Podstawy programowania
Materiały dydaktyczne do laboratorium

9 listopada 2016

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 tablica[10] = {1.1, 3.2, 5.3, 7.4, 9.5, 0.6, 2.7, 4.8, 6.9, 8};
  double *wsk;
  wsk = tablica;
  cout << wsk << endl << wsk + 1 << endl;
  cout << *wsk << " " << *(wsk + 1) << endl << endl;
  wsk++;
  cout << wsk << endl << wsk + 2 << endl;
  cout << *wsk << " " << *(wsk + 2) << endl << endl;
  wsk += 5;
  cout << wsk << endl << wsk - 1 << endl;
  cout << *wsk << " " << *(wsk - 1) << endl << endl;
  cout << wsk - tablica;
  return 0;
}
```

- Dlaczego adresy wyświetlone w pierwszych dwóch liniach nie różnią się o 1?
- Czy usunięcie nawiasu z formuły `*(wsk + 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 `wsk - tablica` 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 losuj(int *n, int **tab){
  *n = rand() % 10 + 1;
  *tab = new int[*n];
  for (int i = 0; i < *n; ++i)
    (*tab)[i] = rand() % 100;
}
void wypisz(int n, int *tab){
  for (; n--; ++tab)
    cout << *tab << ' ';
  cout << endl;
}
int main()
{
  int n, *wsk;
  srand(time(NULL));
  losuj(&n, &wsk);
  wypisz(n, wsk);
  delete[] wsk;
  return 0;
}
```

- Jaki będzie efekt działania takiego programu?
- W funkcji losuj pierwszy parametr jest wskaźnikiem, aby zwrócić przez wskaźnik wylosowaną wielkość tablicy. Parametr tab jest wskaźnikiem na wskaźnik z podobnych powodów. Jak zadziała funkcja, gdy zrobimy pojedynczy wskaźnik?
- Zwróć proszę uwagę, na sposób wywołania funkcji losuj w main. Funkcja pobiera adresy jako parametry, dlatego konieczne jest użycie operatora referencji &.
- Zwalnianie pamięci na końcu funkcji main nie jest konieczne, ale jest dobrym przyzwyczajeniem, 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 punkt{
  double x;
  double y;
};

int wypisz(punkt *wskp){
  cout << wskp->x << " " << wskp->y << endl;
  return 0;
}

int wypisz_tab(punkt *wskp, int liczba){
  for(punkt *temp = wskp; temp < wskp + liczba; temp++)
    wypisz(temp);
  return 0;
}

int main(){
  punkt *tab;
  tab = new punkt[3];
  tab[0].x = tab[0].y = 0;
  (*(tab+1)).x = 1;
  (*(tab+1)).y = 2;
  (tab+2)->x = 4;
  (tab+2)->y = 6.5;
  wypisz_tab(tab,3);
  delete[] tab;
  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). (Dobrą 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 `wypisz_tab(punkt *wskp, int liczba)`. 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 `punkt *losuj_tab(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 wygeneruję i zwróci tablicę wskaźników na punkty posortowanych niemalejąco po ich odległości od początku układu współrzędnych. Nagłówek takiej funkcji to `punkt **sortuj(punkt *tab, int n)`. Funkcja ma zwrócić tablice 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 osoba{
  char nick[10];
  char email[30];
  struct osoba *partner;
};
void wypisz(struct osoba *a){
  if (a == NULL)
    cout << "Pusty rekord\n";
  else
    cout << a->nick << '\t' << a->email;
  if (a->partner != NULL)
    cout << '\t' << a->partner->nick << '\n';
  else
    cout << '\n';
}
void wypisz_wszystkie(struct osoba *tab, int n){
  for (int i = 0; i < n; ++i, ++tab)
    wypisz(tab);
}
bool losowanie(struct osoba *tab, int n){
  if (n % 2 == 0){
    struct osoba *parowany = tab;
    for (int i = 0; i < n / 2; ++i){
      struct osoba *temp;
      while (parowany->partner != NULL)//szukanie niesparowanego
        ++parowany;
      //Losowanie, który następny nieprzydzielony sprarować
      int los = rand() % (n - 2 * i - 1) + 1;
      temp = parowany;
      while (los > 0){//szukanie niesparowanego odległego o los
        ++temp;
        if (temp->partner == NULL) los--;
      }
```

```
      //parowanie
      parowany->partner = temp;
      temp->partner = parowany;
    }
    return true;
  }
  else
    return false;
}
int main()
{
  struct osoba tab[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));
  losowanie(tab, 4);
  wypisz_wszystkie(tab, 4);
  return 0;
}
```

- Struktura osoba zawiera tablice znaków. Czy można te tablice zamienić na wskaźniki? Jakie taka zamiana przyniosła by korzyści i jakie zagrożenia?
- Struktura osoba zawiera wskaźnik na strukturę osoba, wskaźnik ten ma zawierać adres struktury odpowiadającej sparowanej osobie. Co by się stało, gdyby zamienić tę zmienną na zwykłą niewskaźnikową?
- W funkcji `wypisz` zwróć uwagę na zabezpieczenia przed odwołaniem do pustych adresów. Co się stanie gdy owe zabezpieczenia zostaną pominięte?
- W funkcji `wypisz` 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>

//Przykładowa funkcja.
double fu(double x){
  return x/(x+10);
}
//Przykładowa funkcja, oblicza wartość wielomianu w punkcie.
double wiel(double x){
```

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

//Funkcja dzieli obszar pod wykresem funkcji w przedziale [a,b] na prostokąty
//szerokości h i sumuje pola takich prostokątów.
double pole_pod_wykresem(double a, double b, double (*fu)(double), double h){
  double suma = 0;
  for(; a < b; a+=h)
    suma += h * (fu(a)+fu(a+h)) / 2;
  return suma;
}

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

typedef double(*Fn1Arg)(double);

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

int main(){
  int cyfry = 10;
  printf("%lf\n", pole_pod_wykresem(-1, 1, wiel, 1.0/(1ll<<16)));
  return 0;
}
```

- Przetestuj funkcję `pole_pod_wykresem` 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 `pole_pod_wykresem`)
- Funkcja `pole_pod_wykresem` 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 funkcje (zastępującą `wypisz` i `wypisz_tab`) z domyślną wartością parametru `liczba = 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 wypełniona 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. Zwarlnia 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 `tablcia1` 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 elementy tab z indeksem 4 ten natomiast przy inicializacji 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. Insrtukcje 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