DYNAMICZNE PRZYDZIELANIE PAMIECI

Pamięć komputera, dostępna dla programu, dzieli się na cztery obszary:

- **kod programu**,
- dane **statyczne** (np. stałe i zmienne globalne programu),
- dane **automatyczne** (zmienne lokalne funkcji - tworzone i usuwane automatycznie przez kompilator) → tzw. STOS (*ang. stack*)
- dane **dynamiczne** (zmienne, które można tworzyć i usuwać w dowolnym momencie pracy programu) → w pamięci wolnej komputera → tzw. STERTA (*ang. heap*)

Zmienne dynamiczne → są to zmienne tworzone przez programistę w pamięci wolnej komputera (na stercie)

→ dostęp do takiej zmiennej możliwy jest jedynie poprzez jej adres w pamięci (przechowywany w zmiennej wskaźnikowej).

W języku „**C**" do dynamicznego przydzielania pamięci (tworzenia zmiennych dynamicznych) służyły specjalne funkcje z biblioteki **< alloc.h >**

```
void *malloc( size_t  rozmiar );          // przydział bloku o zadanej wielkosci
void *calloc( size_t  il_elementow,  size_t rozmiar);   // przydział tablicy
void  free( void *wskaznik);              // zwolnienie wskazywanego obszaru
unsigned  coreleft( void );               // sprawdzenie wolnego miejsca na stercie
```

np.
```
void main( void )
{
   int *wsk;          // zmienna wskaźnikowa do zapamiętania adresu liczby int
   • • •
   wsk = (int*) malloc( sizeof(int) );     // przydzielenie pamięci na liczbę int
   if( wsk == NULL )
      {  printf( "Błąd przydziału pamięci" );  return;
   • • •
   *wsk = 10;                  // przykładowe operacje na dynamicznej liczbie int
   *wsk *= 2;
   printf( "%d", *wsk );
   scanf( "%d", wsk );
   • • •
   free( wsk );                // zwolnienie pamięci przed zakończeniem programu
}
```

Przykład operacji na dynamicznej tablicy o dowolnej ilości elementów:

```
int rozmiar_tablicy;
double *tablica_liczb;
printf( "Ile liczb chcesz wprowadzić: " );
scanf( "%d", &rozmiar_tablicy );
if( tablica_liczb = (double*) calloc( rozmiar_tablicy, sizeof(double) ) )
  {
    for( int i = 0; i < rozmiar_tablicy, i++ );
      *( tablica_liczb+i ) = 100;          // tablica_liczb[ i ] = 100;
    • • •
    free( tablica_liczb );
  }
```

W języku „**C++**" do dynamicznego przydzielania pamięci wygodniej jest wykorzystywać operatory **new** i **delete** :

```
<wskaźnik_na_obiekt> = new  <typ_obiektu> [parametry_inicjacyjne] ;
delete  <wskaźnik_na_obiekt> ;
```

np.

```
int*  wsk ;                  // wskaźnik na zmienną typu całkowitego
wsk = new int ;              // utworzenie nowego obiektu (nowej zmiennej int)
if( wsk != NULL )
  {
    *wsk = 10 ;              // przypisanie wartości  (poprzez wskaźnik)
    printf( "%d" , *wsk );   // wydrukowanie zawartości zmiennej dynam.
    • • •
    delete wsk ;
  }
```

Porównanie utworzenia zwykłej tablicy i tablicy dynamicznej:

```
const  ROZMIAR_TABLICY = 100;
double zwykła_tablica[ ROZMIAR_TABLICY ];

int rozmiar_tablicy;
cout << "Ile liczb chcesz wprowadzić: " ;
cin >> rozmiar_tablicy ;
double *tablica_dynamiczna;
tablica_dynamiczna = new double[ rozmiar_tablicy ];
• • •
delete [ ] tablica_dynamiczna;
```

Przykład operacji na jednej strukturze utworzonej dynamicznie:

```
// ( typ <struct dane_osobowe> był zdefniniowany na poprzednim wykładzie )
struct dane_osobowe *wsk_osoby;
wsk_osoby = (struct dane_osoby*) malloc( sizeof(struct dane_osoby) );
if( wsk_osoby )                                   // if( wsk_osoby != NULL )
   {
      printf( "Podaj nazwisko: " );
      scanf( "%s" ,wsk_osoby -> nazwisko );
      • • •
      printf( "Podaj stypendium: " );
      scanf( "%lf" , &(wsk_osoby -> stypendium) );
      • • •
      free( wsk_osoby );
   }
```

Operacje na dynamicznej tablicy struktur o dowolnej ilości elementów:

```
int rozmiar_tablicy;
struct dane_osobowe  *baza;
printf( "Ile liczb chcesz wprowadzić: " );
scanf( "%d", &rozmiar_tablicy );
baza = (dane_osobowe*) calloc( rozmiar_tablicy, sizeof(dane_osobowe) );
if( baza == NULL )
   {
      printf( "Błąd przydziału pamięci" );
      exit;
   }
• • •
// wczytanie danych kilku osób do utworzonej dynamicznej tablicy
for( int i = 0; i < rozmiar_tablicy, i++ );
   {
      printf( "Podaj nazwisko: " );
      scanf( "%s" , ( baza+i ) -> nazwisko );
      printf( "Podaj stypendium: " );
      scanf( "%lf" , &( ( baza+i ) -> stypendium) );
      • • •
   }
• • •
if( baza != NULL )
   free( baza );              // zwolnienie pamięci przed zakończeniem programu
```