Podstawy informatyki

Katedra Telekomunikacji, **EiT**

dr inż. Jarosław Bułat

kwant@agh.edu.pl

www.agh.edu.pl

Plan prezentacji

- Wskaźniki
  - organizacja pamięci
  - inicjalizacja
  - przypisanie (zgodność typów)
  - tablice
  - arytmetyka
  - rozmiar wskaźnika
  - dynamiczne zarządzanie pamięcią (stos/sterta)
  - tablica wskaźników (argv z funkcji main(...))

www.agh.edu.pl

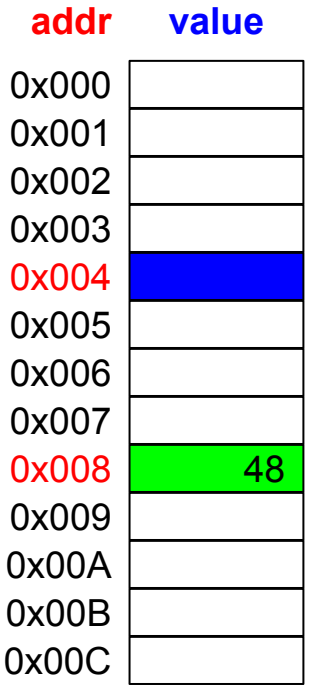
Wskaźniki

można je lubić lub nienawidzić :-)

www.agh.edu.pl

Organizacja pamięci

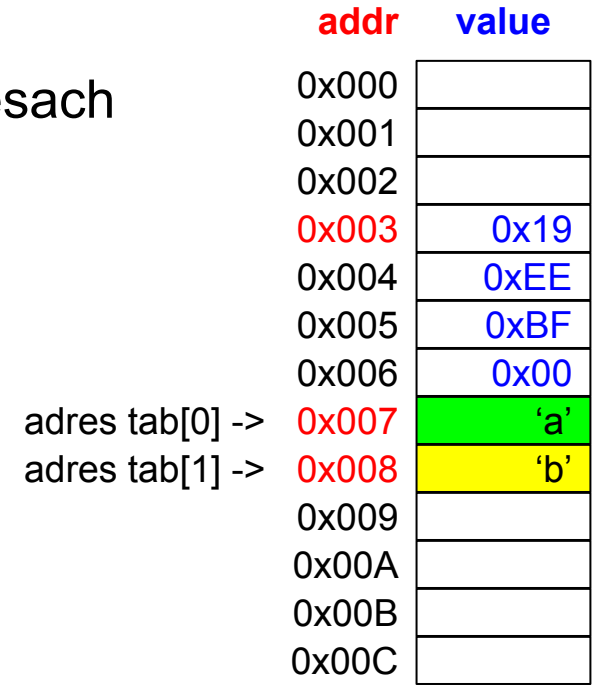
- Pamięć jest ciągła
- Pojedyncza komórka ma wielkość 8b
- Każda komórka ma unikalny adres
- Zawartość komórki jest dostępna przez jej adres
- **Dostęp przez adres to jedyny sposób na poziomie sprzętowym**
- Zmienna char c=48; znajduje się w jednej komórce pamięci (jej wartość)
  - program ma dostęp do niej przez nazwę albo adres
  - Nazwa zmiennej “c” istnieje tylko w programie !!!



www.agh.edu.pl

Organizacja pamięci

- Zmienne o rozmiarze >1B są przechowywane w kolejnych adresach (w ciągłej przestrzeni)
- **int x = 12578329;** // 0xBFEE19
- **char tab[2];**
  - **tab[0] = ‘a’;**
  - **tab[1] = ‘b’;**

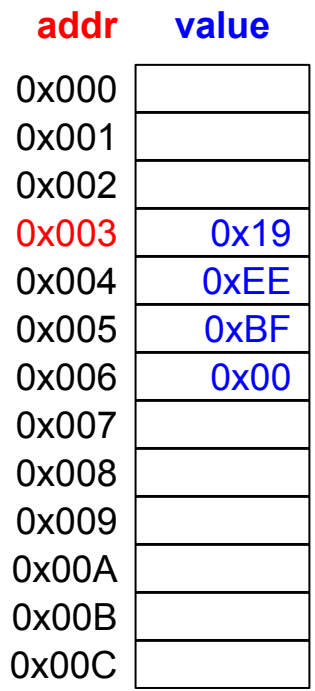


www.agh.edu.pl

Wskaźniki

- Wskaźnik to jest zmienna której wartością jest adres innej zmiennej (zmienna która “wskazuje” inną zmienną)
- Wskaźnik który nie wskazuje innej zmiennej jest *niezainicjalizowany*

int *x;	deklaracja wskaźnika do int (do typu int) **x** jest typu “wskaźnik do int”
x = &y;	operator & to pobranie adresu zmiennej y do x przypisuje się adres a nie jej wartość!!!
int z = *x;	wyłuskanie (ang. dereference) wartości zmiennej



www.agh.edu.pl

inicjalizacja

```
int x = 4;

int *p1 = &x;
int *p2 = p1;

int *p3;
p3 = p2;

p1 = 4;
p1 = x;
x = p2;
```

http://www.cplusplus.com/doc/tutorial/pointers/

www.agh.edu.pl

użycie

```
int x = 4;

int *p1 = &x;
int *p2 = p1;

int *p3;
p3 = p2;

p1 = 4;
p1 = x;
x = p2;
```

```
cout << x << endl;      // 4
cout << p1 << endl;     // 0x7fffbe6781bc
cout << *p1 << endl;    // 4
cout << *p2 << endl;    // 4

*p3 = 7;
cout << *p1 << endl;    // 7
cout << *p2 << endl;    // 7
cout << x << endl;      // 7

int w = *p3;
cout << w << endl;      // 7
```

www.agh.edu.pl

użycie

```
int x = 4;
int *p1 = &x;

int y = *p1;
int *p2 = &y;

cout << x << endl;       // 4
cout << y << endl;       // 4
cout << *p1 << endl;     // 4
cout << *p2 << endl;     // 4

*p2 = 0;    // y = 0;
```

www.agh.edu.pl

użycie

```
int x = 4;
int *p1 = &x;

int y = *p1;
int *p2 = &y;

cout << x << endl;      // 4
cout << y << endl;      // 4
cout << *p1 << endl;    // 4
cout << *p2 << endl;    // 4

*p2 = 0;    // y = 0;
```

```
cout << x << endl;      // 4
cout << y << endl;      // 0
```

www.agh.edu.pl

adres w pamięci

- int x = 12578329; // 0xBFEE19
- int *y = **&x**; // inicjalizacja

- x == 12578329 typ: int
- &x == 0x003 typ: wskaźnik do int
- y == 0x003 typ: wskaźnik do int
- *y == 12578329 typ: int

addr	value
0x000	
0x001	
0x002	
0x003	0x19
0x004	0xEE
0x005	0xBF
0x006	0x00
0x007	
0x008	
0x009	
0x00A	
0x00B	
0x00C	

www.agh.edu.pl

przypisanie - typy

- int x;
- int *y = &x;
- char z;

x = y; błąd: int != int*

y = x; błąd: int* != int

cout << *x; błąd: * w tym kontekście to “mnożenie!!!”

y = &z; błąd: int* != char*

y = 7; błąd*: czy jestem pewien, że “7” to poprawny adres?

www.agh.edu.pl

przypisanie - typy

- int x;
- int *y = &x;
- char z;

x = y; błąd: int != int*

y = x; błąd: int* != int

cout << *x; błąd: w tym kontekście to "mnożenie!!!"

y = &z; błąd: int* != char*

y = 7; błąd*: czy jestem pewien, że "7" to poprawny adres?

Nie mieszaj wartości wskaźnika z wartością zmiennej

www.agh.edu.pl

wskaźniki i tablice

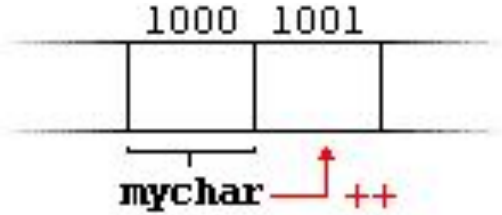
www.agh.edu.pl

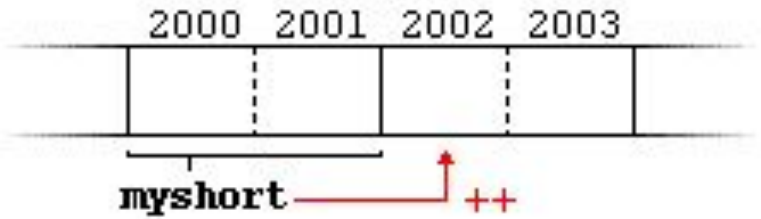
wskaźniki - tablice

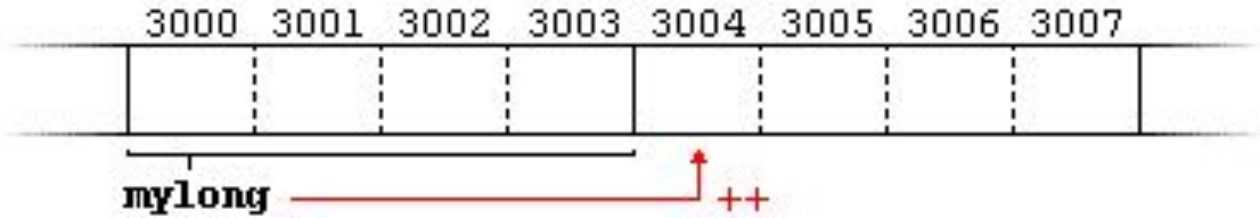
```
int tab[10] = {9,8,7,6,5,4,3,2,1,0};
int *arr;

arr = tab;

cout << tab[2] << endl;    // 7
cout << arr[1] << endl;    // 8
cout << *tab << endl;      // 9
cout << *arr << endl;      // 9
cout << tab << endl;       // 0x7fffbe6781bc

tab = arr;    <-- tab jest “const”, nie mogę zmodyfikować
```

www.agh.edu.pl

wskaźniki - arytmetyka







arytmetyka tylko w kontekście tablic

```
char *mychar;
short *myshort;
long *myshort;
```

http://www.cplusplus.com/doc/tutorial/pointers/

www.agh.edu.pl

wskaźniki - arytmetyka

```
int tab[10] = {9,8,7,6,5,4,3,2,1,0};
int *arr;

cout << arr << endl;            // 0x7fffbe6781bc
cout << arr[1] << endl;         // 8
arr++;
cout << *arr << endl;           // 8
cout << arr[1] << endl;         // 7
cout << (arr+1)[1] << endl;     // 6
cout << arr[1] << endl;         // 7
```

www.agh.edu.pl

wskaźniki - priorytety

```
int tab[10] = {9,8,7,6,5,4,3,2,1,0};
int *arr;

*arr++;         // *(arr++)
*++arr;         // *(++arr)
++*arr;         // ++(*arr)
(*arr)++;       // value incrementation

cout << *arr++ << endl;       // 9  - post-incrementation
cout << *arr++ << endl;       // 8
cout << *arr++ << endl;       // 7
cout << *arr++ << endl;       // 6
```

www.agh.edu.pl

wskaźniki - struktury

```
struct Product {
      int weight;
      float price;
};

Product p = {1, .5};
Product *x = &p;

p.weight = 2;
x->weight = 4;

Product *y = x;  // kopiowanie wskaźnika a nie zawartości struktury !!!!
```

- Wskaźnik na strukturę działa tak samo jak wskaźnik na zmiennej
- Adresowanie struktur:
  - operator . dla zmiennych
  - operator -> dla wskaźników

www.agh.edu.pl

wskaźniki - rozmiar

```
struct Product {
        int shape[20];
        float price;
}prod;


char *pc;
int *pi;
Product *pp = &prod;

cout << sizeof(prod) << endl;    // 84      ← wartość
cout << sizeof(pc) << endl;      // 8       ← wskaźnik
cout << sizeof(pi) << endl;      // 8       ← wskaźnik
cout << sizeof(pp) << endl;          // 8       ← wskaźnik
cout << sizeof(*pp) << endl;     // 84      ← wartość
```

- Wskaźnik pp to nie jest kopia zmiennej prod tylko jej adres
- Rozmiar wskaźnika jest stały, nie zależy od rozmiaru zmiennej na którą wskazuje

www.agh.edu.pl

wskaźniki - nieistniejący obiekt

- **x** ma waratość (przypadkowy adres)
- ***x** próbuje wyłuskać wartość spod przypadkowego adresu

```
int main() {
        int *x;

        // ok (ale bez sensu)
        cout << x << endl;

        // błąd !!!
        cout << *x << endl;
}
```

```
g++ -Wall cc.cc                    // C++14
cc.cc: In function ‘int main()’:
cc.cc:4:12: warning: ‘x’ is used uninitialized in this
function [-Wuninitialized]
 std::cout << x;
 ~~~~~~~~~~^~~~
```

www.agh.edu.pl

Porównywanie wskaźników

```
int tab[11];
int *start = tab;
int *end = &tab[10];
// init tab and print

while (end > start) {
    int tmp = *end;
    *end = *start;
    *start = tmp;
    end--;
    start++;
}
// print tab
```

» Pętla zamienia kolejność elementów w tablicy

www.agh.edu.pl

wskaźniki - alokacja pamięci

```
int *p = new int;
*p = 10;

cout << *p << endl;

delete p;
```

- Dynamiczne zarządzanie pamięcią
- new - rezerwacja pamięci
- delete - zwolnienie pamięci
- new zwraca wskaźnik na rezerwowany typ
- C++ nie ma garbage collector, należy explicite zwolnić zasoby (RAM)
- OS zwolni automatycznie pamięć po zakończeniu programu
- Brak zwolnienia nieużywanej pamięci jest błędem !!!

www.agh.edu.pl

Operator new - tablice

- Dynamiczna deklaracja tablicy
- Adresowanie jak w tablicy
- Zwalnianie tablicy

```
int size = 100;
int *p = new int[size];

for (size_t i = 0; i < size; ++i) {
    p[i] = i;
}

delete [] p;
```

www.agh.edu.pl

main() - argumenty

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

// ./ex01 -v -i file.txt

//int main(int argc, char *argv[]){
int main(int argc, char **argv){

    cout << argc << endl;
    cout << argv[0] << endl;
    cout << argv[1] << endl;
    cout << argv[2] << endl;
    cout << argv[3] << endl;
}
```

- Argumenty programu:
- argc: liczba argumentów
- argv: tablica zawierające wskaźniki do tablic z pojedynczymi argumentami programu (wskaźnik na wskaźnik)
- result:

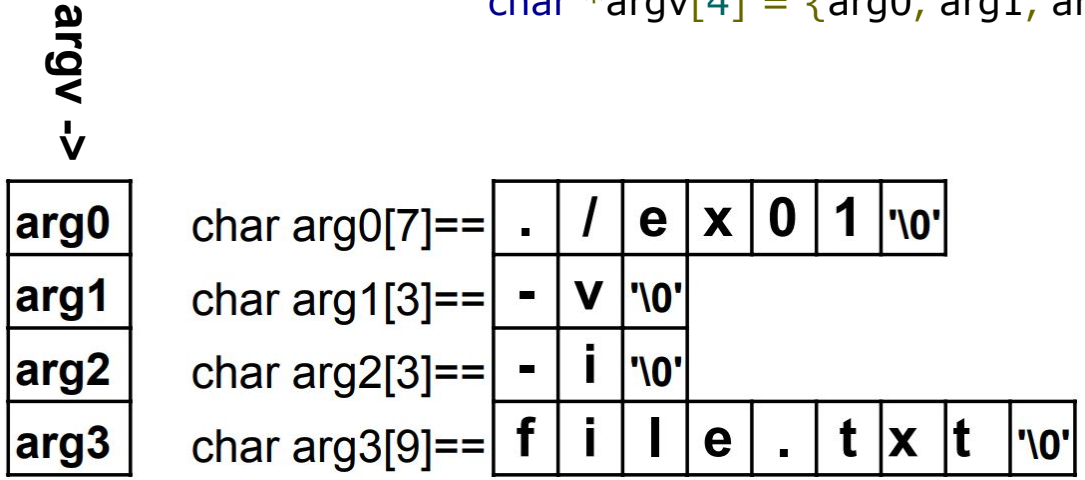
```
4
ex01
-v
-i
file.txt
```

www.agh.edu.pl

main() - argumenty

```
./ex01 -v -i file.txt
int main(int argc, char **argv){
...
}
```

```
char arg0[7] = "./ex01";
char arg1[3] = "-v";
char arg2[3] = "-i";
char arg3[9] = "file.txt";

char *argv[4] = {arg0, arg1, arg2, arg3};
```



Dziękuję