Informatyka 2

Wprowadzenie do dynamiczny struktur danych
Wskaźniki. Dynamiczna alokacja pamięci

Paweł Strączyński
pstraczynski@tu.kielce.pl

Dynamiczne struktury danych

Strukturą danych nazywamy „pojemnik na dane”, który układa dane w odpowiedni sposób np. tablica, rekord (struktura).

Dynamiczną strukturą danych nazywamy taką strukturę danych której wielkość i stopień złożoności może zmieniać się podczas działania programu. Podstawowe dynamiczne struktury danych to:

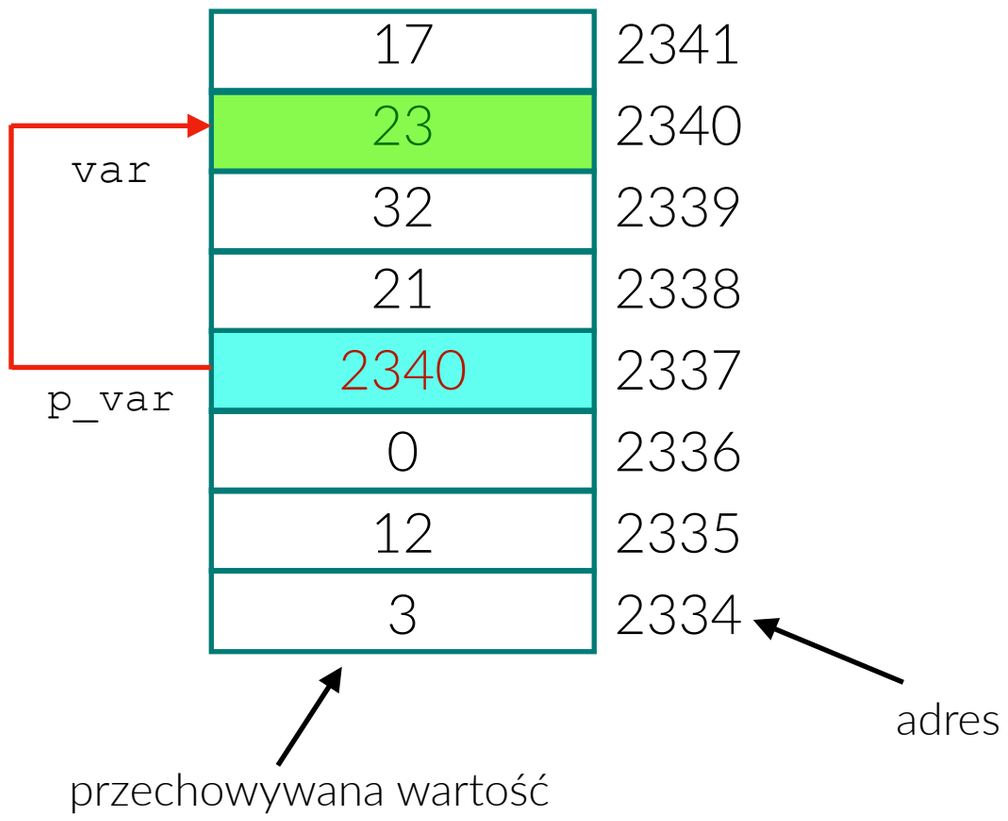
- stos,
- kolejka,
- lista,
- drzewo binarne.

Wskaźniki

Dynamiczne struktury danych zwykle tworzone są z wykorzystaniem wskaźników.

Wskaźnik to specjalny rodzaj zmiennej w której zapisany jest adres. Wskaźnik wskazuje zatem gdzie w pamięci komputera zapisana jest zmienna na którą ten wskaźnik wskazuje.

Zmienna wskaźnikowa `p_var` przechowuje adres zmiennej `var` - wskaźnik `p_var` wskazuje na zmienną var.



Arytmetyka wskaźników w języku C

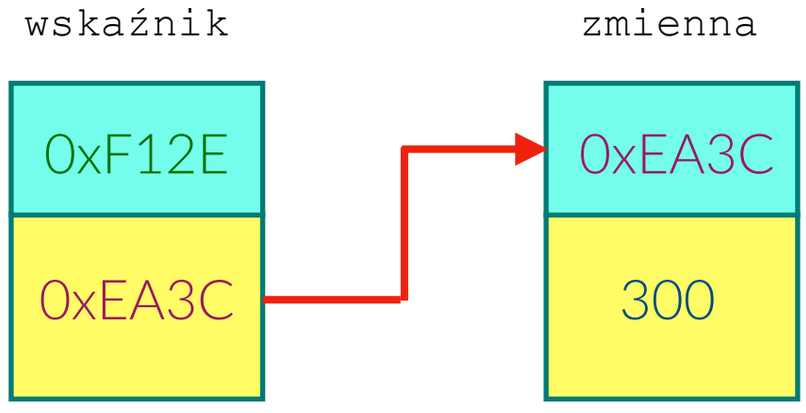
```c
/* wskaznik.c */

#include <stdio.h>

int main ()
{
    int zmienna = 300;
    int *wskaznik = &zmienna;
    printf(„Wartosc zmiennej: %d\n”,zmienna);
    printf(„Wartosc zmiennej: %d\n”,*wskaznik);
    printf(„Adres zmiennej: %d\n”,wskaznik);
    printf(„Adres zmiennej: %d\n”,&zmienna);
}
```

```
Wartosc zmiennej: 300
Wartosc zmiennej: 300
Adres zmiennej: 0xEA3C
Adres zmiennej: 0xEA3C
```

Symbol	Znaczenienie	Użycie
*	weź wartość x	*x
*	deklaracja wskaźnika	int *x;
&	weź adres	&x



Arytmetyka wskaźników w języku C

```c
/* wskaznik1.c */

#include <stdio.h>

int main ()
{
    int zmienna = 300;
    int *wskaznik = &zmienna;
    printf(„Wartosc zmiennej: %d\n”,zmienna);
    printf(„Wartosc zmiennej: %d\n”,*wskaznik);
    printf(„Adres zmiennej: %d\n”,wskaznik);
    printf(„Adres zmiennej: %d\n”,&zmienna);
}
```

Zmiennej wskaźnikowej `wskaznik` przypisany został adres zmiennej `zmienna` z wykorzystaniem operatora pobrania adresu

Wskaźniki posiadają typ - na jakiego typu zmienną wskazują

```
Wartosc zmiennej: 300
Wartosc zmiennej: 300
Adres zmiennej: 0xEA3C
Adres zmiennej: 0xEA3C
```

Symbol	Znaczenienie	Użycie
*	weź wartość x	*x
*	deklaracja wskaźnika	int *x;
&	weź adres	&x

Arytmetyka wskaźników w języku C

```c
/* wskaznik1.c */

#include <stdio.h>

int main ()
{
    int zmienna = 300;
    int *wskaznik = &zmienna;
    printf(„Wartosc zmiennej: %d\n”,zmienna);
    printf(„Wartosc zmiennej: %d\n”,*wskaznik);
    printf(„Adres zmiennej: %d\n”,wskaznik);
    printf(„Adres zmiennej: %d\n”,&zmienna);
}
```

Zmiennej wskaźnikowej `wskaznik` przypisany został adres zmiennej `zmienna` z wykorzystaniem operatora pobrania adresu

Zawartość zmiennej na która wskazuje wskaźnik może zostać pobrana z użyciem operatora wyłuskania *

Wskaźniki posiadają typ - na jakiego typu zmienną wskazują

```
Wartosc zmiennej: 300
Wartosc zmiennej: 300
Adres zmiennej: 0xEA3C
Adres zmiennej: 0xEA3C
```

Symbol	Znaczenienie	Użycie
*	weź wartość x	*x
*	deklaracja wskaźnika	int *x;
&	weź adres	&x

Arytmetyka wskaźników w języku C

Tworząc zmienne wskaźnikowe łączmy gwizdkę ze zmienną a nie z typem. Pomoże to uniknąć niespodziewanych pomyłek!

```c
/* wskaznik2.c */

#include <stdio.h>

int main ()
{
    int* a;
    int *b; //takiego zapisu lepiej unikać!
    int* c,d; //takiego zapisu lepiej unikać, tylko c jest wskaźnikiem!
    int *e,*f;
}
```

Zmienne `a`, `b`, `c`, `e` i `f` są wskaźnikami ale zmienna `d` nie jest wskaźnikiem!

Statyczna alokacja pamięci w języku C

Dotychczas tworząc tablicę na etapie jej tworzenia musieliśmy określić jej wielkość. W przypadku gdy nie została całkowicie wypełniona rezerwowaliśmy nadmiarowo pamięć komputera.

```
/*tab_statyczna.c */

#include <stdio.h>

int main ()
{
    int tab[100]; //program zawsze zablokuje miejsce w pamięci na 100 wartości
    int n,i;
    printf(„Podaj ilość wprowadzanych liczb”);
    scanf(„%d”,&n); //gdy użytkownika poda liczbe <100 część zarezerwowanej
                    //pamięci zostanie niewykorzystana
    for (i=0; i<n; i++)
    {
      scanf(„%d”,&tab[i]); // lub scanf(„%d”,tab+i);
    }
    for (i=0; i<n; i++)
    {
      printf(„Liczba %d to: %d”,i,tab[i]);
    }
}
```

Nazwa tablicy jest wskaźnikiem na jej pierwszy element. Zapis `tab+i` jest zatem tu równoważny z `&tab[i]`, gdyż wskaźnik zostaje przesunięty o `i` adresów

Dynamiczna alokacja pamięci w języku C

Podstawową funkcją do alokacji pamięci w języku C jest funkcja `malloc`. Jest to funkcja która zwraca wskaźnik do zaalokowanego obszaru pamięci.

```
/*tab_dynamiczna.c */

#include <stdio.h>

int main ()
{
    int *tab;
    int n,i;
    printf(„Podaj ilość wprowadzanych liczb”);
    scanf(„%d”,&n);
    tab = (int*)malloc(n*sizeof(*tab));
    for (i=0; i<n; i++)
    {
      scanf(„%d”,tab+i);
    }
    for (i=0; i<n; i++)
    {
      printf(„Liczba %d to: %d”,i,*(tab+i));
    }
    free(tab);
}
```

Dynamiczna alokacja pamięci w języku C

Podstawową funkcją do alokacji pamięci w języku C jest funkcja `malloc`. Jest to funkcja która zwraca wskaźnik do zaalokowanego obszaru pamięci.

```c
/*tab_dynamiczna.c */

#include <stdio.h>

int main ()
{
    int *tab;
    int n,i;
    printf(„Podaj ilość wprowadzanych liczb”);
    scanf(„%d”,&n);
    tab = (int*)malloc(n*sizeof(*tab));
    for (i=0; i<n; i++)
    {
      scanf(„%d”,tab+i);
    }
    for (i=0; i<n; i++)
    {
      printf(„Liczba %d to: %d”,i,*(tab+i));
    }
    free(tab);
}
```

Funkcja `malloc` rezerwuje `n*sizeof(*tab)` bajtów w pamięci komputera a więc n*4 gdyż zmienna typu int zajmuje 4 bajty. Równoznaczny byłby zapis `n*sizeof(int)` ale w przypadku zmiany typu tablicy zapis taki utrudni wykrycie błędu.

Dynamiczna alokacja pamięci w języku C

Podstawową funkcją do alokacji pamięci w języku C jest funkcja `malloc`. Jest to funkcja która zwraca wskaźnik do zaalokowanego obszaru pamięci.

```c
/*tab_dynamiczna.c */

#include <stdio.h>

int main ()
{
    int *tab;
    int n,i;
    printf(„Podaj ilość wprowadzanych liczb”);
    scanf(„%d”,&n);
    tab = (int*)malloc(n*sizeof(*tab));
    for (i=0; i<n; i++)
    {
      scanf(„%d”,tab+i);
    }
    for (i=0; i<n; i++)
    {
      printf(„Liczba %d to: %d”,i,*(tab+i));
    }
    free(tab);
}
```

Robimy rzutowanie tego co zwraca funkcja `malloc` na typ wskaźnika

Funkcja `malloc` rezerwuje `n*sizeof(*tab)` bajtów w pamięci komputera a więc n*4 gdyż zmienna typu int zajmuje 4 bajty. Równoznaczny byłby zapis `n*sizeof(int)` ale w przypadku zmiany typu tablicy zapis taki utrudni wykrycie błędu.

Dynamiczna alokacja pamięci w języku C

Podstawową funkcją do alokacji pamięci w języku C jest funkcja `malloc`. Jest to funkcja która zwraca wskaźnik do zaalokowanego obszaru pamięci.

```c
/*tab_dynamiczna.c */

#include <stdio.h>

int main ()
{
    int *tab;
    int n,i;
    printf(„Podaj ilość wprowadzanych liczb”);
    scanf(„%d”,&n);
    tab = (int*)malloc(n*sizeof(*tab));
    for (i=0; i<n; i++)
    {
      scanf(„%d”,tab+i);
    }
    for (i=0; i<n; i++)
    {
      printf(„Liczba %d to: %d”,i,*(tab+i));
    }
    free(tab);
}
```

Robimy rzutowanie tego co zwraca funkcja `malloc` na typ wskaźnika

Funkcja `malloc` rezerwuje `n*sizeof(*tab)` bajtów w pamięci komputera a więc n*4 gdyż zmienna typu int zajmuje 4 bajty. Równoznaczny byłby zapis `n*sizeof(int)` ale w przypadku zmiany typu tablicy zapis taki utrudni wykrycie błędu.

UWAGA - Zapis `*(tab+i)` nie jest równoważny z `*tab+i`. W pierwszym przesuwamy wskaźnik i dopiero pobieramy zawartość na która ten przesunięty wskaźnik wskazuje. W drugim przypadku pobieramy zawartość na która wskazuje wskaźnik i ja zwiększamy o liczbę `i`

Dynamiczna alokacja pamięci w języku C

Podstawową funkcją do alokacji pamięci w języku C jest funkcja `malloc`. Jest to funkcja która zwraca wskaźnik do zaalokowanego obszaru pamięci.

```c
/*tab_dynamiczna.c */

#include <stdio.h>

int main ()
{
    int *tab;
    int n,i;
    printf(„Podaj ilość wprowadzanych liczb”);
    scanf(„%d”,&n);
    tab = (int*)malloc(n*sizeof(*tab));
    for (i=0; i<n; i++)
    {
      scanf(„%d”,tab+i);
    }
    for (i=0; i<n; i++)
    {
      printf(„Liczba %d to: %d”,i,*(tab+i));
    }
    free(tab);
}
```

Robimy rzutowanie tego co zwraca funkcja `malloc` na typ wskaźnika

Funkcja `malloc` rezerwuje `n*sizeof(*tab)` bajtów w pamięci komputera a więc n*4 gdyż zmienna typu int zajmuje 4 bajty. Równoznaczny byłby zapis `n*sizeof(int)` ale w przypadku zmiany typu tablicy zapis taki utrudni wykrycie błędu.

UWAGA - Zapis `*(tab+i)` nie jest równoważny z `*tab+i`. W pierwszym przesuwamy wskaźnik i dopiero pobieramy zawartość na która ten przesunięty wskaźnik wskazuje. W drugim przypadku pobieramy zawartość na która wskazuje wskaźnik i ja zwiększamy o liczbę `i`

Należy pamietać o zwalnaninu pamięci funkcja `free` aby nie doszło do **wycieku pamięci**