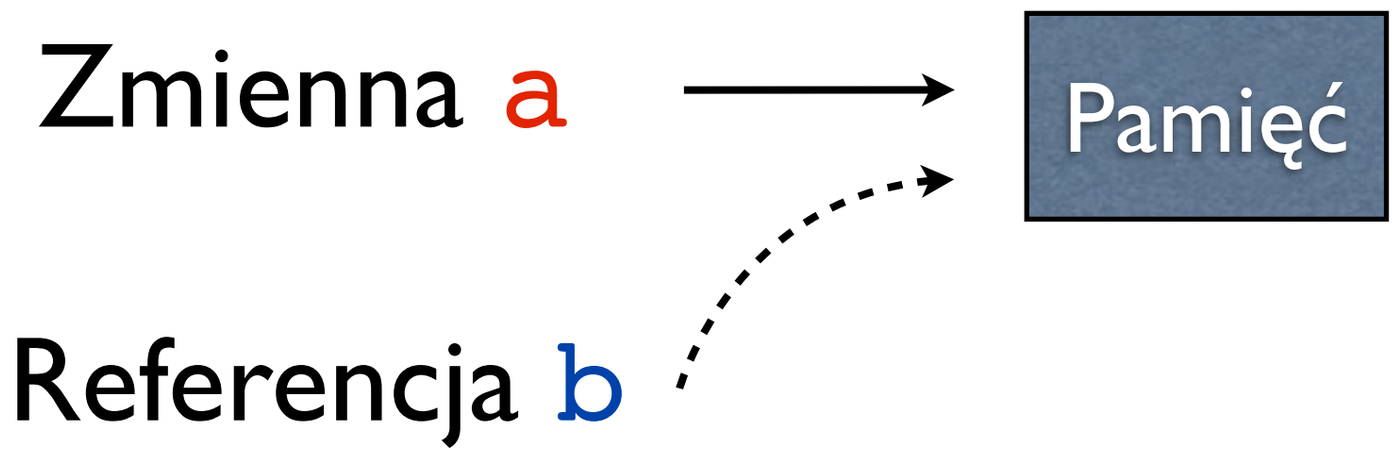


Referencje



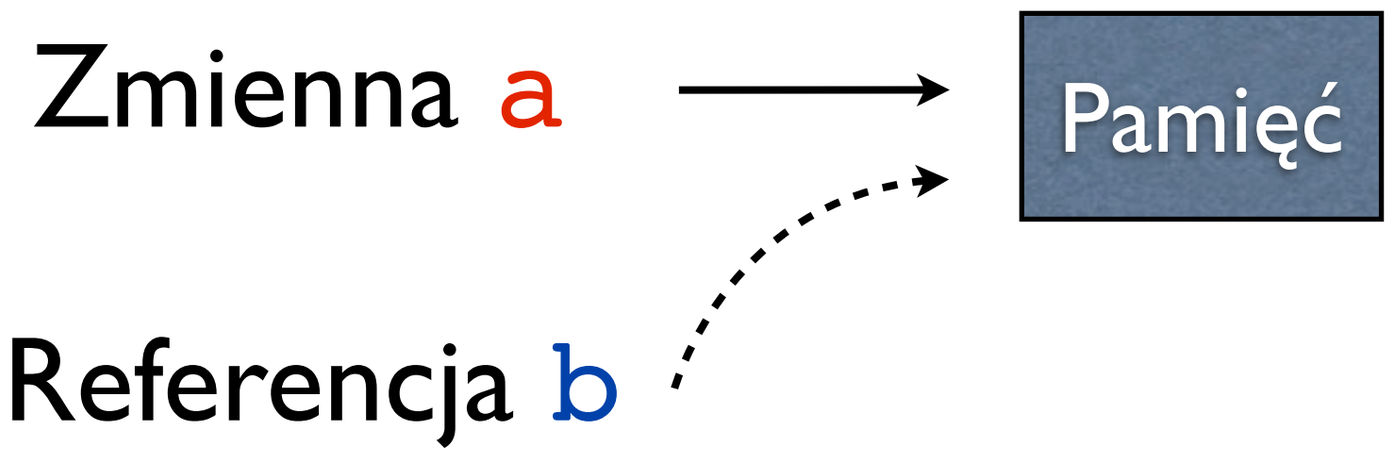

```
double a = 5;
double& b = a;

b = 7; // => a = 7
```

Referencje




```
double a = 5;
double& b = a;          double& b;

b = 7; // => a = 7
```

Referencje




```
double a = 5;
double& b = a;

b = 7; // => a = 7
```

```
double& b;
```

Przekazywanie referencji do funkcji

```
#include<iostream>
using namespace std;

void add(int a, int b){
  a += 2;
  b += 2; // Funkcja pracuje na kopii!!!
}

int main(){
  int a = 10;
  int b = 20;
  add(a, b);
  cout << a << "\t" << b << endl;
  return 0;
}
```

```
10   20
```

Przekazywanie referencji do funkcji

```
#include<iostream>
using namespace std;

void add(int& a, int& b){
  a += 2;
  b += 2; // Funkcja pracuje na oryginale!!!
}

int main(){
  int a = 10;
  int b = 20;
  add(a, b);
  cout << a << "\t" << b << endl;
  return 0;
}
```

```
12   22
```

Wskaźniki

Wskaźnik jest zmienną, która przechwuje adres innej zmiennej.

Wskaźniki

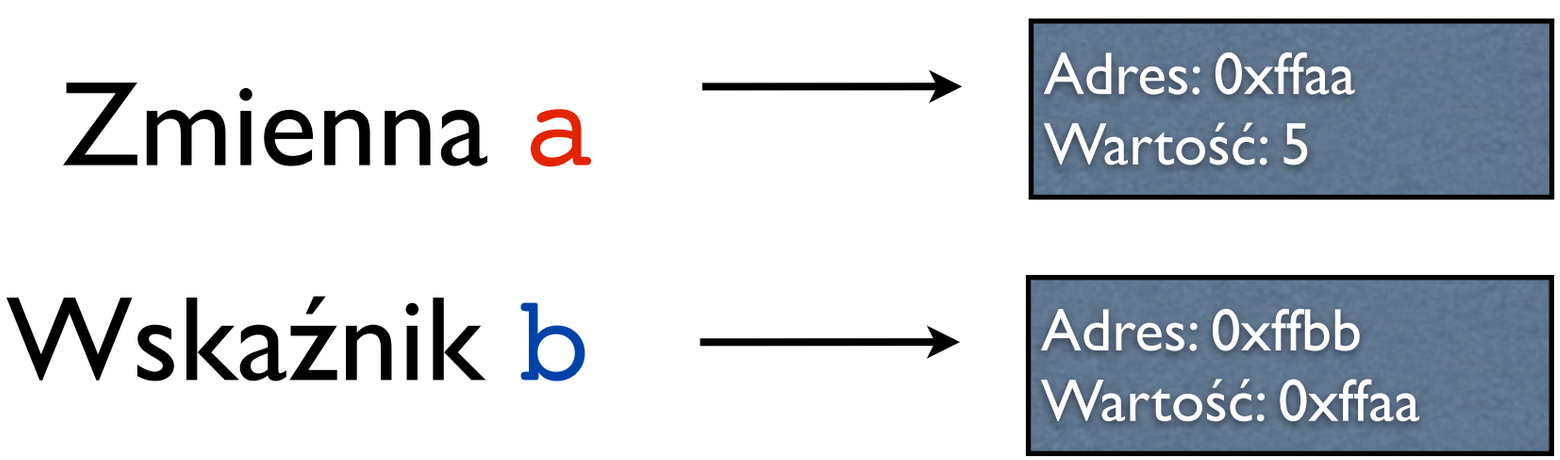
Wskaźnik jest zmienną, która przechwuje adres innej zmiennej.

```
double a = 5;
double* b = &a;

*b = 7; // => a = 7
```

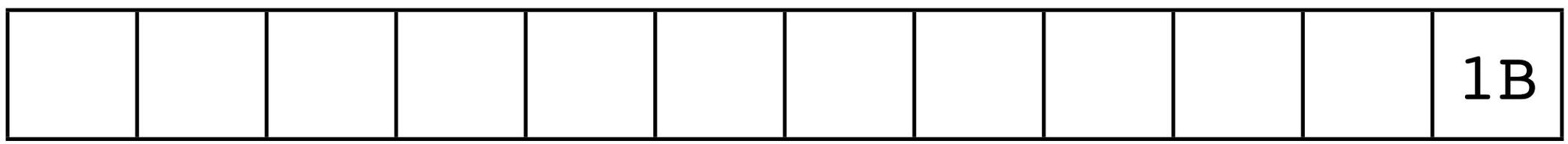
Wskaźniki

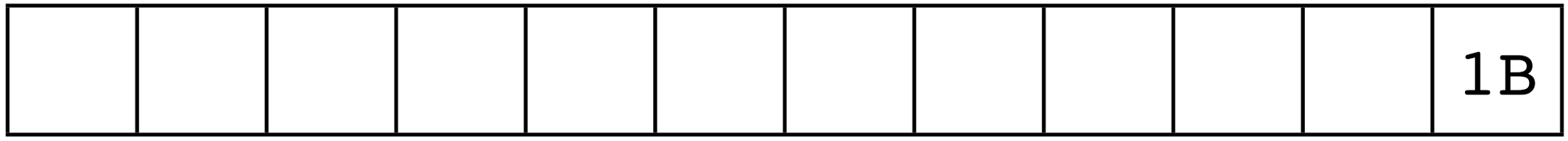
Wskaźnik jest zmienną, która przechwuje adres innej zmiennej.

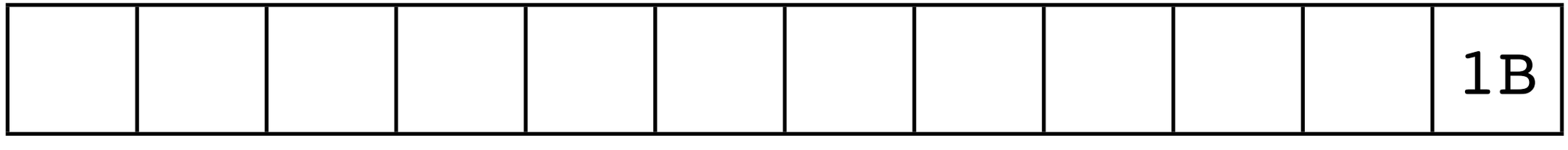
```
double a = 5;
double* b = &a;

*b = 7; // => a = 7
```



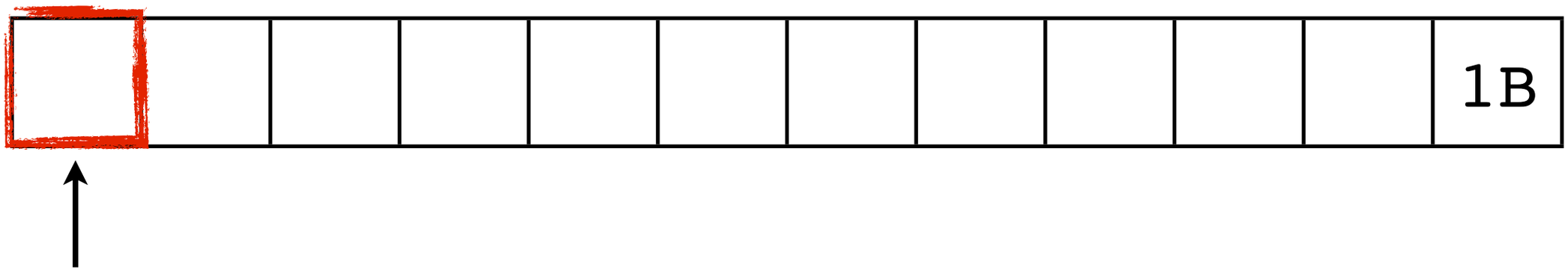
Wskaźniki



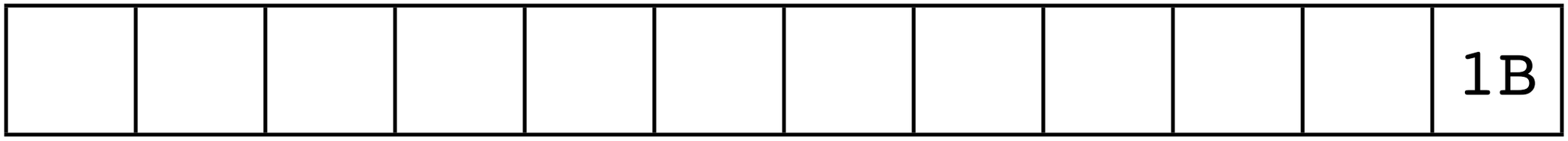


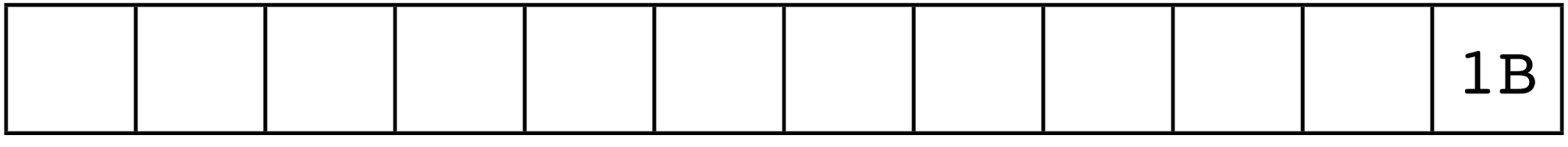


Wskaźniki

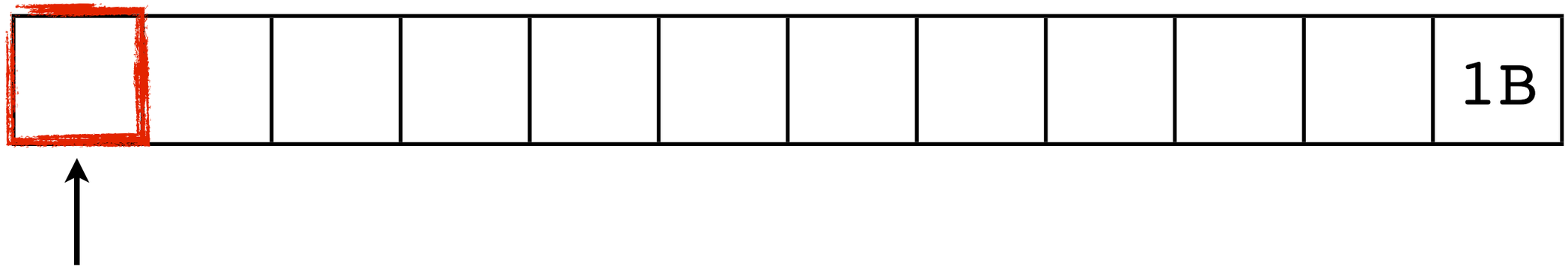



```
char* wsk;
```

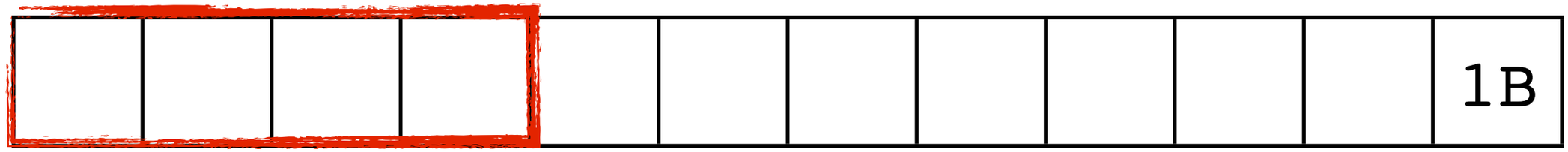




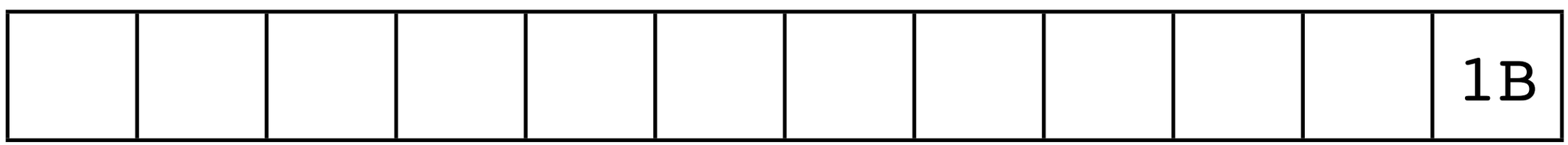
Wskaźniki



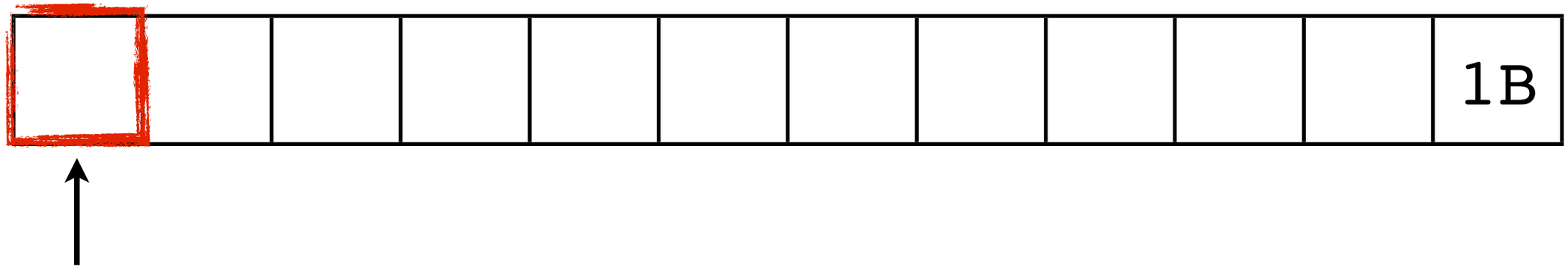

```
char* wsk;
```



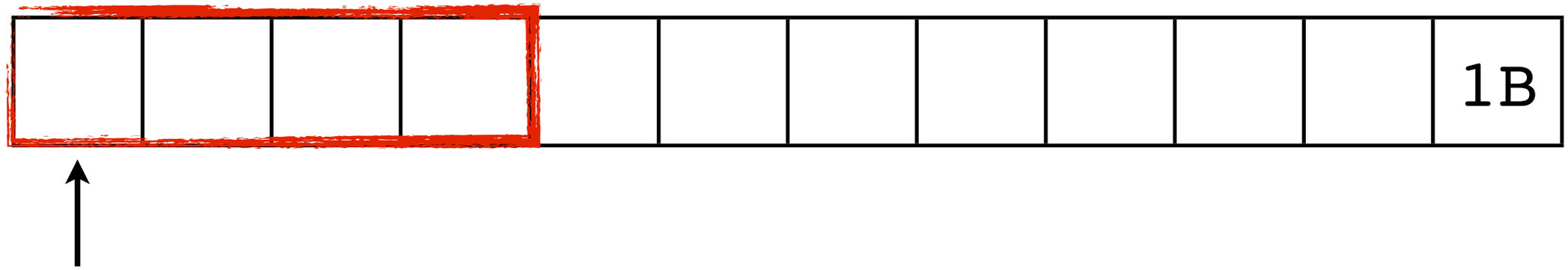

```
int* wsk;
```



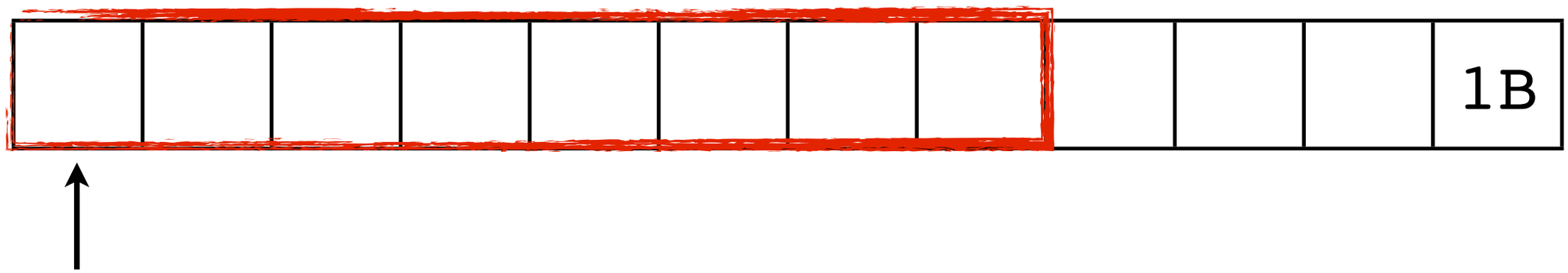
Wskaźniki




```
char* wsk;
```




```
int* wsk;
```




```
double* wsk;
```

Przekazywanie wskaźnika do funkcji

```
#include<iostream>
using namespace std;

void add(int a, int b){
  a += 2;
  b += 2; // Funkcja pracuje na kopii!!!
}

int main(){
  int a = 10;
  int b = 20;
  add(a, b);
  cout << a << "\t" << b << endl;
  return 0;
}
```

```
10   20
```

Przekazywanie wskaźnika do funkcji

```
#include<iostream>
using namespace std;

void add(int* a, int* b){
  a += 2;
  b += 2; // Funkcja pracuje na oryginale!!!
}

int main(){
  int a = 10;
  int b = 20;
  add(&a, &b);
  cout << a << "\t" << b << endl;
  return 0;
}
```

```
12   22
```

Przekazywanie wskaźnika do funkcji

```
#include<iostream>
using namespace std;

void add(int* a, int* b){
  a += 2;
  b += 2; // Funkcja pracuje na oryginale!!!
}

int main(){
  int a = 10;
  int b = 20;
  int* s = &a;
  int* t = &b;
  add(s, t);
  cout << a << "\t" << b << endl;
  return 0;
}

12  22
```

Przekazywanie funkcji do funkcji

```
typ funkcja(typ (*fun)(typ, typ, ...), ...)
```

Przekazywanie funkcji do funkcji

```
typ funkcja(typ (*fun)(typ, typ, ...), ...)
```

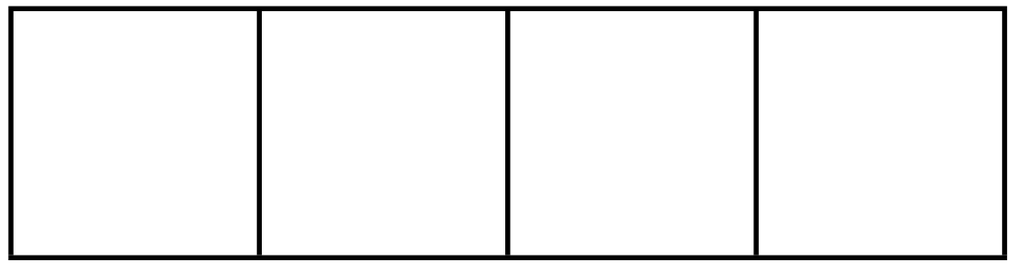
```
#include <cmath>
#include <iostream>
using namespace std;

double trapezoid(double a,double b,int n)
{
  double h=(b-a)/n, s=0;
  for (int i=0; i<n; i++)
  s += h*(sin(a+i*h) + sin(a+(i+1)*h))/2;
  return s;
}

int main ()
{
  cout << trapezoid(0,M_PI,100) << endl;
  return 0;
}
```

Przekazywanie funkcji do funkcji

```
typ funkcja(typ (*fun)(typ, typ, ...), ...)
```

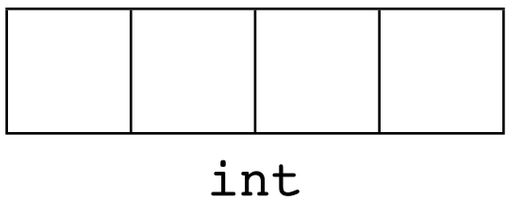
```
#include <cmath>
#include <iostream>
using namespace std;

double trapezoid(double (*f)(double), double a,double b,int n)
{
  double h=(b-a)/n, s=0;
  for (int i=0; i<n; i++)
  s += h*(f(a+i*h) + f(a+(i+1)*h))/2;
  return s;
}

int main ()
{
  cout << trapezoid(sin,0,M_PI,100) << endl;
  return 0;
}
```

Tablice



`int`

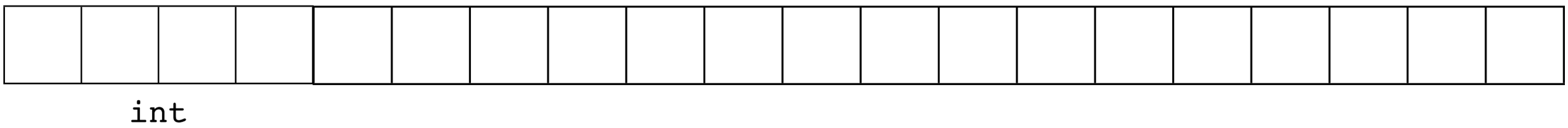
Tablice



Tablice

Tablice

Tablice



element 0	element 1	element 2	element 3	element 4

```
int tab[5] = {1,2,3,4,5};
```

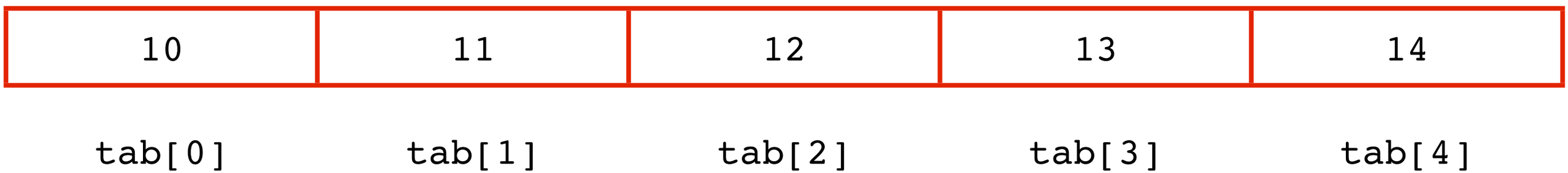
Tablice

Tablice

```
int tab[5];
for(int i=0;i<5;i++)
 tab[i] = i+10;
```

Tablice

```
int tab[5];
for(int i=0;i<5;i++)
 tab[i] = i+10;
```



Tablice wielowymiarowe

Tablice wielowymiarowe

```
int tab[3][5];// tablica 2D
```

Tablice wielowymiarowe

```
int tab[3][5];// tablica 2D
```

tab[0][0]	tab[0][1]	tab[0][2]	tab[0][3]	tab[0][4]
tab[1][0]	tab[1][1]	tab[1][2]	tab[1][3]	tab[1][4]
tab[2][0]	tab[2][1]	tab[2][2]	tab[2][3]	tab[2][4]

Tablice wielowymiarowe

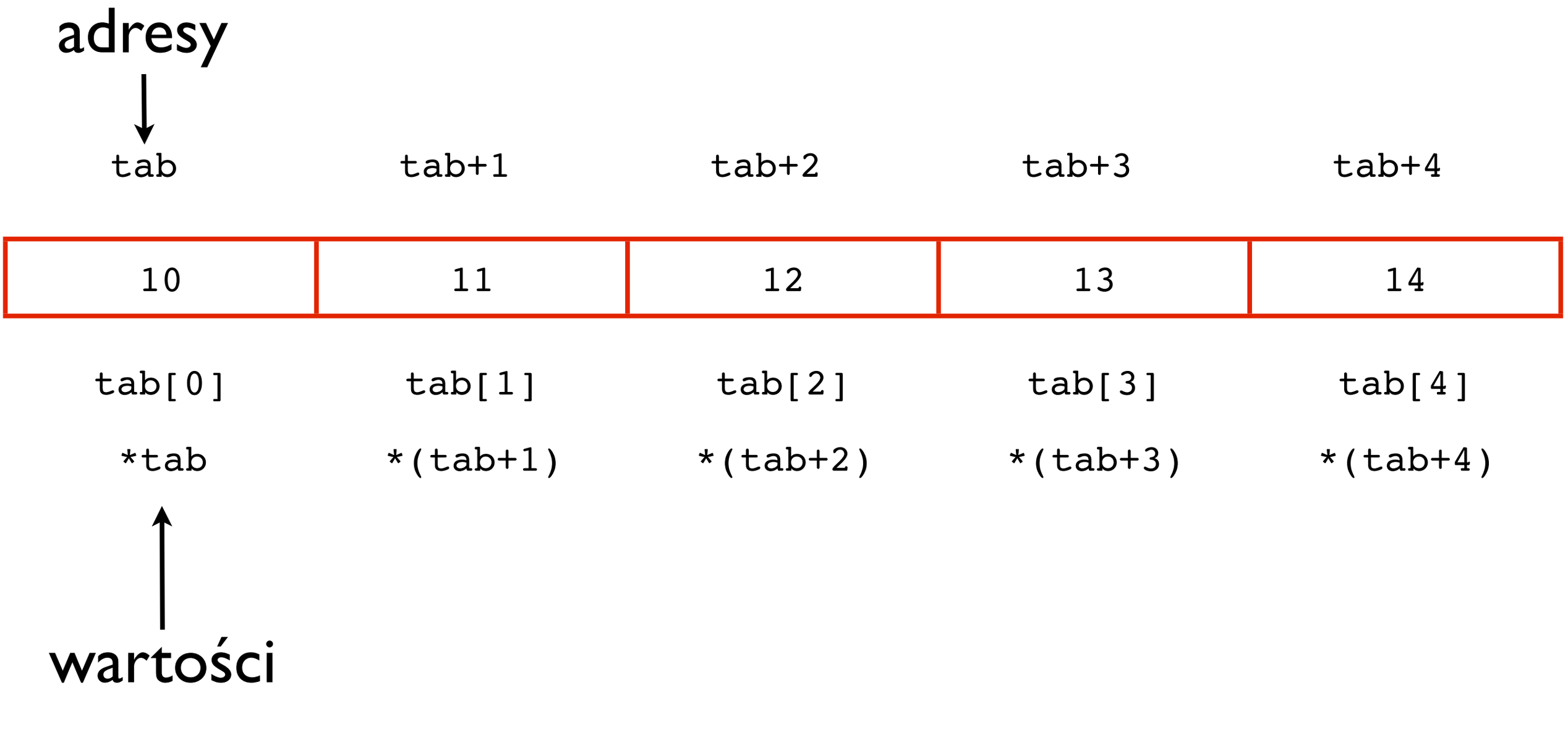
```
int tab[3][5];// tablica 2D
```

tab[0][0]	tab[0][1]	tab[0][2]	tab[0][3]	tab[0][4]
tab[1][0]	tab[1][1]	tab[1][2]	tab[1][3]	tab[1][4]
tab[2][0]	tab[2][1]	tab[2][2]	tab[2][3]	tab[2][4]

```
char wiek[100][365][24][60][60];
// 3GB wymaga pamieci!!!
```

Tablice a wskaźniki

Nazwa tablicy jest jednocześnie wskaźnikiem do jej pierwszego elementu.



Tablice a wskaźniki

Nazwa tablicy jest jednocześnie wskaźnikiem do jej pierwsego elementu.

Tablice a wskaźniki

Nazwa tablicy jest jednocześnie wskaźnikiem do jej pierwsego elementu.

```
#include<iostream>
using namespace std;

int main() {

    int t[5]={8 ,3 ,6 ,5 ,7};
     cout << &t [0] << " " << t <<endl;

     for(int i=0;i<5;i++)
      cout<< i <<" "<< &t[i] <<" "<< t+i <<" "<< t[i]
      <<" "<< *(t+i) <<endl;

     for (int *p=t;p<t+5;p++)
              cout << p-t << " " << p << " " << *p << endl;
     return 0;
}
```

Przekazywanie tablicy do funkcji

```
typ funkcja(typ tab[], typ a, ...)
{ … }

typ funkcja(typ* tab, typ a, ...)
{ … }
```

Przekazywanie tablicy do funkcji

```
void min(int* t, int n)
{
  for(int i = 0; i < n; i++)
    t[i] = t[i] + 10;
    //*(t+i) = *(t+i) + 10;
}

int main()
{
  int n = 5;
  int tab[5] = {32,22,4,65,75}
  for(int i = 0; i < n; i++) cout << tab[i] << endl;
  min(tab, n);
  for(int i = 0; i < n; i++) cout << tab[i] << endl;
  return 0;
}
```