Podstawy Informatyki –
wskaźniki, referencje, tablice, funkcje

25.10.2019

dr inż. Michał Rad

Plan wykładu:

1. Organizacja pamięci
2. Wskaźniki
3. Referencje
4. Tablice
5. Funkcje
   a. Przekazywanie i zwracanie wartości
   b. Rekurencja

Organizacja pamięci

» Co przechowujemy w pamięci operacyjnej podczas działania programu?

- kod programu
- zmienne globalne (dane statyczne)
- zmienne lokalne (stos - *stack*)
- dane alokowane dynamicznie (sterta - *heap*)

Wskaźniki

```
int liczba1 = 55;
```

Typ: int
Nazwa: liczba1
Wartość: 55
Adres: 1532 *(przykładowy)*

» Adresu zmiennej nie możemy zmienić ani nadać ale możemy odczytać:

```
&nazwa_zmiennej;
&liczba1;
```

Wskaźniki

- Zmienna która przechowuje wartość adresu komórki pamięci to wskaźnik
- W języku C/C++ istnieje specjalny typ wskaźnikowy, a właściwie typy wskaźnikowe do każdego rodzaju zmiennych
- Czyli liczby typu **int** wskazujemy wskaźnikiem typu ***int**, **float** typu ***float** itd…

Przykład

```
int main()
{ int zm=5;
  int* wsk_do_int=&zm;
cout<<"Wart zm: "<<zmienna1<<"Adres:"<<wsk_do_int;
cout<<endl;


cout<<"Wart pod adresem "<<wsk_do_int<<"to:
"<<*wsk_do_int;
}
```

Wskaźniki

» Wyłuskanie wartości

```
*wsk_do_int;
```

» Dlatego potrzebny jest typ – czyli informacja na jaką zmienną wskazuje wskaźnik

» Istnieją także wskaźniki typu *void (bez typu) ale o tym później

Wskaźniki

- Jedyną wartość jaką można bezpiecznie nadać wskaźnikowi to wartość NULL (nullptr dla c++11)
- Oznacza to że wskaźnik nie wskazuje na nic.
- Jeżeli spróbujemy wyłuskać wartość spod wskaźnika równego NULL spowoduje to błąd

Referencje

- Referencję można z pewnym uproszczeniem uznać za inną nazwę, (alias) dla tej samej zmiennej
- Referencja ma największe znaczenie w kontekście funkcji (o tym później)
- Referencja dotyczy tylko języka C++

Przykład

```
int main()
{  int zm_x=5;
   int& referencja_X=x;
cout<<"Wart zm_x: "<<zm_x<<endl;
referencja_x=7;
cout<<"Wart zm_x: "<<zm_x<<endl;
}
```

Tablice

» Zmienna, która może przechować wiele wartości określonego typu to tablica

» Deklaracja tablicy:

```
float nazwa_tablicy[rozmiar];
```

Przykład

```
int main()
{ float tab1[5]={1.1,2.2,3.3,4.1,5.0};
//nie trzeba inicjować wszystkich elementów
  cout<<"trzecia wartość w tablicy to: "<<tab1[2]
<<endl;
for(int i=0;i<5;i++)
        tab1[i]=10;
cout<<" a teraz trzecia wartość w tablicy to:
"<<tab1[2]<<endl;
}
```

Tablice

- W języku C i C++ tablice to twór dość niskopoziomowy
- Jest to po prostu pewna ilość elementów (równa rozmiarowi) zapisana w kolejnych komórkach pamięci
- Nazwa tablicy to wskaźnik do jej pierwszego elementu

Tablice

» Dostęp do elementów tablicy realizowany jest za pomocą indeksu

```
tabl[i];
```

» Pierwsza komórka ma indeks równy 0

» UWAGA: nigdzie i nigdy nie jest sprawdzane czy indeks nie wychodzi poza tablicę – to już rola programisty

Przykład

```
int main()
{ float tab1[5]={1.1,2.2,3.3,4.1,5.0};
  cout<<"trzecia wartość w tablicy to: "<<tab1[2]
<<endl;
for(int i=0;i<5;i++)
        tab1[i]=10;
cout<<" a teraz trzecia wartość w tablicy to:
"<<tab1[2]<<endl;
}
```

Tablice wielowymiarowe

» Tablice mogą być również wielowymiarowe

```
float nazwa_tablicy[n][m];
```

» Sposób dostępu do elementów jest podobny

```
tabl1[x][y];
```

Tablice wielowymiarowe

» Inicjacja tablicy wielowymiarowej:

```
int main()
{ float tab1[5][2]={{1.1,2.2},{3.3,4.1},{5.0,2.1}};


//nadanie wartości każdej komórce tablicy w
programie:
for(int j=0;j<2;j++){
   for(int i=0;i<5;i++){
        tab1[i][j]=i*j;}
          }
}
```

AGH

Funkcje

- Język C i C++ jak wiele innych języków daje możliwość modułowego tworzenia programów
- Tworzenie i używanie funkcji można uznać właśnie za taką funkcjonalność

Co to jest funkcja?

» Funkcja to osobny fragment kodu który ma własną nazwę, może pobierać argumenty, wykonywać zadane operacje i zwracać wartość

» Funkcja również może używać własnych lokalnych zmiennych oraz zmiennych globalnych

Przykład

```
float moja_funkcja(int n,float x)
{
 float wynik=n*x;
 return wynik;
}
int main()
{ float a;
  a=moja_funkcja(2,3.1);
  cout<<”a = ”<<a<<endl;
}
```

Po co stosować funkcje?

- Wielokrotne użycie kodu (zarówno w jednym programie jak i w różnych programach)
- Czytelniejszy program główny
- Hermetyzacja kodu
- Możliwość stosowania rekurencji

Podnoszenie do potęgi

```
float n_power(float x,int n)
{
   float wynik=1;
   int i;
    for(i=1;i<=n;i++)
      wynik=wynik*x;
return wynik;
}
int main()
{
cout<<n_power(2,8)<<endl;
}
```

.. rekurencyjnie

```
float n_power(float x,int n)
{
  float wynik;
  if(n==1) { return x;}
  else{
  wynik=x*n_power(x,n-1);
  return wynik;
      }
}
int main()
{
cout<<n_power(2,8)<<endl;
}
```

Co dzieje się w trakcie wywołania funkcji?

- Następuje skok do miejsca w pamięci gdzie zapisany jest kod funkcji
- Wartości parametrów podanych inicjują zmienne lokalne funkcji, które to zmienne lokalne są odkładane na stosie
- funkcja działa, a gdy zakończy działanie w miejscu wywołania pojawia się wartość zwrócona przez funkcję

Funkcje c.d.

- W języku C i C++ funkcja może zwrócić tylko jedną wartość bądź obiekt
- Jeżeli chcemy zmienić wartość więcej niż jednej zmiennej trzeba radzić sobie inaczej
- Z pomocą przychodzą wskaźniki i referencje (C++)

Przykład (wskaźniki)

```
void zwieksz(int* x,int* y,int* z)
{
*x=*x+1; *y=*y+1; *z=*z+1;
}
int main()
{
int a=1,b=2,c=3;
cout<<a<<" "<<b<<" "<<c<<endl;
zwieksz(&a,&b,&c);
cout<<a<<" "<<b<<" "<<c<<endl;
}
```

Przykład (referencja)

```
void zwieksz(int& x,int& y,int& z)
{
x++,y++,z++;
}
int main()
{
int a=1,b=2,c=3;
cout<<a<<" "<<b<<" "<<c<<endl;
zwieksz(a,b,c);
cout<<a<<" "<<b<<" "<<c<<endl;
}
```

Przekazywanie tablic jako argumentów

```
void zwieksz(int tab[])
{
tab[0]++; tab[1]++; tab[2]++;
}
int main()
{
int liczby[]={5,6,7};
cout<<liczby[0]<<" "<<liczby[1]<<"
"<<liczby[2]<<endl;
zwieksz(liczby);
cout<<liczby[0]<<" "<<liczby[1]<<"
"<<liczby[2]<<endl;
}
```

Quiz!

» Po zalogowaniu się można wybrać swój nick

» Nick będzie widoczny na ekranie