# Създаване на приложения с указатели

# Понятие за указател

Указателят е едно от основните предимства, с които разполагат програмните езици като С и С++. Те позволяват директен достъп до елементи в паметта, както и динамична заделяне на памет. Основната идея при указателите е, че за достъп до данните, записани в паметта са нужни само мястото в паметта и типа на данните, които са записани там. По своя смисъл указателят представлява число. Значението на това число е малко по-различно от до сега разгледаните числови стойности. Това число е 32 битово (съответно за 32 битов компилатор) и има значение на адрес в паметта. Всяка клетка в паметта има свой собствен (пореден) номер. Този номер се нарича адрес.

За по-доброто разбиране на понятието указател, нека се разгледа следната асоциация: Дадено е студентско общежитие. Всяка стая е номерирана с пореден номер. В една стая може да живее само един студент. Управител-домакинът може да разпределя студентите по стаите. Къде в този пример са указателите и какво всичко има общо с програмирането? Студентското общежитие, в случая, е паметта на една компютърна система. Номерацията на стаите са адресите в паметта. Самите стаи са клетките памет. Студентът притежава собствено име (да се приеме, че името на студента е уникално в рамките на общежитието) – това е името на променливата, а самият човек, може да се приеме тогава, е стойността на самата променлива. Ролята на управител-домакина тук е ролята на операционната система – той настанява студентите по стаи и по зададено име може да каже къде се намира даденият студент в общежитието(паметта). Променливата от тип указател е малко по-специална: това е стая, в която няма настанен студент, а има бележка „Студентът се намира в стая Номер...“.

Някои проблеми, които решаваме се реализират по- лесно чрез указатели, а други свързани с динамичното заделяне на памет дори не могат да бъдат реализирани без тях.

Всяка променлива заема памет, а тази памет се намира на определен адрес в компютърната логика. Този адрес може да бъде достъпен чрез оператора &.

Пример:

#include <stdio.h>

int main () {

int var1;

printf("Address of var1 variable: %x\n", &var1 );

return 0;

}

# Какво всъщност е указателя?

Указател е променлива, чиято стойност е адрес на друга променлива, тоест директен достъп до позицията в паметта. Както всяка променлива и константа трябва да декларираме указателя за да може да го използваме.

<тип> \*<име на променлива>;

Тук пот <тип> се има предвид типа на данните, към които ще сочи указателя!

Няколко декларации на указатели:

int \*ip; /\* указател към integer \*/

double \*dp; /\* указател към double \*/

float \*fp; /\* указател към float \*/

char \*ch /\* указател към character \*/

Във всички декларирани указатели имаме една и съща съхранявана информация – адреса на променлива от паметта. Размера им е един и същ и той е равен на размера на long число.

# Използване

Основни действия с указатели са следните:

* Дефиниране на променлива от вида на указател
* Подаване на адреса на променлива към указател
* Достъпване на стойността на адреса, който преди това сме съхранили – чрез използване на унарния оператор ‚\*‘

int var = 20; /\* декларация на реална променлива\*/

int \*ip; /\* декларация на указател \*/

ip = &var; /\* подаване на стойност на указателя адреса на променлива \*/

printf("Address of var variable: %x\n", &var );

/\* адреса запазен в указателя \*/

printf("Address stored in ip variable: %x\n", ip );

/\* достъп до стойността в указателя \*/

printf("Value of \*ip variable: %d\n", \*ip );

# NULL указатели

Добра практика е когато нямаме конкретна стойност за подаване на указателя да се задава нулева стойност, което означава да му се подаде NULL. Това най – често се прави по време на декларация на променливата. В някои библиотеки задаването на стойност NULL се приема като даване на стойност 0 на указателя, но в паметта това не е така, защото нулевата клетка си остава валиден адрес.

int \*ptr = NULL;

printf("The value of ptr is : %x\n", ptr );

В повечето операционни системи програмите нямат достъп до адрес 0 от паметта, защото той е резервиран за самата операционна система. Това обаче се използва като знак, че указателя не сочи към никоя конкретна част от паметта.

За проверка, дали сочи към нещо, указателя може да бъде комбиниран с if оператор:

if(ptr) /\* успешно ако p не е null \*/

if(!ptr) /\* успешно ако p е null \*/

# Детайли свързани с указателите:

1. Следните аритметични оператори могат да се използват с указатели: ++, --, +, -
2. Може да се декларират масиви, които да съхраняват указатели
3. Може да имаме указател към указател
4. Може да подаваме указатели като параметри на функции или да подаваме променливи чрез тяхната референция (адрес)
5. Може да връщаме указатели от функции

# Чести грешки допускани при работа с указатели:

Искаме указател pc да сочи към адреса на променливата с

int c, \*pc;

// pc is address but c is not

pc = c; // Error

// &c is address but \*pc is not

\*pc = &c; // Error

// both &c and pc are addresses

pc = &c; // Not an error

// both c and \*pc are values

\*pc = c; // Not an error

Пример за синтаксис, който обърква начинаещи програмисти

#include <stdio.h>

int main() {

int c = 5;

int \*p = &c;

printf("%d", \*p); // 5

  return 0;

}

Когато имаме \*p=&c не получаваме грешка, защото е еквивалентно на

int \*p:

p = &c;

# Задачи

Решете задачи от семинарните и миналите упр, като използвате указатели където е възможно