

# Podstawy programowania

## Materiały dydaktyczne do laboratorium

12 listopada 2022

---

### Zadania domowe

---

#### 1 Wskaźniki i zmienne dynamiczne, instrukcja przed zajęciami

Celem tych zajęć jest zrozumienie i oswojenie z technikami programowania przy pomocy wskaźników w języku C++. Proszę przeczytać rozdział 8. z "Symfonia C++ Standard". Oraz powtórzyć adekwatny wykład, dr. Dereniowskiego. Polecam też Państwa uwadze problem priorytetów operatorów, t.j. jakie miejsce zajmuje jednoargumentowa operacja dereferencji \*.

##### 1.1 Operatory pobrania adresu i dereferencji

###### Przykład 1.

```
#include<iostream>

using namespace std;

int main(){
    int a = 1, b = 2, c = 3;
    int *wsk;
    wsk = &a;
    *wsk = a + 5;
    wsk = &b;
    c = *wsk * 2;
    ++*wsk;
    cout << a << " " << b << " " << c << " " << *wsk << endl;
    return 0;
}
```

- Przeanalizuj powyższy program i bez kompilacji spróbuj przewidzieć wartości wyświetlone na ekranie.
- Zwróć uwagę na sposób deklarowania zmiennej wskaźnikowej `wsk`.
- Operator `&` pojawia się dwukrotnie, czy wiesz jakie jest jego zadanie?
- Przyjrzyj się proszę operatorowi dereferencji `*` (ten sam symbol jest wykorzystywany ponadto jako operator mnożenia i do deklarowania zmiennych wskaźnikowych), który zamienia zmienną wskaźnikową na obiekt znajdujący się pod wskazywanym adresem.
- Skompiluj program i zobacz, czy Twoje przewidywania były słuszne.

## 1.2 Arytmetyka wskaźników

Skompiluj i uruchom proszę poniższy kod.

### Przykład 2.

```
#include<iostream>

using namespace std;

int main(){
    double numbers[] = {1.1,3.2,5.3,7.4,9.5,0.6,2.7,4.8,6.9,8};
    double *pointer;
    pointer = numbers;
    cout << pointer << endl << pointer + 1 << endl;
    cout << *pointer << " " << *(pointer + 1) << endl << endl;
    pointer++;
    cout << pointer << endl << pointer + 2 << endl;
    cout << *pointer << " " << *(pointer + 2) << endl << endl;
    pointer += 5;
    cout << pointer << endl << pointer - 1 << endl;
    cout << *pointer << " " << *(pointer - 1) << endl << endl;
    cout << pointer - numbers;
    return 0;
}
```

- Dlaczego adresy wyświetlone w pierwszych dwóch liniach nie różnią się o 1?
- Czy usunięcie nawiasu z formuły `*(pointer + 1)` zmienia wynik? (jaka jest kolejność wykonywania operacji?)
- Zwróć uwagę na operacje jakie możesz wykonywać na zmiennych wskaźnikowych. Zastanów się jakie jeszcze operacje mają sens.
- czy wiesz dlaczego wyrażenie `pointer - numbers` dało wynik 6?

### 1.3 Przekazywanie wartości przez wskaźnik

Skompiluj i uruchom proszę poniższy kod.

**Przykład 3. Funkcja losująca tablicę losowej wielkości**

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;
void draw(int *n, int **arr){
    *n = rand() % 10 + 1;
    *arr = new int[*n];
    for (int i = 0; i < *n; ++i)
        (*arr)[i] = rand() % 100;
}
void print(int n, int *arr){
    for (; n--; ++arr)
        cout << *arr << ' ';
    cout << endl;
}
int main()
{
    int n, *pointer;
    srand(time(NULL));
    draw(&n, &pointer);
    print(n, pointer);
    delete [] pointer;
    return 0;
}
```

- Jaki będzie efekt działania takiego programu?
- W funkcji **draw** pierwszy parametr jest wskaźnikiem, aby zwrócić przez wskaźnik wylosowaną wielkość tablicy. Parametr **arr** jest wskaźnikiem na wskaźnik z podobnych powodów. Jak zadziała funkcja, gdy zrobimy pojedynczy wskaźnik?
- Zwróć proszę uwagę, na sposób wywołania funkcji **draw** w **main**. Funkcja pobiera adresy jako parametry, dlatego konieczne jest użycie operatora referencji **&**.
- Zwalnianie pamięci na końcu funkcji **main** nie jest konieczne. Tym niemniej dobrym przyzwyczajeniem jest, zwalnianie pamięci którą się samodzielnie zaalokowało.

### 1.4 Wskaźniki i struktury

**Przykład 4. Punkty na płaszczyźnie. Dynamiczna tablica struktur**

```
#include <iostream>

using namespace std;
```

```
struct point{
    double x;
    double y;
};

int print(point *pbegin){
    cout << pbegin->x << " " << pbegin->y << endl;
    return 0;
}

int print_arr(point *pbegin, int n){
    for (point *temp = pbegin; temp < pbegin + n; temp++)
        print(temp);
    return 0;
}

int main(){
    point *darr;
    darr = new point[3];
    darr[0].x = darr[0].y = 0;
    (*(darr + 1)).x = 1;
    (*(darr + 1)).y = 2;
    (darr + 2)->x = 4;
    (darr + 2)->y = 6.5;
    print_arr(darr, 3);
    delete[] darr;
    return 0;
}
```

- Zwróć uwagę na sposób generowania tablic dynamicznych struktur, czy z klasami i typami podstawowymi można postępować analogicznie?
- Proszę przyjrzyj się sposobom dostępu do zmiennych pod danym adresem (w tym wypadku w tablicy). (Dobłą praktyką programistyczną byłoby stosowanie jednolitego sposobu odnoszenia się do wartości pod zadanymi adresami.)
- Warto przeanalizować w jaki sposób zmienne przekazywane są do funkcji przez wskaźnik, proszę zauważyć, że pod adresem może kryć się pojedyncza zmienna lub cała tablica.
- Przyjrzyj się proszę sposobowi przeglądania tablicy w funkcji `print_arr(point *pbegin, int n)`. Sprawdź czy zwyczajne iterowanie tablicy indeksami też działa.
- Proszę napisać funkcję, która wylosuje tablicę punktów, funkcja powinna pobierać liczbę punktów i zwracać adres zaalokowanej tablicy. Niech nagłówek tej funkcji wygląda tak `point *draw_arr(int n)`.
  - Proszę przydzielić pamięć w tablicy za pomocą funkcji `malloc` z biblioteki `stdlib.h`. Proszę zwrócić uwagę na potrzebę rzutowania adresów zwracanych przez funkcję `malloc`.

- Proszę uważać na zwalnianie pamięci.
- Proszę napisać funkcję, która dla tablicy punktów wygeneruje i zwróci tablicę wskaźników na punkty posortowanych niemalejąco po ich odległości od początku układu współrzędnych. Nagłówek takiej funkcji to `point **sort(point *tab, int n)`. Funkcja ma zwrócić tablicę wskaźników, zatem zwraca wskaźnik na wskaźnik.

### Przykład 5. Łączenie w pary

Poniższy program pozwala w losowy sposób połączyć ludzi w pary. Program mógłby zostać zastosowany na przykład do losowania partnerów do projektu dwuosobowego.

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;
struct person{
    char nick[10];
    char email[30];
    struct person *match;
};
void print(struct person *a){
    if (a == NULL)
        cout << "Empty record\n";
    else
        cout << a->nick << '\t' << a->email;
    if (a->match != NULL)
        cout << '\t' << a->match->nick << '\n';
    else
        cout << '\n';
}
void print_all(struct person *arr, int n){
    for (int i = 0; i < n; ++i, ++arr)
        print(arr);
}
bool draw(struct person *arr, int n){
    if (n % 2 == 0){
        struct person *matched = arr;
        for (int i = 0; i < n / 2; ++i){
            struct person *temp;
            while (matched->match != NULL) //search unmatched
                ++matched;
            //draw next unmatched
            int ran = rand() % (n - 2 * i - 1) + 1;
            temp = matched;
            while (ran > 0) //searching next unpaired
                ++temp;
            if (temp->match == NULL) ran--;
        }
    }
}
```

```

        //matching
        matched->match = temp;
        temp->match = matched;
    }
    return true;
}
else
    return false;
}
int main()
{
    struct person arr[4] = { { "Olo", "alek@qwer.eu", NULL }, \
    {"zuza", "zuz@sp3.gda.pl", NULL}, \
    {"Misiu", "sos@sp3.gda.pl", NULL}, \
    {"Bolo", "bloleslaw1990@wwpp.pl", NULL} };
    srand(time(NULL));
    draw(arr, 4);
    print_all(arr, 4);
    return 0;
}

```

- Struktura **person** zawiera tablice znaków. Czy można te tablice zamienić na wskaźniki? Jakiej takiej zamiana przyniosła by korzyści i jakie zagrożenia?
- Struktura **person** zawiera wskaźnik na strukturę **person**, wskaźnik ten ma zawierać adres struktury odpowiadającej sparowanej osobie. Co by się stało, gdyby zamienić tę zmienną na zwykłą niewskaźnikową?
- W funkcji **print** zwróć uwagę na zabezpieczenia przed odwołaniem do pustych adresów. Co się stanie gdy owe zabezpieczenia zostaną pominięte?
- W funkcji **print** zastosowano zapis **a->partner->nick**, czy rozumiesz go?
- Dopisz fragment funkcji losuj dla **n** nieparzystych. W takim przypadku tylko jedna osoba powinna być sparowana sama ze sobą.
- Napisz funkcję, która wczyta z pliku dane osób do losowania.

## 1.5 Wskaźniki na funkcje

**Przykład 6. Obliczanie pola pod wykresem funkcji na określonym przedziale**

```
#include <stdio.h>
```

```

//Exemplary function.
double fu(double x){
    return x / (x + 10);
}
//Exemplary function, returns value of polynomial in a point
double poly(double x){

```

```

    int i, st = 4;
    double coef[5] = { 1, 0, 0, 1, 0 }, val = 0, pow;
    for (i = 0, pow = 1; i <= st; i++, pow *= x)
        val += coef[i] * pow;
    return val;
}

//Function divides field under function graph into rectangles and
//adds up their fields.
double field_under_graph(double a, double b,\
    double(*fu)(double), double h){
    double sum = 0;
    for (; a < b; a += h)
        sum += h * (fu(a) + fu(a + h)) / 2;
    return sum;
}

double(*Fu(int n))(double){
    if (n == 1) return fu;
    if (n == 2) return poly;
    return NULL;
}

typedef double(*Fn1Arg)(double);

Fn1Arg Fn(int n) {
    if (n == 1) return fu;
    if (n == 2) return poly;
    return NULL;
}

int main(){
    int cyfry = 10;
    printf("%lf\n", field_under_graph(-1, 1, poly, 1.0/(111 << 16)));
    return 0;
}

```

- Przetestuj funkcję `field_under_graph` na różnych funkcjach matematycznych i przedziałach oraz z różną wartością parametru `h`, jeśli potrafisz policzyć wyniki matematycznie sprawdź poprawność obliczeń.
- Przyjrzyj się sposobowi deklarowania wskaźnika na funkcję (deklaracja funkcji `field_under_graph`)
- Funkcja `field_under_graph` powinna reagować na sytuację, gdy wykres funkcji przecina osie układu współrzędnych. Popraw funkcję tak by reagowała na takie zdarzenie.
- Funkcje `Fu` i robiąca dokładnie to samo funkcja `Fn` nie zostały wykorzystane w programie, potrafisz odgadnąć co robią?

## 2 Zadania do samodzielnego rozwiązania

**Zadanie 1.** Zmodyfikuj przykład 4. tak by zawierał jedną funkcję (zastępującą `print` i `print_arr`) z domyślną wartością parametru `number = 1`?

**Zadanie 2.** Proszę napisać funkcję, która wypisze z tablicy wszystkie liczby większe od średniej, funkcja pobiera jako parametr wskaźnik na tablicę i liczbę liczb. Proszę napisać funkcję bez wykorzystania nawiasu kwadratowego.

**Zadanie 3.** Bez używania nawiasu kwadratowego, proszę napisać funkcję, która zwróci maksimum pewnej tablicy podanej jako parametr. Liczba elementów również jest parametrem funkcji.

**Zadanie 4.** Proszę napisać serię funkcji obsługujących tablice liczb całkowitych o rozmiarze zapisanym na elemencie o indeksie 0.

- `int * utworz(int rozmiar)` tworzy i zwraca nową tablicę wypełnioną zerami.
- `void losuj(int *tablica, int zakres_min, int zakres_max)` wypełnia tablicę wylosowanymi elementami z podanego zakresu.
- `int usun(int *tablica, int wartosc)` usuwa wszystkie elementy równe `wartosc`. Zwraca liczbę usuniętych elementów.
- `int usun(int **tablica)` usuwa wszystkie elementy z tablicy. Zwalnia pamięć i zapisuje tablicę jako `NULL`.
- `int sumuj(int *tablica)` zwraca sumę elementów w tablicy.
- `int * sklej(int *tablica1, int *tablica2)` tworzy i zwraca nową tablicę wypełnioną elementami z `tablica1` i `tablica2`.
- `void sortuj(int *tablica)` sortuje elementy zawarte w tablicy.

**Zadanie 5.** Proszę napisać analogiczne funkcje jak powyżej dla tablicy zapisanej jako struktura (wskaźnik na blok pamięci i liczba elementów).

**Zadanie 6.** Proszę napisać analogiczne funkcje jak powyżej dla tablicy zapisanej jako lista jednokierunkowa (porównaj wykład z dynamicznych struktur danych).

## 3 Quiz

1. Co wypisze poniższy kod?

```
int a = 5, *wsk = &a;
(*wsk) += 3;
cout << a;
```

- (a) 5
- (b) 6
- (c) 8
- (d) nie skompiluje się



2. Dany jest c-string (tablica znaków zakończona znakiem `'\0'`) `char *napis = "jakis napis";` Które z poniższych wyrażeń stanowią poprawne odwołanie do 3 elementu (litera k)?

- a. `napis + 2;`
- b. `*napis + 2;`
- c. `*(napis + 2);`
- d. `napis[2];`

3. Dana jest tablica i wskaźnik `int tab[5] = 1, 3, 5, 7, *wsk = tab + 2;` co wypiszą następujące komendy?

- a. `cout << *wsk << endl;`
- b. `cout << wsk[2] << endl;`
- c. `cout << wsk[-1] << endl;`
- d. `cout << wsk - tab << endl;`
- e. `cout << wsk << endl;`

4. Dane są zmienne

```
char c;  
int a, *wska;  
double x, *wskx;  
long b;  
long long d;
```

Co wypiszą następujące instrukcje?

- a. `cout << sizeof(c) << endl;`
- b. `cout << sizeof(a) << endl;`
- c. `cout << sizeof(&a) << endl;`
- d. `cout << sizeof(wska) << endl;`
- e. `cout << sizeof(wskx) << endl;`
- f. `cout << sizeof(x) << endl;`
- g. `cout << sizeof(*wskx) << endl;`
- h. `cout << sizeof(b) << endl;`
- i. `cout << sizeof(d) << endl;`

### Quiz odpowiedzi

- 1. Instrukcja `(*wsk) += 3;` zmienia wartość pod adresem `wsk`, czyli zmienną `a`, zatem poprawna odpowiedź to c.
- 2. Poprawne odpowiedzi to c i d. W punkcie a otrzymujemy adres 3. litery, zaś w punkcie b wartość pierwszej litery jest powiększana o 2.
- 3. Instrukcje wypiszą:
  - (a) 5 bo `wsk` wskazuje na element z indeksem 2.
  - (b) 0 bo odwołujemy się do elementu `tab` z indeksem 4 ten natomiast przy inicjalizacji dostał wartość 0

- (c) 3 bo operacja `wsk[i]` jest równoważna `*(wsk+i)`.
  - (d) 2 ponieważ o tyle intigerów różnią się te dwa wskaźniki.
  - (e) Adres przechowywany w zmiennej `wsk`. Trudno przewidzieć z góry konkretną wartość
4. Insrtukeje wpiszą wielkości w byte'ach odpowiednich zmienny, które mogą różnić się w zależności od systemu i kompilatora:
- (a) 1
  - (b) 4 choć standard mówi, że wielkość intigera jest przynajmniej 2
  - (c) W zależności od wielkości szyny adresowej 4 lub 8.
  - (d) j.w.
  - (e) j.w.
  - (f) 8 choć wartość ta zależy od systemu i kompilatora
  - (g) j.w.
  - (h) 4.
  - (i) 8